Focus search box

Expresiones »
« Sintaxis

- Manual de PHP
- Referencia del lenguaje
- Constantes

Change language: Spanish

Submit a Pull Request Report a Bug

Constantes predefinidas ¶

PHP ofrece un gran número de constantes predefinidas a cualquier script en ejecucción. Muchas de estas constantes, sin embargo, son creadas por diferentes extensiones, y sólo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque han sido compiladas.

Hay nueve constantes mágicas que cambian dependiendo de dónde se emplean. Por ejemplo, el valor de **__LINE__** depende de la línea en que se use en el script. Todas estas constantes «mágicas» se resuelven durante la compilación, a diferencia de las constantes normales que lo hacen durante la ejecución. Estas constantes especiales son sensibles a mayúsculas Estas constantes especiales distinguen mayúsculas y minúsculas, y son las siguientes:

Varias constantes "mágicas" de PHP

Nombre	Descripción
__LINE__	El número de línea actual en el fichero.
__FILE__	Ruta completa y nombre del fichero con enlaces simbólicos resueltos. Si se usa dentro de un include, devolverá el nombre del fichero incluido.
__DIR__	Directorio del fichero. Si se utiliza dentro de un include, devolverá el directorio del fichero incluído. Esta constante es igual que `dirname(__FILE__)`. El nombre del directorio no lleva la barra final a no ser que esté en el directorio root.
__FUNCTION__	Nombre de la función.
__CLASS__	Nombre de la clase. El nombre de la clase incluye el namespace declarado en (p.e.j. `Foo\Bar`). Tenga en cuenta que a partir de PHP 5.4 __CLASS__ también funciona con traits. Cuando es usado en un método trait, __CLASS__ es el nombre de la clase del trait que está siendo utilizado.
__TRAIT__	El nombre del trait. El nombre del trait incluye el espacio de nombres en el que fue declarado (p.e.j. `Foo\Bar`).
__METHOD__	Nombre del método de la clase.
__NAMESPACE__	Nombre del espacio de nombres actual.
ClassName::class	El nombre de clase completamente cualificado. Véase también ::class.

Véase también get_class(), get_object_vars(), file_exists() y function_exists().

Historial de cambios ¶

Versión	Descripción
5.5.0	Se añadió la constante mágica **::class**
5.4.0	Se añadió la constante **__TRAIT__**
5.3.0	Se añadieron las constantes **__DIR__** y **__NAMESPACE__**
5.0.0	Se añadió la constante **__METHOD__**

Versión	Descripción
5.0.0	Antes de esta versión, los valores de algunas constantes mágicas estaban siempre en minúsculas. Ahora todas ellas están en mayúsculas (contienen nombres mientras eran declaradas).
4.3.0	Se añadieron las constantes **__FUNCTION__** y **__CLASS__**
4.0.2	**__FILE__** siempre contiene una ruta absoluta con enlaces simbólicos resueltos, mientras que en versiones antiguas contenía una ruta relativa bajo algunas circunstancias

User Contributed Notes 8 notes

291
vijaykoul_007 at rediffmail dot com ¶
17 years ago

```
the difference between
__FUNCTION__ and __METHOD__ as in PHP 5.0.4 is that

__FUNCTION__ returns only the name of the function

while as __METHOD__ returns the name of the class alongwith the name of the function

class trick
{
      function doit()
      {
                echo __FUNCTION__;
      }
      function doitagain()
      {
                echo __METHOD__;
      }
}
$obj=new trick();
$obj->doit();
output will be ----  doit
$obj->doitagain();
output will be ----- trick::doitagain
```


47
Tomek Perlak [tomekperlak at tlen pl] ¶
16 years ago

```
The __CLASS__ magic constant nicely complements the get_class() function.

Sometimes you need to know both:
- name of the inherited class
- name of the class actually executed

Here's an example that shows the possible solution:

<?php

class base_class
{
```

```
    function say_a()
    {
        echo "'a' - said the " . __CLASS__ . "<br/>";
    }

    function say_b()
    {
        echo "'b' - said the " . get_class($this) . "<br/>";
    }

}

class derived_class extends base_class
{
    function say_a()
    {
        parent::say_a();
        echo "'a' - said the " . __CLASS__ . "<br/>";
    }

    function say_b()
    {
        parent::say_b();
        echo "'b' - said the " . get_class($this) . "<br/>";
    }
}

$obj_b = new derived_class();

$obj_b->say_a();
echo "<br/>";
$obj_b->say_b();

?>

The output should look roughly like this:

'a' - said the base_class
'a' - said the derived_class

'b' - said the derived_class
'b' - said the derived_class
```

up
down
5
public at taliesinnuin dot net ¶
2 years ago

If you're using PHP with fpm (common in this day and age), be aware that __DIR__ and __FILE__ will return values based on the fpm root which MAY differ from its actual location on the file system.

