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

Пространства имён и динамические особенности языка

(PHP 5 >= 5.3.0, PHP 7, PHP 8)

На реализацию пространств имён в PHP повлияли и динамические свойства языка. Поэтому, чтобы преобразовать код наподобие следующего примера в код, который будет работать внутри пространства имён:…

Пример #1 Динамически доступные элементы

example1.php:

```
<?php

class classname
{
function __construct()
{
echo __METHOD__,"\n";
}
}

function funcname()
{
echo __FUNCTION__,"\n";
}

const constname = "global";

$a = 'classname';
$obj = new $a; // Выводит classname::__construct
$b = 'funcname';
$b(); // Выводит funcname
echo constant('constname'), "\n"; // Выводит global

?>
```

…нужно указать абсолютное имя (имя класса с префиксом пространства имён). Обратите внимание, поскольку между полным и абсолютным именем внутри динамического имени класса, функции или константы нет разницы, начальный обратный слеш не нужен.

Пример #2 Динамически доступные элементы пространства имён

```
<?php

namespace namespacename;

class classname
{
function __construct()
{
echo __METHOD__,"\n";
}
}

function funcname()
{
echo __FUNCTION__,"\n";
}

const constname = "namespaced";

include 'example1.php';
```

```
$a = 'classname';
$obj = new $a; // Выводит classname::__construct
$b = 'funcname';
$b(); // Выводит funcname
echo constant('constname'), "\n"; // Выводит global

/* Обратите внимание, что в двойных кавычках символ обратного слеша нужно заэкранировать. Например,
"\\namespacename\\classname" */
$a = '\namespacename\classname';
$obj = new $a; // Выводит namespacename\classname::__construct
$a = 'namespacename\classname';
$obj = new $a; // Тоже выводит namespacename\classname::__construct
$b = 'namespacename\funcname';
$b(); // Выводит namespacename\funcname
$b = '\namespacename\funcname';
$b(); // Тоже выводит namespacename\funcname
echo constant('\namespacename\constname'), "\n"; // Выводит namespaced
echo constant('namespacename\constname'), "\n"; // Тоже выводит namespaced

?>
```

Обязательно прочитайте примечание об экранировании имён пространства имён в строках.

+ add a note

User Contributed Notes 8 notes

up
down
75
Alexander Kirk ¶
12 years ago
When extending a class from another namespace that should instantiate a class from within the current namespace, you need to pass on the namespace.

```
<?php // File1.php
namespace foo;
class A {
public function factory() {
return new C;
}
}
class C {
public function tell() {
echo "foo";
}
}
?>

<?php // File2.php
namespace bar;
class B extends \foo\A {}
class C {
public function tell() {
echo "bar";
}
}
?>

<?php
include "File1.php";
include "File2.php";
$b = new bar\B;
$c = $b->factory();
$c->tell(); // "foo" but you want "bar"
?>
```

You need to do it like this:

When extending a class from another namespace that should instantiate a class from within the current namespace, you need to pass on the namespace.

```
<?php // File1.php
namespace foo;
class A {
protected $namespace = __NAMESPACE__;
public function factory() {
$c = $this->namespace . '\C';
return new $c;
}
}
class C {
public function tell() {
echo "foo";
}
}
?>

<?php // File2.php
namespace bar;
class B extends \foo\A {
protected $namespace = __NAMESPACE__;
}
class C {
public function tell() {
echo "bar";
}
}
?>

<?php
include "File1.php";
include "File2.php";
$b = new bar\B;
$c = $b->factory();
$c->tell(); // "bar"
?>
```

(it seems that the namespace-backslashes are stripped from the source code in the preview, maybe it works in the main view. If not: fooA was written as \foo\A and barB as bar\B)

up
down
9

Daan ¶

4 years ago

Important to know is that you need to use the *fully qualified name* in a dynamic class name. Here is an example that emphasizes the difference between a dynamic class name and a normal class name.

```
<?php
namespace namespacename\foo;

class classname
{
```

```
function __construct()
{
echo 'bar';
}
}

$a = '\namespacename\foo\classname'; // Works, is fully qualified name
$b = 'namespacename\foo\classname'; // Works, is treated as it was with a prefixed "\"
$c = 'foo\classname'; // Will not work, it should be the fully qualified name

// Use dynamic class name
new $a; // bar
new $b; // bar
new $c; // [500]: / - Uncaught Error: Class 'foo\classname' not found in

// Use normal class name
new \namespacename\foo\classname; // bar
new namespacename\foo\classname; // [500]: / - Uncaught Error: Class 'namespacename\foo\namespacename\foo\classname' not found
new foo\classname; // [500]: / - Uncaught Error: Class 'namespacename\foo\foo\classname' not found
```

up
down
6

museyib dot e at gmail dot com ¶

4 years ago

Be careful when using dynamic accessing namespaced elements. If you use double-quote backslashes will be parsed as escape character.

```
<?php
$a="\namespacename\classname"; //Invalid use and Fatal error.
$a="\\namespacename\\classname"; //Valid use.
$a='\namespacename\classname'; //Valid use.
?>
```

up
down
16

guilhermeblanco at php dot net ¶

14 years ago

Please be aware of FQCN (Full Qualified Class Name) point.
Many people will have troubles with this:

```
<?php

// File1.php
namespace foo;

class Bar { ... }

function factory($class) {
return new $class;
}

// File2.php
$bar = \foo\factory('Bar'); // Will try to instantiate \Bar, not \foo\Bar

?>
```

