Search

Dutch PHP Conference 2024

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

Управляющие конструкции »
« Массивы

- Руководство по PHP
- Справочник языка
- Операторы

Change language: Russian

Оператор проверки типа

Оператор `instanceof` определяет, относится ли сохранённый в PHP-переменной объект к конкретному классу.

Пример #1 Пример использования оператора `instanceof` с классами

```
<?php

class MyClass {}

class NotMyClass {}

$a = new MyClass();

var_dump($a instanceof MyClass);
var_dump($a instanceof NotMyClass);
```

Результат выполнения приведённого примера:

```
bool(true)
bool(false)
```

Оператор `instanceof` также определяет, принадлежит ли сохранённый в переменной объект к классу-наследнику:

Пример #2 Использование оператора `instanceof` с наследуемыми классами

```
<?php

class ParentClass {}

class MyClass extends ParentClass {}

$a = new MyClass();

var_dump($a instanceof MyClass);
var_dump($a instanceof ParentClass);
```

Результат выполнения приведённого примера:

```
bool(true)
bool(true)
```

Для проверки *непринадлежности* объекта классу, указывают логический оператор `not`.

Пример #3 Использование оператора `instanceof` для проверки того, что объект — это *не* экземпляр класса

```
<?php

class MyClass {}

$a = new MyClass();
var_dump(!($a instanceof stdClass));
```

Результат выполнения приведённого примера:

```
bool(true)
```

Наконец, оператор `instanceof` также проверяет, реализует ли объект интерфейс:

Пример #4 Использование оператора `instanceof` с интерфейсами

```
<?php

interface MyInterface {}

class MyClass implements MyInterface {}
```

```php
$a = new MyClass();

var_dump($a instanceof MyClass);
var_dump($a instanceof MyInterface);
```

Результат выполнения приведённого примера:

```
bool(true)
bool(true)
```

Хотя оператор `instanceof` обычно указывают с буквальным именем класса, его можно также указывать с переменной объекта или строковой переменной:

Пример #5 Использование оператора `instanceof` с другими переменными

```php
<?php

interface MyInterface {}

class MyClass implements MyInterface {}

$a = new MyClass();
$b = new MyClass();
$c = 'MyClass';
$d = 'NotMyClass';
var_dump($a instanceof $b); // $b — объект класса MyClass
var_dump($a instanceof $c); // $c — строка 'MyClass'
var_dump($a instanceof $d); // $d — строка 'NotMyClass'
```

Результат выполнения приведённого примера:

```
bool(true)
bool(true)
bool(false)
```

Оператор instanceof не выбрасывает никаких ошибок, если проверяемая переменная — не объект, он просто возвращает **`false`**. Константы, однако, не были разрешены до PHP 7.3.0.

Пример #6 Пример использования оператора `instanceof` для проверки других переменных

```php
<?php

$a = 1;
$b = NULL;
$c = imagecreate(5, 5);
var_dump($a instanceof stdClass); // $a — целое типа integer
var_dump($b instanceof stdClass); // $b — NULL
var_dump($c instanceof stdClass); // $c — значение типа resource
var_dump(FALSE instanceof stdClass);
```

Результат выполнения приведённого примера:

```
bool(false)
bool(false)
bool(false)
PHP Fatal error:  instanceof expects an object instance, constant given
```

Начиная с PHP 7.3.0 константы разрешены в левой части оператора `instanceof`.

Пример #7 Использование `instanceof` для проверки констант

```php
<?php

var_dump(FALSE instanceof stdClass);
```

Результат выполнения приведённого примера в PHP 7.3:

```
bool(false)
```

Начиная с PHP 8.0.0 `instanceof` можно использовать с произвольными выражениями. Выражение должно быть заключено в круглые скобки и быть строкой (string).

Пример #8 Пример использования `instanceof` с произвольным выражением

```
<?php

class ClassA extends \stdClass {}
class ClassB extends \stdClass {}
class ClassC extends ClassB {}
class ClassD extends ClassA {}

function getSomeClass(): string
{
return ClassA::class;
}

var_dump(new ClassA instanceof ('std' . 'Class'));
var_dump(new ClassB instanceof ('Class' . 'B'));
var_dump(new ClassC instanceof ('Class' . 'A'));
var_dump(new ClassD instanceof (getSomeClass()));
```

Результат выполнения приведённого примера в PHP 8:

```
bool(true)
bool(true)
bool(false)
bool(true)
```

Оператор `instanceof` аналогичен функции is_a().

Смотрите также

- get_class()
- is_a()

＋add a note

User Contributed Notes 16 notes

up
down
121
jphaas at gmail dot com ¶
16 years ago
Posting this so the word typeof appears on this page, so that this page will show up when you google 'php typeof'.

