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

Анонимные классы

Анонимные классы полезны, когда нужно создать простые, одноразовые объекты.

```
<?php

// Использование явного класса
class Logger
{
public function log($msg)
{
echo $msg;
}
}

$util->setLogger(new Logger());

// Использование анонимного класса
$util->setLogger(new class {
public function log($msg)
{
echo $msg;
}
});
```

Они могут передавать аргументы в конструкторы, расширять другие классы, реализовывать интерфейсы и использовать трейты как обычный класс:

```
<?php

class SomeClass {}
interface SomeInterface {}
trait SomeTrait {}

var_dump(new class(10) extends SomeClass implements SomeInterface {
private $num;

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

use SomeTrait;
});
```

Результат выполнения приведённого примера:

```
object(class@anonymous)#1 (1) {
  ["Command line code0x104c5b612":"class@anonymous":private]=>
  int(10)
}
```

Вложение анонимного класса в другой класс не даёт ему доступ к закрытым или защищённым методам и свойствам этого внешнего класса. Для того, чтобы использовать защищённые свойства и методы внешнего класса, анонимный класс может расширить внешний класс. Чтобы использовать закрытые свойства внешнего класса в анонимном классе, их нужно передать в конструктор:

```
<?php

class Outer
{
private $prop = 1;
protected $prop2 = 2;

protected function func1()
{
return 3;
}

public function func2()
{
return new class($this->prop) extends Outer {
private $prop3;

public function __construct($prop)
{
$this->prop3 = $prop;
}

public function func3()
{
return $this->prop2 + $this->prop3 + $this->func1();
}
};
}
}

echo (new Outer)->func2()->func3();
```

Результат выполнения приведённого примера:

```
6
```

Все объекты, созданные одним и тем же объявлением анонимного класса, являются экземплярами этого самого класса.

```
<?php
function anonymous_class()
{
return new class {};
}

if (get_class(anonymous_class()) === get_class(anonymous_class())) {
echo 'Тот же класс';
} else {
echo 'Другой класс';
}
```

Результат выполнения приведённого примера:

```
Тот же класс
```

> **Замечание**:
>
> Обратите внимание, что анонимным классам присваиваются имена движком PHP, как показано в примере ниже. Это имя следует рассматривать как особенность реализации, на которую не следует полагаться.
>
> ```
> <?php
> echo get_class(new class {});
> ```
>
> Вывод приведённого примера будет похож на:
>
> ```
> class@anonymous/in/oNi1A0x7f8636ad2021
> ```

+ add a note

User Contributed Notes 9 notes

up
down
45
***Anonymous* ¶**
7 years ago

```
Below three examples describe anonymous class with very simple and basic but quite understandable example

<?php
// First way - anonymous class assigned directly to variable
$ano_class_obj = new class{
public $prop1 = 'hello';
public $prop2 = 754;
const SETT = 'some config';

public function getValue()
{
// do some operation
return 'some returned value';
}

public function getValueWithArgu($str)
{
// do some operation
return 'returned value is '.$str;
}
};

echo "\n";

var_dump($ano_class_obj);
echo "\n";

echo $ano_class_obj->prop1;
echo "\n";

echo $ano_class_obj->prop2;
echo "\n";

echo $ano_class_obj::SETT;
echo "\n";

echo $ano_class_obj->getValue();
echo "\n";

echo $ano_class_obj->getValueWithArgu('OOP');
echo "\n";

echo "\n";

// Second way - anonymous class assigned to variable via defined function
$ano_class_obj_with_func = ano_func();

function ano_func()
{
return new class {
public $prop1 = 'hello';
public $prop2 = 754;
const SETT = 'some config';

public function getValue()
{
// do some operation
```

```
return 'some returned value';
}

public function getValueWithArgu($str)
{
// do some operation
return 'returned value is '.$str;
}
};
}

echo "\n";

var_dump($ano_class_obj_with_func);
echo "\n";

echo $ano_class_obj_with_func->prop1;
echo "\n";

echo $ano_class_obj_with_func->prop2;
echo "\n";

echo $ano_class_obj_with_func::SETT;
echo "\n";

echo $ano_class_obj_with_func->getValue();
echo "\n";

echo $ano_class_obj_with_func->getValueWithArgu('OOP');
echo "\n";

echo "\n";

// Third way - passing argument to anonymous class via constructors
$arg = 1; // we got it by some operation
$config = [2, false]; // we got it by some operation
$ano_class_obj_with_arg = ano_func_with_arg($arg, $config);

function ano_func_with_arg($arg, $config)
{
return new class($arg, $config) {
public $prop1 = 'hello';
public $prop2 = 754;
public $prop3, $config;
const SETT = 'some config';

public function __construct($arg, $config)
{
$this->prop3 = $arg;
$this->config =$config;
}

public function getValue()
{
// do some operation
return 'some returned value';
}

public function getValueWithArgu($str)
{
// do some operation
```

```
return 'returned value is '.$str;
}
};
}

echo "\n";

var_dump($ano_class_obj_with_arg);
echo "\n";

echo $ano_class_obj_with_arg->prop1;
echo "\n";

echo $ano_class_obj_with_arg->prop2;
echo "\n";

echo $ano_class_obj_with_arg::SETT;
echo "\n";

echo $ano_class_obj_with_arg->getValue();
echo "\n";

echo $ano_class_obj_with_arg->getValueWithArgu('OOP');
echo "\n";

echo "\n";
```
up
down
22
ytubeshareit at gmail dot com ¶
6 years ago

Anonymous classes are syntax sugar that may appear deceiving to some.
The 'anonymous' class is still parsed into the global scope, where it is auto assigned a name, and every time the class is needed, that global class definition is used. Example to illustrate....

