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

Исполнение »
« Сравнение

- Руководство по PHP
- Справочник языка
- Операторы

Change language: Russian

Оператор управления ошибками

PHP поддерживает один оператор управления ошибками: знак `@`. В случае если он предшествует какому-либо выражению в PHP-коде, любые сообщения об ошибках, генерируемые этим выражением, будут подавлены.

Если пользовательская функция обработчика ошибок установлена функцией set_error_handler(), она всё равно будет вызываться, даже если диагностика была подавлена.

Внимание

До версии PHP 8.0.0 функция error_reporting(), вызываемая внутри пользовательского обработчика ошибок, всегда возвращала `0`, если ошибка была подавлена оператором `@`. Начиная с PHP 8.0.0 она возвращает значение этого (побитового) выражения: `E_ERROR | E_CORE_ERROR | E_COMPILE_ERROR | E_USER_ERROR | E_RECOVERABLE_ERROR | E_PARSE`.

Любое сообщение об ошибке, сгенерированное выражением, доступно в элементе массива с ключом `«message»`, возвращаемого функцией error_get_last(). Результат этой функции будет меняться при каждой ошибке, поэтому его необходимо проверить заранее.

```
<?php

/* Преднамеренная ошибка при работе с файлами */
$my_file = @file ('non_existent_file') or
die ("Ошибка при открытии файла: сообщение об ошибке было таким: '" . error_get_last()['message'] . "'");

// работает для любых выражений, а не только для функций
$value = @$cache[$key];
// В случае если ключа $key нет, сообщение об ошибке (notice) не будет отображено
```

> **Замечание**: Оператор `@` работает только с выражениями. Есть простое правило: если что-то возвращает значение, значит, можно указать перед ним оператор `@`. Например, можно указать оператор `@` перед именем переменной, произвольной функцией, вызовом выражения include и т. д. При этом нельзя указывать этот оператор перед определением функции или класса, условными конструкциями, например: `if`, foreach и т. д.

Внимание

До PHP 8.0.0 оператор `@` мог подавлять критические ошибки, которые прерывали выполнение скрипта. Например, добавление оператора `@` к вызову несуществующей функции, когда она недоступна или вызвана с опечаткой, приведёт к прерыванию выполнения скрипта без указания причины.

Смотрите также

- error_reporting()
- Обработка ошибок и функции логирования

User Contributed Notes 16 notes

226
taras dot dot dot di at gmail dot com ¶
15 years ago
I was confused as to what the @ symbol actually does, and after a few experiments have concluded the following:

* the error handler that is set gets called regardless of what level the error reporting is set on, or whether the statement is preceeded with @

* it is up to the error handler to impart some meaning on the different error levels. You could make your custom error handler echo all errors, even if error reporting is set to NONE.

* so what does the @ operator do? It temporarily sets the error reporting level to 0 for that line. If that line triggers

an error, the error handler will still be called, but it will be called with an error level of 0

Hope this helps someone

up
down
141

***M. T.* ¶**

14 years ago

Be aware of using error control operator in statements before include() like this:

```
<?PHP

(@include("file.php"))
OR die("Could not find file.php!");

?>
```

This cause, that error reporting level is set to zero also for the included file. So if there are some errors in the included file, they will be not displayed.

up
down
32

***Anonymous* ¶**

10 years ago

This operator is affectionately known by veteran phpers as the stfu operator.

up
down
60

***anthon at piwik dot org* ¶**

12 years ago

If you're wondering what the performance impact of using the @ operator is, consider this example. Here, the second script (using the @ operator) takes 1.75x as long to execute...almost double the time of the first script.

So while yes, there is some overhead, per iteration, we see that the @ operator added only .005 ms per call. Not reason enough, imho, to avoid using the @ operator.

```
<?php
function x() { }
for ($i = 0; $i < 1000000; $i++) { x(); }
?>

real 0m7.617s
user 0m6.788s
sys 0m0.792s

vs

<?php
function x() { }
for ($i = 0; $i < 1000000; $i++) { @x(); }
?>

real 0m13.333s
user 0m12.437s
sys 0m0.836s
```

up
down
52

***gerrywastaken* ¶**

14 years ago

Error suppression should be avoided if possible as it doesn't just suppress the error that you are trying to stop, but will also suppress errors that you didn't predict would ever occur. This will make debugging a nightmare.

It is far better to test for the condition that you know will cause an error before preceding to run the code. This way only the error that you know about will be suppressed and not all future errors associated with that piece of code.

There may be a good reason for using outright error suppression in favor of the method I have suggested, however in the many years I've spent programming web apps I've yet to come across a situation where it was a good solution. The examples given on this manual page are certainly not situations where the error control operator should be used.

up
down
31

dkellner ¶

7 years ago

There is no reason to NOT use something just because "it can be misused". You could as well say "unlink is evil, you can delete files with it so don't ever use unlink".

It's a valid point that the @ operator hides all errors - so my rule of thumb is: use it only if you're aware of all possible errors your expression can throw AND you consider all of them irrelevant.

A simple example is

```
<?php

$x = @$a["name"];

?>
```

There are only 2 possible problems here: a missing variable or a missing index. If you're sure you're fine with both cases, you're good to go. And again: suppressing errors is not a crime. Not knowing when it's safe to suppress them is definitely worse.

up
down
13

Ryan C ¶

2 years ago

It's still possible to detect when the @ operator is being used in the error handler in PHP8. Calling error_reporting() will no longer return 0 as documented, but using the @ operator does still change the return value when you call error_reporting().

My PHP error settings are set to use E_ALL, and when I call error_reporting() from the error handler of a non-suppressed error, it returns E_ALL as expected.

