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

Глобальное пространство »
« Ключевое слово namespace и константа __NAMESPACE__

- Руководство по PHP
- Справочник языка
- Пространства имён

Change language: Russian

Пространства имён: псевдонимирование и импорт

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

Способность ссылаться на внешнее абсолютное имя по псевдониму или импортировать внешние абсолютные имена — это важное свойство пространств имён. Это похоже на способность файловых систем на основе Unix создавать символические ссылки на файл или директорию.

PHP умеет создавать псевдонимы или импортировать константы, функции, классы, интерфейсы, трейты, перечисления и пространства имён.

Псевдоним имени создают через ключевое слово `use`. Вот пример, который показывает 5 типов импорта:

Пример #1 Импорт или псевдонимирование через ключевое слово use

```php
<?php

namespace foo;

use My\Full\Classname as Another;

// Это то же, что и My\Full\NSname as NSname
use My\Full\NSname;

// Импортирование глобального класса
use ArrayObject;

// Импортирование функции
use function My\Full\functionName;

// Создание псевдонима функции
use function My\Full\functionName as func;

// Импортирование константы
use const My\Full\CONSTANT;

$obj = new namespace\Another; // Создаёт экземпляр класса foo\Another
$obj = new Another; // Создаёт объект класса My\Full\Classname
NSname\subns\func(); // Вызывает функцию My\Full\NSname\subns\func

$a = new ArrayObject(array(1)); // Создаёт объект класса ArrayObject
// без выражения use ArrayObject был бы создан объект класса foo\ArrayObject

func(); // вызывает функцию My\Full\functionName
echo CONSTANT; // выводит содержимое константы My\Full\CONSTANT

?>
```

Обратите внимание, что именам внутри пространства имён (абсолютным именам пространств имён, которые содержат разделитель пространств имён, например `Foo\Bar`, в отличие от глобальных имён, которые его не содержат, например `FooBar`) начальный обратный слеш (\) не нужен и не рекомендован, поскольку импортируемые имена должны быть абсолютными и не обрабатываются относительно текущего пространства имён.

PHP дополнительно поддерживает удобное сокращение для задания нескольких операторов use в одной и той же строке

Пример #2 Импорт или создание псевдонима через ключевое слово use, комбинирование нескольких выражений

```php
<?php

use My\Full\Classname as Another, My\Full\NSname;

$obj = new Another; // Создаёт объект класса My\Full\Classname
```

```php
NSname\subns\func(); // Вызывает функцию My\Full\NSname\subns\func

?>
```

Импорт выполняется во время компиляции поэтому не влияет на имена динамических классов, функций или констант.

Пример #3 Импорт и динамические имена

```php
<?php

use My\Full\Classname as Another, My\Full\NSname;

$obj = new Another; // Создаёт объект класса My\Full\Classname
$a = 'Another';
$obj = new $a; // Создаёт объект класса Another

?>
```

В дополнение, импорт распространяется только на неполные и полные имена. На абсолютные имена операция импорта не влияет.

Пример #4 Импортирование и абсолютные имена

```php
<?php

use My\Full\Classname as Another, My\Full\NSname;

$obj = new Another; // Создаёт объект класса My\Full\Classname
$obj = new \Another; // Создаёт объект класса Another
$obj = new Another\thing; // Создаёт объект класса My\Full\Classname\thing
$obj = new \Another\thing; // Создаёт объект класса Another\thing

?>
```

Обзор правил для импорта

Ключевое слово `use` должно быть указано в самом начале файла (в глобальной области) или внутри объявления пространства имён. Это нужно, потому что импорт выполняется во время компиляции, а не во время исполнения, поэтому его нельзя ограничить блоком кода. Следующий пример показывает недопустимое указание ключевого слова `use`:

Пример #5 Недопустимое правило импорта

```php
<?php

namespace Languages;

function toGreenlandic()
{
use Languages\Danish;

//...
}

?>
```

> **Замечание**:
>
> Правила импорта задают на каждый файл отдельно. Поэтому присоединяемые файлы *НЕ* будут наследовать правила импорта из родительского файла.

Групповые объявления через ключевое слово `use`

Классы, функции и константы, импортируемые из одного и того же пространства имён (namespace), разрешено

группировать в одном выражении с ключевым словом use.

```php
<?php

use some\namespace\ClassA;
use some\namespace\ClassB;
use some\namespace\ClassC as C;

use function some\namespace\fn_a;
use function some\namespace\fn_b;
use function some\namespace\fn_c;

use const some\namespace\ConstA;
use const some\namespace\ConstB;
use const some\namespace\ConstC;

// Эквивалентно следующему групповому объявлению с ключевым словом use
use some\namespace\{ClassA, ClassB, ClassC as C};
use function some\namespace\{fn_a, fn_b, fn_c};
use const some\namespace\{ConstA, ConstB, ConstC};
```


User Contributed Notes 21 notes

166
***dominic_mayers at yahoo dot com* ¶**
7 years ago
The keyword "use" has been recycled for three distinct applications:
1- to import/alias classes, traits, constants, etc. in namespaces,
2- to insert traits in classes,
3- to inherit variables in closures.
This page is only about the first application: importing/aliasing. Traits can be inserted in classes, but this is different from importing a trait in a namespace, which cannot be done in a block scope, as pointed out in example 5. This can be confusing, especially since all searches for the keyword "use" are directed to the documentation here on importing/aliasing.