The anonymous class version...
```
<?php

function return_anon(){
return new class{
public static $str="foo";
};
}
$test=return_anon();
echo $test::$str; //ouputs foo

//we can still access the 'anon' class directly in the global scope!
$another=get_class($test); //get the auto assigned name
echo $another::$str; //outputs foo
?>
```

The above is functionally the same as doing this....
```
<?php
class I_named_this_one{
public static $str="foo";
}
function return_not_anon(){
return 'I_named_this_one';
}
$clzz=return_not_anon();//get class name
echo $clzz::$str;
?>
```

up
down
9
sebastian.wasser at gmail ¶
5 years ago

```
I wanted to share my findings on static properties of anonymous classes.

So, given an anonymous class' object generating function like this:

<?php
function nc () {
return new class {
public static $prop = [];
};
}
?>

Getting a new object and changing the static property:

<?php
$a = nc();
$a::$prop[] = 'a';

var_dump($a::$prop);
// array(1) {
// [0] =>
// string(1) "a"
// }
?>

Now getting another object and changing the static property will change the original one, meaning that the static property
is truly static:

<?php
$b = nc();
$b::$prop[] = 'b';

var_dump($b::$prop); // Same as var_dump($a::$prop);
// array(2) {
// [0] =>
// string(1) "a"
// [1] =>
// string(1) "b"
// }

assert($a::$prop === $b::$prop); // true
?>
```

up
down
5
joey ¶
4 years ago

```
The only way to type hint this would appear to be as object.

If you need multiple instances of an anonymous class in a function you can use:

$class = function(string $arg):object {
return new class($arg) {
public function __construct(string $arg) {
$this->ow = $arg;
}
};
};

Though for the sake of structure it's ill advised to do something like this outside of a single scope or that's used
across multiple files. If you class is only used in one scope however then it's probably not a code mess problem.
```
up
down
6
j.m | jamesweb | ca ¶
6 years ago
```
/* I like the idea of OneShot classes.
Thanks to that Anonymous bro\sist for precising
new class( $a, $b )
---------

If you are looking for "Delayed OneShot Anonymous Classes" for any reason (like the reason: loading files in a readable
manner while not using autoload), it would probably look something like this; */

$u = function()use(&$u){
$u = new class{private $name = 'Utils';};
};

$w = function(&$rewrite)use(&$w){
$w = null;
$rewrite = new class{private $name = 'DataUtils';};
};

// Usage;
var_dump(
array(
'Delayed',
'( Self Destructive )',
'Anonymous Class Creation',
array(
'Before ( $u )' => $u,
'Running ( $u() )' => $u(),
'After ( $u )' => $u,
),
0,0,
0,0,
0,0,
'Delayed',
'( Overwriting && Self Destructive )',
'Anonymous Class Creation',
array(
'Before ( $w )' => $w,
'Running ( $w($u) )' => $w($u),
'After ( $w )' => $w,
'After ( $u )' => $u
)
)
);

// btw : oh shoot I failed a spam challenge
```
up
down
2
razvan_bc at yahoo dot com ¶
3 years ago
```
you can try these

<?php
```

```
$oracle=&$_['nice_php'];
$_['nice_php']=(function(){
return new class{
public static function say($msg){
echo $msg;
}

public static function sp(){
echo self::say(' ');
}

};
});

/*
$_['nice_php']()::say('Hello');
$_['nice_php']()::sp();
$_['nice_php']()::say('World');
$_['nice_php']()::sp();
$_['nice_php']()::say('!');
//almost the same code bottom
*/

$oracle()::say('Hello');
$oracle()::sp();
$oracle()::say('World');
$oracle()::sp();
$oracle()::say('!');
?>
```

up
down
1

piotr at maslosoft dot com ¶

6 years ago

```
Please note that class name returned by `get_class` might contain null bytes, as is the case in my version of PHP (7.1.4).

Name will change when class starting line or it's body is changed.

Yes, name is implementation detail that should not be relied upon, but in some rare use cases it is required (annotating anonymous class).
```

up
down
-11

solobot ¶

6 years ago

```
eval() is workaround for generating multiple anonymous classes with static properties in loop
<?php
public function generateClassMap()
{
foreach ($this->classMap as $tableName => $class)
{
$c = null;
eval('$c = new class extends \common\MyStaticClass {
public static $tableName;
public static function tableName()
{
return static::$tableName;
}
};');
$c::$tableName = $this->replicationPrefix.$tableName;
$this->classMap[$tableName] = $c;
}
}
?>
thus every class will have its own $tableName instead of common ancestor.
```
up
down
-21
primipilus13 at gmail dot com ¶
7 years ago
```
<?php

// using constructor and extends in anonymous class

class A
{
private $name;

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

public function getName()
{
return $this->name;
}
}

$b = new class('anonymous') extends A
{
public function getName()
{
return parent::getName() . ' class';
}
};

echo $b->getName(), PHP_EOL;

// result: anonimous class
```
+add a note