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

return »
« match

- Руководство по PHP
- Справочник языка
- Управляющие конструкции

Change language: Russian

declare

(PHP 4, PHP 5, PHP 7, PHP 8)

Конструкция `declare` используется для установки директив исполнения для блока кода. Синтаксис `declare` аналогичен с синтаксисом других конструкций управления выполнением:

```
declare (directive)
    statement
```

Секция `directive` позволяет установить поведение блока `declare`. В настоящее время распознаются только три директивы: директива `ticks` (Дополнительная информация о директиве ticks доступна ниже), директива `encoding` (Дополнительная информация о директиве encoding доступна ниже) и директива `strict_types` (подробности в разделе про строгую типизацию на странице аргументов функции)

Поскольку директивы обрабатываются при компиляции файла, то только символьные данные могут использоваться как значения директивы. Нельзя использовать переменные и константы. Пример:

```
<?php
// Правильно:
declare(ticks=1);

// Недопустимо:
const TICK_VALUE = 1;
declare(ticks=TICK_VALUE);
?>
```

Часть `statement` блока `declare` будет выполнена — как выполняется и какие побочные эффекты возникают во время выполнения, может зависеть от директивы, которая установлена в блоке `directive`.

Конструкция `declare` также может быть использована в глобальной области видимости, влияя на весь следующий за ней код (однако, если файл с `declare` был включён, то она не будет действовать на родительский файл).

```
<?php
// это то же самое:

// можно так:
declare(ticks=1) {
// прочие действия
}

// или так:
declare(ticks=1);
// прочие действия
?>
```

Тики

Тик - это событие, которое случается каждые *N* низкоуровневых операций, выполненных парсером внутри блока `declare`. Значение *N* задаётся, используя `ticks=`*N* внутри секции `directive` блока `declare`.

Не все выражения подсчитываются. Обычно, условные выражения и выражения аргументов не подсчитываются.

Событие (или несколько событий), которое возникает на каждом тике определяется, используя register_tick_function(). Смотрите пример ниже для дополнительной информации. Имейте в виду, что для одного тика может возникать несколько событий.

Пример #1 Пример использования тика

```
<?php

declare(ticks=1);

// Функция, исполняемая при каждом тике
```

```php
function tick_handler()
{
echo "Вызывается tick_handler()\n";
}

register_tick_function('tick_handler'); // вызывает событие тика

$a = 1; // вызывает событие тика

if ($a > 0) {
$a += 2; // вызывает событие тика
print $a; // вызывает событие тика
}

?>
```

Смотрите также register_tick_function() и unregister_tick_function().

Кодировка

Кодировка скрипта может быть указана для каждого скрипта, используя директиву `encoding`.

Пример #2 Определение кодировки для скрипта.

```php
<?php
declare(encoding='ISO-8859-1');
// прочий код
?>
```

Предостережение

В сочетании с пространством имён единственно допустимый синтаксис для declare является `declare(encoding='...');` где `...` это значение кодировки. `declare(encoding='...') {}` приведёт к ошибке парсера, если используется вместе с пространством имён.

Смотрите также zend.script_encoding.

User Contributed Notes 10 notes

61
Anonymous ¶
13 years ago
It's amazing how many people didn't grasp the concept here. Note the wording in the documentation. It states that the tick handler is called every n native execution cycles. That means native instructions, not including system calls (i'm guessing). This can give you a very good idea if you need to optimize a particular part of your script, since you can measure quite effectively how many native instructions are in your actual code.

A good profiler would take that into account, and force you, the developer, to include calls to the profiler as you're entering and leaving every function. That way you'd be able to keep an eye on how many cycles it took each function to complete. Independent of time.

That is extremely powerful, and not to be underestimated. A good solution would allow aggregate stats, so the total time in a function would be counted, including inside called functions.

22
Kubo2 ¶
8 years ago
Note that in PHP 7 <?php declare(encoding='...'); ?> throws an E_WARNING if Zend Multibyte is turned off.

down
10
digitalaudiorock at gmail dot com ¶
4 years ago
A few important things to note for anyone using this in conjunction with signal handlers:

If anyone is trying to optionally use either pcntl_async_signals() when available (PHP >= 7.1) or ticks for older versions, this is not possible...at least not in a way that does NOT enable ticks for newer PHP versions. This is because there is simply no way to conditionally declare ticks. For example, the following will "work" but not in the way you might expect:

```
<?php
if (function_exists('pcntl_async_signals')) {
pcntl_async_signals(true);
} else {
declare(ticks=1);
}
?>
```

While signal handlers will work with this for old and new version, ticks WILL be enabled even in the case where pcntl_async_signals exists, simply because the declare statement exists. So the above is functionally equivalent to:

```
<?php
if (function_exists('pcntl_async_signals')) pcntl_async_signals(true);
declare(ticks=1);
?>
```

Another thing to be aware of is that the scoping of this declaration changed a bit from PHP 5.6 to 7.x...actually it was corrected apparently as noted here:

http://php.net/manual/en/function.register-tick-function.php#121204

This can cause some very confusing behavior. One example is with the pear/System_Daemon module. With PHP 5.6 that will work with a SIGTERM handler even if the script using it doesn't itself use declare(ticks=1), but does not work in PHP 7 unless the script itself has the declaration. Not only does the handler not get called, but the signal does nothing at all, and the script doesn't exit.