But when an error occurs on an expression where I tried to suppress the error with the @ operator, it returns: E_ERROR | E_PARSE | E_CORE_ERROR | E_COMPILE_ERROR | E_USER_ERROR | E_RECOVERABLE_ERROR (or the number 4437).

I didn't want to use 4437 in my code in case it changes with different settings or future versions of PHP, so I now use:

```
<?php
function my_error_handler($err_no, $err_msg, $filename, $linenum) {
if (error_reporting() != E_ALL) {
return false; // Silenced
}

// ...
}
?>
```

If the code needs to work with all versions of PHP, you could check that error_reporting() doesn't equal E_ALL or 0.

And, of course, if your error_reporting settings in PHP is something other than E_ALL, you'll have to change that to whatever setting you do use.

up
down
13

man13or at hotmail dot fr ¶

4 years ago

```
Quick debugging methods :

@print($a);
is equivalent to
if isset($a) echo $a ;

@a++;
is equivalent to
if isset($a) $a++ ;
else $a = 1;
```

up
down
7

jcmargentina at gmail dot com ¶

4 years ago

```
Please be aware that the behaviour of this operator changed from php5 to php7.

The following code will raise a Fatal error no matter what, and you wont be able to suppress it

<?php

function query()
{
$myrs = null;
$tmp = @$myrs->free_result();

return $tmp;
}

var_dump(query());

echo "THIS IS NOT PRINT";
?>

more info at: https://bugs.php.net/bug.php?id=78532&thanks=3
```

up
down
19

darren at powerssa dot com ¶

13 years ago

```
After some time investigating as to why I was still getting errors that were supposed to be suppressed with @ I found the following.

1. If you have set your own default error handler then the error still gets sent to the error handler regardless of the @ sign.

2. As mentioned below the @ suppression only changes the error level for that call. This is not to say that in your error handler you can check the given $errno for a value of 0 as the $errno will still refer to the TYPE(not the error level) of error e.g. E_WARNING or E_ERROR etc

3. The @ only changes the rumtime error reporting level just for that one call to 0. This means inside your custom error handler you can check the current runtime error_reporting level using error_reporting() (note that one must NOT pass any parameter to this function if you want to get the current value) and if its zero then you know that it has been suppressed.
<?php
// Custom error handler
function myErrorHandler($errno, $errstr, $errfile, $errline)
{
if ( 0 == error_reporting () ) {
// Error reporting is currently turned off or suppressed with @
return;
```

```
}
// Do your normal custom error reporting here
}
?>
```

For more info on setting a custom error handler see: http://php.net/manual/en/function.set-error-handler.php
For more info on error_reporting see: http://www.php.net/manual/en/function.error-reporting.php

up
down
17

auser at anexample dot com ¶

13 years ago

Be aware that using @ is dog-slow, as PHP incurs overhead to suppressing errors in this way. It's a trade-off between speed and convenience.

up
down
10

bohwaz ¶

12 years ago

If you use the ErrorException exception to have a unified error management, I'll advise you to test against error_reporting in the error handler, not in the exception handler as you might encounter some headaches like blank pages as error_reporting might not be transmitted to exception handler.

So instead of :

```
<?php

function exception_error_handler($errno, $errstr, $errfile, $errline )
{
throw new ErrorException($errstr, 0, $errno, $errfile, $errline);
}

set_error_handler("exception_error_handler");

function catchException($e)
{
if (error_reporting() === 0)
{
return;
}

// Do some stuff
}

set_exception_handler('catchException');

?>
```

It would be better to do :

```
<?php

function exception_error_handler($errno, $errstr, $errfile, $errline )
{
if (error_reporting() === 0)
{
return;
}

throw new ErrorException($errstr, 0, $errno, $errfile, $errline);
}
```

```
set_error_handler("exception_error_handler");

function catchException($e)
{
// Do some stuff
}

set_exception_handler('catchException');

?>
```

up
down
7

frogger at netsurf dot de ¶

19 years ago

Better use the function trigger_error() (http://de.php.net/manual/en/function.trigger-error.php)
to display defined notices, warnings and errors than check the error level your self. this lets you write messages to logfiles if defined in the php.ini, output
messages in dependency to the error_reporting() level and suppress output using the @-sign.

up
down
4

karst dot REMOVETHIS at onlinq dot nl ¶

8 years ago

While you should definitely not be too liberal with the @ operator, I also disagree with people who claim it's the ultimate sin.

For example, a very reasonable use is to suppress the notice-level error generated by parse_ini_file() if you know the .ini file may be missing.
In my case getting the FALSE return value was enough to handle that situation, but I didn't want notice errors being output by my API.

TL;DR: Use it, but only if you know what you're suppressing and why.

up
down
1

ricovox ¶

6 years ago

What is PHP's behavior for a variable that is assigned the return value of an expression protected by the Error Control Operator when the expression encounteres an error?

Based on the following code, the result is NULL (but it would be nice if this were confirmed to be true in all cases).

```
<?php

$var = 3;
$arr = array();

$var = @$arr['x']; // what is the value of $var after this assignment?

// is it its previous value (3) as if the assignment never took place?
// is it FALSE or NULL?
// is it some kind of exception or error message or error number?

var_dump($var); // prints "NULL"

?>
```

up
down
0

fy dot kenny at gmail dot com ¶

3 years ago

```
* How to make deprecated super global variable `$php_errormsg` work

>1. modify php.ini
>track_errors = On
>error_reporting = E_ALL & ~E_NOTICE
>2. Please note, if you already using customized error handler, it will prompt `undefined variable`
>please insert code`set_error_handler(null);` before executing code, e.g:
>```php
>set_error_handler(null);
>$my_file = @file ('phpinfo.phpx') or
>die ("<br>Failed opening file: <br>\t$php_errormsg");
>```

>(c)Kenny Fang
```