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

Основы »
« Подпространства имён

- Руководство по PHP
- Справочник языка
- Пространства имён

Change language: Russian

Описание нескольких пространств имён в одном файле

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

В одном файле разрешено объявлять несколько пространств имён. Есть два разрешённых синтаксиса:

Пример #1 Описание нескольких пространств имён, простой синтаксис

```
<?php

namespace MyProject;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }

namespace AnotherProject;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }

?>
```

Этот синтаксис не рекомендован для комбинирования пространств имён в одном файле. Вместо него лучше пользоваться альтернативным синтаксисом со скобками.

Пример #2 Описание нескольких пространств имён, синтаксис со скобками

```
<?php

namespace MyProject {
const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }
}

namespace AnotherProject {
const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }
}

?>
```

Практика написания кода настоятельно не рекомендует объединять пространства имён в одном файле. Главный сценарий того, когда это потребуется, — объединение нескольких PHP-файлов в один файл.

Для объединения кода в глобальном пространстве имён с кодом в других пространствах имён пользуются только синтаксисом со скобками. Глобальный код должен быть помещён в конструкцию описания пространства имён без указания имени:

Пример #3 Описание глобального и обычного пространства имён в одном файле

```
<?php

namespace MyProject {
const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }
}
```

```
namespace { // Глобальный код
session_start();
$a = MyProject\connect();
echo MyProject\Connection::start();
}

?>
```

Никакой PHP-код нельзя размещать за пределами скобок пространства имён, кроме начального выражения declare.

Пример #4 Описание глобального и обычного пространства имён в одном файле

```
<?php

declare(encoding='UTF-8');
namespace MyProject {
const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }
}

namespace { // Глобальный код
session_start();
$a = MyProject\connect();
echo MyProject\Connection::start();
}

?>
```


User Contributed Notes 6 notes

96
leaksin [at] gmail [dot] com ¶
10 years ago
using of global namespaces and multiple namespaces in one PHP file increase the complexity and decrease readability of the code.
Let's try not use this scheme even it's very necessary (although there is not)

49
jigar dot vy at gmail dot com ¶
8 years ago
```
<?php

// You cannot mix bracketed namespace declarations with unbracketed namespace declarations - will result in a Fatal error

namespace a;

echo "I belong to namespace a";

namespace b {
echo "I'm from namespace b";
}
```


27
Rahul Sonar ¶
8 years ago
```
<?php
```

```
//Namespace can be used in this way also
namespace MyProject {

function connect() { echo "ONE"; }
Sub\Level\connect();
}

namespace MyProject\Sub {

function connect() { echo "TWO"; }
Level\connect();
}

namespace MyProject\Sub\Level {

function connect() { echo "THREE"; }
\MyProject\Sub\Level\connect(); // OR we can use this as below
connect();
}
```

up
down
6
dominic_mayers at yahoo dot com ¶
7 years ago

If you have the habit to always use the closing PHP tag "?>" in your test files, remember that with the bracketed syntax code outside the brackets, including new lines outside the PHP tags, is not allowed. In particular, even though PHP sees a new line after the closing tag as a part of the line and eats it, some editors, such as Gedit, Gvim, Vim and Nano in Ubuntu, will add yet another new line after this new line and this will create an error.

up
down
2
dauser at daexample dot com ¶
6 years ago

There are rational examples of where the ability to blend multiple namespaces into a single file is not only desirable but also absolutely necessary. An example of where this ability is useful is over in the very popular phpseclib library where they are PSR-4 compliant but, in order to be compliant, they have to read a directory of files to know what classes are available so that the autoloader can load the correct files. If they, instead, just bundled the defaults into one file using this mechanism already supported by PHP core, there would be no need to do extraneous scanning of the file system.

That's just one legitimate use-case where strict compliance with PSRs gets in the way of good software development.

up
down
6
Ishan Fernando ¶
8 years ago

```
//call same named function using namespace

//food.php

<?php
namespace Food;

require ('Apple.php');
require('Orange.php');

use Apples;
use Oranges;

Apples\eat();
Oranges\eat();
?>
```

```
//Apple.php
<?php
namespace Apples;

function eat()
{
echo "eat apple";
}
?>

//Orange.php
<?php
namespace Oranges;

function eat()
{
echo "eat Orange";
}
?>
```

＋add a note