To fix that, and also incorporate a 2 step namespace resolution, you can check for \ as first char of $class, and if not present, build manually the FQCN:

```
<?php
```

```
// File1.php
namespace foo;

function factory($class) {
if ($class[0] != '\\') {
echo '->';
$class = '\\' . __NAMESPACE__ . '\\' . $class;
}

return new $class();
}

// File2.php
$bar = \foo\factory('Bar'); // Will correctly instantiate \foo\Bar

$bar2 = \foo\factory('\anotherfoo\Bar'); // Wil correctly instantiate \anotherfoo\Bar

?>
```

up
down
6
akhoondi+php at gmail dot com ¶
10 years ago

It might make it more clear if said this way:

One must note that when using a dynamic class name, function name or constant name, the "current namespace", as in http://www.php.net/manual/en/language.namespaces.basics.php is global namespace.

One situation that dynamic class names are used is in 'factory' pattern. Thus, add the desired namespace of your target class before the variable name.

namespaced.php

```
<?php
// namespaced.php
namespace Mypackage;
class Foo {
public function factory($name, $global = FALSE)
{
if ($global)
$class = $name;
else
$class = 'Mypackage\\' . $name;
return new $class;
}
}

class A {
function __construct()
{
echo __METHOD__ . "<br />\n";
}
}
class B {
function __construct()
{
echo __METHOD__ . "<br />\n";
}
}
?>
```

global.php

```
<?php
```

```
// global.php
class A {
function __construct()
{
echo __METHOD__;
}
}
?>

index.php
<?php
// index.php
namespace Mypackage;
include('namespaced.php');
include('global.php');

$foo = new Foo();

$a = $foo->factory('A'); // Mypackage\A::__construct
$b = $foo->factory('B'); // Mypackage\B::__construct

$a2 = $foo->factory('A',TRUE); // A::__construct
$b2 = $foo->factory('B',TRUE); // Will produce : Fatal error: Class 'B' not found in ...namespaced.php on line ...
?>
```

up

down

2

m dot mannes at gmail dot com ¶

6 years ago

```
Case you are trying call a static method that's the way to go:

<?php
class myClass
{
public static function myMethod()
{
return "You did it!\n";
}
}

$foo = "myClass";
$bar = "myMethod";

echo $foo::$bar(); // prints "You did it!";
?>
```

up

down

2

anisgazig at gmail dot com ¶

2 years ago

```
<?php

//single or double quotes with single or double backslash in dynamic namespace class.

namespace Country_Name{
class Mexico{
function __construct(){
echo __METHOD__,"<br>";
}
}

$a = 'Country_Name\Mexico';//Country_Name\Mexico::__construct
$a = "Country_Name\Mexico";
//Country_Name\Mexico::__construct
$a = '\Country_Name\Mexico';
//Country_Name\Mexico::__construct
$a = "\Country_Name\Mexico";
//Country_Name\Mexico::__construct
$a = "\\Country_Name\\Mexico";
//Country_Name\Mexico::__construct
$o = new $a;

}

/* if your namespace name or class name start with lowercase n then you should be alart about the use of single or double
quotes with backslash */

namespace name_of_country{
class Japan{
function __construct()
{
echo __METHOD__,"<br>";
}

}

$a = 'name_of_country\Japan';
//name_of_country\Japan::__construct
$a = "name_of_country\Japan";
//name_of_country\Japan::__construct
$a = '\name_of_country\Japan';
//name_of_country\Japan::__construct
//$a = "\name_of_country\Japan";
//Fatal error: Uncaught Error: Class ' ame_of_country\Japan' not found
//In this statement "\name_of_country\Japan" means -first letter n with "\ == new line("\n). for fix it we can use double
back slash or single quotes with single backslash.
$a = "\\name_of_country\\Japan";
//name_of_country\Japan::__construct
$o = new $a;
}

namespace Country_Name{
class name{
function __construct(){
echo __METHOD__,"<br>";
}
}

$a = 'Country_Name\name';
//Country_Name\Norway::__construct
$a = "Country_Name\name";
//Country_Name\Norway::__construct
$a = '\Country_Name\name';
//Country_Name\Norway::__construct
//$a = "\Country_Name\name";
//Fatal error: Uncaught Error: Class '\Country_Name ame' not found

//In this statement "\Country_Name\name" at class name's first letter n with "\ == new line("\n). for fix it we can use
double back slash or single quotes with single backslash
$a = "\\Country_Name\\name";
//Country_Name\name::__construct
$o = new $a;

}
```

```
//"\n == new line are case insensitive so "\N could not affected

?>
```
up
down
1
scott at intothewild dot ca ¶
14 years ago
```
as noted by guilhermeblanco at php dot net,

<?php

// fact.php

namespace foo;

class fact {

public function create($class) {
return new $class();
}
}

?>

<?php

// bar.php

namespace foo;

class bar {
...
}

?>

<?php

// index.php

namespace foo;

include('fact.php');

$foofact = new fact();
$bar = $foofact->create('bar'); // attempts to create \bar
// even though foofact and
// bar reside in \foo

?>
```
+ add a note