This can cause temporary head-scratching if deploying an app where php files within the web root pull in PHP files from outside of itself (a very common case). You may be wondering why __DIR__ returns "/" when the file itself lives in /var/www/html or whathaveyou.

You might handle such a situation by having NGINX explicitly add the necessary part of the path in its fastcgi request and then you can set the root on the FPM process / server / container to be something other than the webroot (so long as no other way it could become publicly accessible).

Hope that saves someone five minutes who's moving code to FPM that uses __DIR__.

up
down
18

david at thegallagher dot net ¶
10 years ago

You cannot check if a magic constant is defined. This means there is no point in checking if __DIR__ is defined then defining it. `defined('__DIR__')` always returns false. Defining __DIR__ will silently fail in PHP 5.3+. This could cause compatibility issues if your script includes other scripts.

Here is proof:

```
<?php
echo (defined('__DIR__') ? '__DIR__ is defined' : '__DIR__ is NOT defined' . PHP_EOL);
echo (defined('__FILE__') ? '__FILE__ is defined' : '__FILE__ is NOT defined' . PHP_EOL);
echo (defined('PHP_VERSION') ? 'PHP_VERSION is defined' : 'PHP_VERSION is NOT defined') . PHP_EOL;
echo 'PHP Version: ' . PHP_VERSION . PHP_EOL;
?>
```

Output:

```
__DIR__ is NOT defined
__FILE__ is NOT defined
PHP_VERSION is defined
PHP Version: 5.3.6
```

up
down
13

php at kenman dot net ¶
8 years ago

Just learned an interesting tidbit regarding __FILE__ and the newer __DIR__ with respect to code run from a network share: the constants will return the *share* path when executed from the context of the share.

Examples:

```
// normal context
// called as "php -f c:\test.php"
__DIR__ === 'c:\';
__FILE__ === 'c:\test.php';

// network share context
// called as "php -f \\computerName\c$\test.php"
__DIR__ === '\\computerName\c$';
__FILE__ === '\\computerName\c$\test.php';
```

NOTE: realpath('.') always seems to return an actual filesystem path regardless of the execution context.

up
down
11

Sbastien Fauvel ¶
6 years ago

Note a small inconsistency when using __CLASS__ and __METHOD__ in traits (stand php 7.0.4): While __CLASS__ is working as advertized and returns dynamically the name of the class the trait is being used in, __METHOD__ will actually prepend the trait name instead of the class name!

up

down
8
meindertjan at gmail dot spamspamspam dot com ¶
8 years ago
A lot of notes here concern defining the __DIR__ magic constant for PHP versions not supporting the feature. Of course you can define this magic constant for PHP versions not yet having this constant, but it will defeat its purpose as soon as you are using the constant in an included file, which may be in a different directory then the file defining the __DIR__ constant. As such, the constant has lost its *magic*, and would be rather useless unless you assure yourself to have all of your includes in the same directory.

Concluding: eye catchup at gmail dot com's note regarding whether you can or cannot define magic constants is valid, but stating that defining __DIR__ is not useless, is not!
up
down
11
chris dot kistner at gmail dot com ¶
11 years ago
There is no way to implement a backwards compatible __DIR__ in versions prior to 5.3.0.

The only thing that you can do is to perform a recursive search and replace to dirname(__FILE__):
find . -type f -print0 | xargs -0 sed -i 's/__DIR__/dirname(__FILE__)/'

+ add a note

- Constantes
  - Sintaxis
  - Constantes predefinidas

- Copyright © 2001-2022 The PHP Group
- My PHP.net
- Contact
- Other PHP.net sites
- Privacy policy
- View Source