140
***anon* ¶**
10 years ago
The <?php use ?> statement does not load the class file. You have to do this with the <?php require ?> statement or by using an autoload function.

44
***Mawia HL* ¶**
6 years ago
Here is a handy way of importing classes, functions and conts using a single use keyword:

```
<?php
use Mizo\Web\ {
Php\WebSite,
Php\KeyWord,
Php\UnicodePrint,
JS\JavaScript,
function JS\printTotal,
function JS\printList,
const JS\BUAIKUM,
const JS\MAUTAM
};
```

```
?>
```
up
down
76
k at webnfo dot com ¶
10 years ago
Note that you can not alias global namespace:

```
use \ as test;

echo test\strlen('');
```

won't work.
up
down
29
xzero at elite7hackers dot net ¶
6 years ago
I couldn't find answer to this question so I tested myself.
I think it's worth noting:

```
<?php
use ExistingNamespace\NonExsistingClass;
use ExistingNamespace\NonExsistingClass as whatever;
use NonExistingNamespace\NonExsistingClass;
use NonExistingNamespace\NonExsistingClass as whatever;
?>
```

None of above will actually cause errors unless you actually try to use class you tried to import.

```
<?php
// And this code will issue standard PHP error for non existing class.
use ExistingNamespace\NonExsistingClass as whatever;
$whatever = new whatever();
?>
```
up
down
16
me at ruslanbes dot com ¶
7 years ago
Note the code `use ns1\c1` may refer to importing class `c1` from namespace `ns1` as well as importing whole namespace `ns1\c1` or even import both of them in one line. Example:

```
<?php
namespace ns1;

class c1{}

namespace ns1\c1;

class c11{}

namespace main;

use ns1\c1;

$c1 = new c1();
$c11 = new c1\c11();

var_dump($c1); // object(ns1\c1)#1 (0) { }
var_dump($c11); // object(ns1\c1\c11)#2 (0) { }
```
up

down
27
c dot 1 at smithies dot org ¶
12 years ago
If you are testing your code at the CLI, note that namespace aliases do not work!

(Before I go on, all the backslashes in this example are changed to percent signs because I cannot get sensible results to display in the posting preview otherwise. Please mentally translate all percent signs henceforth as backslashes.)

Suppose you have a class you want to test in myclass.php:

```
<?php
namespace my%space;
class myclass {
// ...
}
?>
```

and you then go into the CLI to test it. You would like to think that this would work, as you type it line by line:

```
require 'myclass.php';
use my%space%myclass; // should set 'myclass' as alias for 'my%space%myclass'
$x = new myclass; // FATAL ERROR
```

I believe that this is because aliases are only resolved at compile time, whereas the CLI simply evaluates statements; so use statements are ineffective in the CLI.

If you put your test code into test.php:

```
<?php
require 'myclass.php';
use my%space%myclass;
$x = new myclass;
//...
?>
```

it will work fine.

I hope this reduces the number of prematurely bald people.
up
down
17
cl ¶
10 years ago
Something that is not immediately obvious, particular with PHP 5.3, is that namespace resolutions within an import are not resolved recursively. i.e.: if you alias an import and then use that alias in another import then this latter import will not be fully resolved with the former import.

```
For example:
use \Controllers as C;
use C\First;
use C\Last;
```

Both the First and Last namespaces are NOT resolved as \Controllers\First or \Controllers\Last as one might intend.
up
down
19
x at d dot a dot r dot k dot REMOVEDOTSANDTHIS dot gray dot org ¶
11 years ago
You are allowed to "use" the same resource multiple times as long as it is imported under a different alias at each invocation.

For example:

```
<?php
use Lend;
use Lend\l1;
use Lend\l1 as l3;
use Lend\l2;
use Lend\l1\Keller;
use Lend\l1\Keller as Stellar;
use Lend\l1\Keller as Zellar;
use Lend\l2\Keller as Dellar;

...

?>
```

In the above example, "Keller", "Stellar", and "Zellar" are all references to "\Lend\l1\Keller", as are "Lend\l1\Keller", "l1\Keller", and "l3\Keller".