A side note regarding ticks that's annoyed me for some time: As if there wasn't enough confusion around all this, the Internet is full of false rumors that ticks were deprecated and are being removed, and I believe they all started here:

http://www.hackingwithphp.com/4/21/0/the-declare-function-and-ticks

Despite a very obscure author's note at the very end of the page saying he got that wrong (that even I just noticed), the first very prominent sentence of the article still says this, and that page is near the top of any Google search.
up
down
22
sawyerrken at gmail dot com ¶
10 years ago
In the following example:

```
<?php
function handler(){
print "hello <br />";
}

register_tick_function("handler");

declare(ticks = 1){
$b = 2;
} //closing curly bracket tickable
?>
```

"Hello" will be displayed twice because the closing curly bracket is also tickable.

One may wonder why the opening curly bracket is not tickable if the closing is tickable. This is because the instruction for PHP to start ticking is given by the opening curly bracket so the ticking starts immediately after it.

up
down
5

digitalaudiorock at gmail dot com ¶

4 years ago

Regarding my previous comment as to the change in scope of declare(ticks=1) between 5.6 and 7.x, I intended to mention another example of the affect this can have on signal handlers:

If your script uses declare(ticks=1) and assigns handlers, in 5.6 signals will get caught and call the handler even when the code that is running is in an included file (where the included file doesn't have the declaration). However in 7.x the signal wouldn't get caught until the code returns to the main script.

The best solution to that is to use pcntl_async_signals(true) when it's available, which will allow the signals to get caught regardless of what file the code happens to be in.

up
down
6

php dot net at e-z dot name ¶

10 years ago

you can register multiple tick functions:

```
<?PHP
function a() { echo "a\n"; }
function b() { echo "b\n"; }

register_tick_function('a');
register_tick_function('b');
register_tick_function('b');
register_tick_function('b');

?>
```

will output on every tick:

```
a
b
b
b
```

up
down
5

fok at nho dot com dot br ¶

20 years ago

This is a very simple example using ticks to execute a external script to show rx/tx data from the server

```
<?php

function traf(){
passthru( './traf.sh' );
echo "<br />\n";
flush(); // keeps it flowing to the browser...
sleep( 1 );
}

register_tick_function( "traf" );

declare( ticks=1 ){
while( true ){} // to keep it running...
```

```
}

?>

contents of traf.sh:
# Shows TX/RX for eth0 over 1sec
#!/bin/bash

TX1=`cat /proc/net/dev | grep "eth0" | cut -d: -f2 | awk '{print $9}'`
RX1=`cat /proc/net/dev | grep "eth0" | cut -d: -f2 | awk '{print $1}'`
sleep 1
TX2=`cat /proc/net/dev | grep "eth0" | cut -d: -f2 | awk '{print $9}'`
RX2=`cat /proc/net/dev | grep "eth0" | cut -d: -f2 | awk '{print $1}'`

echo -e "TX: $[ $TX2 - $TX1 ] bytes/s \t RX: $[ $RX2 - $RX1 ] bytes/s"
#--= the end. =--
```

up
down
3
ja2016 at wir dot pl ¶
6 years ago

```
Don't use uft-8 encoding with BOM. Then fatal error occurs ALWAYS. Substitute it with utf-8 without BOM.

---

*BOM*
<?php
declare(strict_types=1);
//Fatal error: strict_types declaration must be the very first statement in the script
```

up
down
4
markandrewslade at dontspamemeat dot gmail ¶
15 years ago

```
Note that the two methods for calling declare are not identical.

Method 1:

<?php
// Print "tick" with a timestamp and optional suffix.
function do_tick($str = '') {
list($sec, $usec) = explode(' ', microtime());
printf("[%.4f] Tick.%s\n", $sec + $usec, $str);
}
register_tick_function('do_tick');

// Tick once before declaring so we have a point of reference.
do_tick('--start--');

// Method 1
declare(ticks=1);
while(1) sleep(1);

/* Output:
[1234544435.7160] Tick.--start--
[1234544435.7161] Tick.
[1234544435.7162] Tick.
[1234544436.7163] Tick.
[1234544437.7166] Tick.
*/

?>
```

```
Method 2:
<?php
// Print "tick" with a timestamp and optional suffix.
function do_tick($str = '') {
list($sec, $usec) = explode(' ', microtime());
printf("[%.4f] Tick.%s\n", $sec + $usec, $str);
}
register_tick_function('do_tick');

// Tick once before declaring so we have a point of reference.
do_tick('--start--');

// Method 2
declare(ticks=1) {
while(1) sleep(1);
}

/* Output:
[1234544471.6486] Tick.--start--
[1234544472.6489] Tick.
[1234544473.6490] Tick.
[1234544474.6492] Tick.
[1234544475.6493] Tick.
*/
?>

Notice that when using {} after declare, do_tick wasn't auto-called until about 1 second after we entered the declare {}
block. However when not using the {}, do_tick was auto-called not once but twice immediately after calling declare();.

I'm assuming this is due to how PHP handles ticking internally. That is, declare() without the {} seems to trigger more
low-level instructions which in turn fires tick a few times (if ticks=1) in the act of declaring.
```

up
down
1

ohcc at 163 dot com ¶

4 years ago

```
It's possible to set directives at one time if every directive is supported.
<?php
declare(strict_types=1, encoding='UTF-8');
?>
```

＋add a note