Search

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

Перечисления »
« Массивы

- Руководство по PHP
- Справочник языка
- Типы

Change language: Russian

Объекты

Инициализация объекта

Для создания нового объекта, используйте выражение `new`, создающее в переменной экземпляр класса:

```
<?php
class foo
{
function do_foo()
{
echo "Код foo.";
}
}

$bar = new foo;
$bar->do_foo();
?>
```

Полное рассмотрение производится в разделе Классы и Объекты.

Преобразование в объект

Если object преобразовывается в object, объект не изменится. Если значение другого типа преобразовывается в object, создаётся новый экземпляр встроенного класса stdClass. Если значение было **`null`**, новый экземпляр будет пустым. Массивы преобразуются в object с именами полей, названными согласно ключам массива и соответствующими им значениям. Обратите внимание, что в этом случае до PHP 7.2.0 числовые ключи не будут доступны, пока не проитерировать объект.

```
<?php
$obj = (object) array('1' => 'foo');
var_dump(isset($obj->{'1'})); // выводит 'bool(true)', начиная с PHP 7.2.0; 'bool(false)' ранее
var_dump(key($obj)); // выводит 'string(1) "1"', начиная с PHP 7.2.0; 'int(1)' ранее
?>
```

При преобразовании любого другого значения, оно будет помещено в поле с именем `scalar` соответствующему типу.

```
<?php
$obj = (object) 'привет';
echo $obj->scalar; // выведет 'привет'
?>
```


User Contributed Notes 8 notes

660
helpful at stranger dot com ¶
12 years ago
By far the easiest and correct way to instantiate an empty generic php object that you can then modify for whatever purpose you choose:

```
<?php $genericObject = new stdClass(); ?>
```

I had the most difficult time finding this, hopefully it will help someone else!

252
Anthony ¶
8 years ago

In PHP 7 there are a few ways to create an empty object:

```
<?php
$obj1 = new \stdClass; // Instantiate stdClass object
$obj2 = new class{}; // Instantiate anonymous class
$obj3 = (object)[]; // Cast empty array to object

var_dump($obj1); // object(stdClass)#1 (0) {}
var_dump($obj2); // object(class@anonymous)#2 (0) {}
var_dump($obj3); // object(stdClass)#3 (0) {}
?>
```

$obj1 and $obj3 are the same type, but $obj1 !== $obj3. Also, all three will json_encode() to a simple JS object {}:

```
<?php
echo json_encode([
new \stdClass,
new class{},
(object)[],
]);
?>
```

Outputs: [{},{},{}]

up
down
44

twitter/matt2000 ¶

8 years ago

As of PHP 5.4, we can create stdClass objects with some properties and values using the more beautiful form:

```
<?php
$object = (object) [
'propertyOne' => 'foo',
'propertyTwo' => 42,
];
?>
```

up
down
21

developer dot amankr at gmail dot com (Aman Kuma) ¶

7 years ago

```
<!--Example shows how to convert array to stdClass Object and how to access its value for display -->
<?php
$num = array("Garha","sitamarhi","canada","patna"); //create an array
$obj = (object)$num; //change array to stdClass object

echo "<pre>";
print_r($obj); //stdClass Object created by casting of array

$newobj = new stdClass();//create a new
$newobj->name = "India";
$newobj->work = "Development";
$newobj->address="patna";

$new = (array)$newobj;//convert stdClass to array
echo "<pre>";
print_r($new); //print new object

##How deals with Associative Array

$test = [Details=>['name','roll number','college','mobile'],values=>['Naman Kumar','100790310868','Pune
college','9988707202']];
```

```
$val = json_decode(json_encode($test),false);//convert array into stdClass object

echo "<pre>";
print_r($val);

echo ((is_array($val) == true ? 1 : 0 ) == 1 ? "array" : "not an array" )."</br>"; // check whether it is array or not
echo ((is_object($val) == true ? 1 : 0 ) == 1 ? "object" : "not an object" );//check whether it is object or not
?>
```

up
down
39

Ashley Dambra ¶

9 years ago

Here a new updated version of 'stdObject' class. It's very useful when extends to controller on MVC design pattern, user can create it's own class.

Hope it help you.

```
<?php
class stdObject {
public function __construct(array $arguments = array()) {
if (!empty($arguments)) {
foreach ($arguments as $property => $argument) {
$this->{$property} = $argument;
}
}
}

public function __call($method, $arguments) {
$arguments = array_merge(array("stdObject" => $this), $arguments); // Note: method argument 0 will always referred to the main class ($this).
if (isset($this->{$method}) && is_callable($this->{$method})) {
return call_user_func_array($this->{$method}, $arguments);
} else {
throw new Exception("Fatal error: Call to undefined method stdObject::{$method}()");
}
}
}

// Usage.

$obj = new stdObject();
$obj->name = "Nick";
$obj->surname = "Doe";
$obj->age = 20;
$obj->adresse = null;

$obj->getInfo = function($stdObject) { // $stdObject referred to this object (stdObject).
echo $stdObject->name . " " . $stdObject->surname . " have " . $stdObject->age . " yrs old. And live in " . $stdObject->adresse;
};

$func = "setAge";
$obj->{$func} = function($stdObject, $age) { // $age is the first parameter passed when calling this method.
$stdObject->age = $age;
};

$obj->setAge(24); // Parameter value 24 is passing to the $age argument in method 'setAge()'.

// Create dynamic method. Here i'm generating getter and setter dynimically
// Beware: Method name are case sensitive.
foreach ($obj as $func_name => $value) {
```

```
if (!$value instanceOf Closure) {

$obj->{"set" . ucfirst($func_name)} = function($stdObject, $value) use ($func_name) { // Note: you can also use keyword
'use' to bind parent variables.
$stdObject->{$func_name} = $value;
};

$obj->{"get" . ucfirst($func_name)} = function($stdObject) use ($func_name) { // Note: you can also use keyword 'use' to
bind parent variables.
return $stdObject->{$func_name};
};

}
}

$obj->setName("John");
$obj->setAdresse("Boston");

$obj->getInfo();
?>
```

up
down
10

Mithras ¶

15 years ago

```
In response to harmor: if an array contains another array as a value, you can recursively convert all arrays with:

<?php
function arrayToObject( $array ){
foreach( $array as $key => $value ){
if( is_array( $value ) ) $array[ $key ] = arrayToObject( $value );
}
return (object) $array;
}
?>
```

up
down
2

qeremy [atta] gmail [dotta] com ¶

11 years ago

```
Do you remember some JavaScript implementations?

// var timestamp = (new Date).getTime();

Now it's possible with PHP 5.4.*;

<?php
class Foo
{
public $a = "I'm a!";
public $b = "I'm b!";
public $c;

public function getB() {
return $this->b;
}

public function setC($c) {
$this->c = $c;
return $this;
}
```

```
public function getC() {
return $this->c;
}
}

print (new Foo)->a; // I'm a!
print (new Foo)->getB(); // I'm b!
?>

or

<?php
// $_GET["c"] = "I'm c!";
print (new Foo)
->setC($_GET["c"])
->getC(); // I'm c!
?>
```

up
down
5

mailto dot aurelian at gmail dot com ¶

14 years ago

```
You can create [recursive] objects with something like:
<?php
$literalObjectDeclared = (object) array(
'foo' => (object) array(
'bar' => 'baz',
'pax' => 'vax'
),
'moo' => 'ui'
);
print $literalObjectDeclared->foo->bar; // outputs "baz"!
?>
```

＋add a note

- Privacy policy