up
down
1

eithed at google mail ¶

2 years ago

Bear in mind that it's perfectly fine to alias namespaces, ie:

```
<?php
use A\B\C\D\E\User;

new User();
?>
```

can be also written as:

```
<?php
use A\B\C\D\E as ENamespace;

new ENamespace\User();
?>
```

however following will not work:

```
<?php
use A\B\C\D\E as ENamespace;
use ENamespace\User;

new User();
?>
```

```
> PHP Error: Class "ENamespace\User" not found
```

up
down
3

ultimater at gmail dot com ¶

7 years ago

Note that "use" importing/aliasing only applies to the current namespace block.

```
<?php

namespace SuperCoolLibrary
{
class Meta
{
static public function getVersion()
{
```

```
return '2.7.1';
}
}
}

namespace
{
use SuperCoolLibrary\Meta;
echo Meta::getVersion();//outputs 2.7.1
}

namespace
{
echo Meta::getVersion();//fatal error
}

?>

To get the expected behavior, you'd use:
class_alias('SuperCoolLibrary\Meta','Meta');
```

up
down
2

ZhangLiang ¶

6 years ago

```
In Chinese,there is an error in translation:
// 如果不使用 "use \ArrayObject" ，例化一个 foo\ArrayObject 象
it should be
// 如果不使用 "use ArrayObject" ，例化一个 foo\ArrayObject 象

/*****************************************/
中文下翻有
// 如果不使用 "use \ArrayObject" ，例化一个 foo\ArrayObject 象
句是
// 如果不使用 "use ArrayObject" ，例化一个 foo\ArrayObject 象
```

up
down
0

tuxedobob ¶

1 year ago

```
Note that because this is processed at compile time, this doesn't work when running PHP in interactive mode. use commands won't throw an error, but they won't do anything, either.
```

up
down
0

thinice at gmail.com ¶

13 years ago

```
Because imports happen at compile time, there's no polymorphism potential by embedding the use keyword in a conditonal.

e.g.:

<?php
if ($objType == 'canine') {
use Animal\Canine as Beast;
}
if ($objType == 'bovine') {
use Animal\Bovine as Beast;
}

$oBeast = new Beast;
$oBeast->feed();
?>
```

up
down
-4
kelerest123 at gmail dot com ¶
8 years ago
For the fifth example (example #5):

When in block scope, it is not an illegal use of use keyword, because it is used for sharing things with traits.
up
down
-8
Anonymous ¶
10 years ago
The last example on this page shows a possibly incorrect attempt of aliasing, but it is totally correct to import a trait \Languages\Languages\Danish.
up
down
-5
dominic_mayers at yahoo dot com ¶
7 years ago
To clarify the distinction between inserting a trait in a class and importing a trait in a namespace, here is an example where we first import and then insert a trait.

```
<?php
namespace ns1;
trait T {
static $a = "In T";
}

namespace ns2;
use ns1\T; // Importing the name of trait ns1\T in the namespace ns2
class C {
use T; // Inserting trait T in the class C, making use of the imported name.
}

namespace main;
use ns2\C;
echo C::$a; // In T;
```
up
down
-3
info at ensostudio dot ru ¶
3 years ago
Note: you can import not existed items without errors:

```
<?php
use UndefinedClass;
use function undefined_fn;
use const UNDEFINED_CONST;
?>
```
but you cant use/call they:
```
<?php
$new UndefinedClass; // Error: Use of undefined class
use function undefined_fn; // Error: Use of undefined function
use const UNDEFINED_CONST; // Error: Use of undefined constant
?>
```
up
down
-11
Joey ¶
5 years ago
For those hoping for an easy fix of DRY with {} unfortunately you can't nest it or do anything cool with it. In case you can only have one per use and if you don't have one then the whole use is assumed to be wrapped in brackets. That means if

```
you do have one you can't use , as normal with use!

Not possible:

use A\B\C\{
D, E,
F\{X, Y, Z}
},
X\Y\Z,
H\{
I\{
Y\Y\Y\Y\Y,
Z, H, E
},
J\{
A\{
G\H\J
B\N
},
G\H\J
}
};
```

up

down

-13

www.codewars.com ¶

4 years ago

```
amazing!!

use function strval as numberToString;

var_dump(numberToString(123));

//print
//string(3) "123"
```

up

down

-9

Dhairya Lakhera ¶

4 years ago

```
namespace test{

use test\xyz\tikku;
use test\xyz\tikku as class_alias;
use function test\xyz\tikku\foo;
use function test\xyz\tikku\foo as func_alias;
use const test\xyz\tikku\ABC;
use const test\xyz\tikku\ABC as const_alias;




$obj=new tikku;
$obj->Display(); // I am in test\xyz namespace

$obj=new tikku\dhairya;
$obj->Display(); // I am in test\xyz\tikku namespace

$obj=new class_alias\dhairya;
$obj->Display(); // I am in test\xyz\tikku namespace

$obj=new \class_alias\dhairya;
```

```
$obj->Display(); // I am in class_alias namespace

}

namespace test\xyz{

class tikku{
function Display(){
echo "I am in ".__namespace__." namespace<br/><hr/>";
}
}
}

namespace test\xyz\tikku{

class dhairya{
function Display(){
echo "I am in ".__namespace__." namespace<br/><hr/>";
}
}
}

namespace class_alias{

class dhairya{
function Display(){
echo "I am in ".__namespace__." namespace<br/><hr/>";
}
}
}
```