...yeah, former Java user.

up
down
61
knarlin at yahoo dot com dot au ¶
10 years ago
Checking an object is not an instance of a class, example #3 uses extraneous parentheses.

```
<?php
var_dump(!($a instanceof stdClass));
?>
```

Because instanceof has higher operator precedence than ! you can just do

```
<?php
var_dump( ! $a instanceof stdClass );
?>
```

up
down

47

Sudarshan Wadkar ¶

12 years ago

I don't see any mention of "namespaces" on this page so I thought I would chime in. The instanceof operator takes FQCN as second operator when you pass it as string and not a simple class name. It will not resolve it even if you have a `use MyNamespace\Bar;` at the top level. Here is what I am trying to say:

```
## testinclude.php ##
<?php
namespace Bar1;
{
class Foo1{ }
}
namespace Bar2;
{
class Foo2{ }
}
?>
## test.php ##
<?php
include('testinclude.php');
use Bar1\Foo1 as Foo;
$foo1 = new Foo(); $className = 'Bar1\Foo1';
var_dump($foo1 instanceof Bar1\Foo1);
var_dump($foo1 instanceof $className);
$className = 'Foo';
var_dump($foo1 instanceof $className);
use Bar2\Foo2;
$foo2 = new Foo2(); $className = 'Bar2\Foo2';
var_dump($foo2 instanceof Bar2\Foo2);
var_dump($foo2 instanceof $className);
$className = 'Foo2';
var_dump($foo2 instanceof $className);
?>
## stdout ##
bool(true)
bool(true)
bool(false)
bool(true)
bool(true)
bool(false)
```

up

down

42

dava ¶

10 years ago

You are also able to compare 2 objects using instanceOf. In that case, instanceOf will compare the types of both objects. That is sometimes very useful:

```
<?php

class A { }
class B { }

$a = new A;
$b = new B;
$a2 = new A;

echo $a instanceOf $a; // true
echo $a instanceOf $b; // false
echo $a instanceOf $a2; // true
```

```
?>
```
up
down
2
wapinet at mail dot ru ¶
2 years ago
if you have only class names (not objects) you can use that snippet: https://3v4l.org/mUKUC
```
<?php
interface i{}
class a implements i{}

var_dump(a::class instanceof i); // false
var_dump(in_array(i::class, class_implements(a::class), true)); // true
```
up
down
3
ASchmidt at Anamera dot net ¶
4 years ago
Using an undefined variable will result in an error.

If variable is in doubt, one must prequalify:

```
if ( isset( $MyInstance ) and $MyInstance instanceof MyClass ) ...
```
up
down
4
wadih at creationmw dot com ¶
6 years ago
Doing $a instanceof stdClass from inside a namespace will not work on its own.

You will have to do:

```
<?php
if ($a instanceof \stdClass)
?>
```
up
down
17
jtaal at eljakim dot nl ¶
15 years ago
You can use "self" to reference to the current class:

```
<?php
class myclass {
function mymethod($otherObject) {
if ($otherObject instanceof self) {
$otherObject->mymethod(null);
}
return 'works!';
}
}

$a = new myclass();
print $a->mymethod($a);
?>
```
up
down
13
fbableus ¶
12 years ago
If you want to test if a classname is an instance of a class, the instanceof operator won't work.

```
<?php
$classname = 'MyClass';
if( $classname instanceof MyParentClass) echo 'Child of it';
else echo 'Not child of it';
?>

Will always output
Not child of it

You must use a ReflectionClass :
<?php
$classname = 'MyClass';
$myReflection = new ReflectionClass($classname);
if( $myReflection->isSubclassOf('MyParentClass')) echo 'Child of it';
else echo 'Not child of it';
?>

Will output the good result.
If you're testing an interface, use implementsInterface() instead of isSublassOf().
```

up
down
6

kevin dot benton at beatport dot com ¶

15 years ago

```
Example #5 could also be extended to include...

var_dump($a instanceof MyInterface);

The new result would be

bool(true)

So - instanceof is smart enough to know that a class that implements an interface is an instance of the interface, not
just the class. I didn't see that point made clearly enough in the explanation at the top.
```

up
down
5

wbcarts at juno dot com ¶

11 years ago

```
SIMPLE, CLEAN, CLEAR use of the instanceof OPERATOR

First, define a couple of simple PHP Objects to work on -- I'll introduce Circle and Point. Here's the class definitions
for both:

<?php

class Circle
{
protected $radius = 1.0;

/*
* This function is the reason we are going to use the
* instanceof operator below.
*/
public function setRadius($r)
{
$this->radius = $r;
}

public function __toString()
{
return 'Circle [radius=' . $this->radius . ']';
```

```
}
}

class Point
{
protected $x = 0;
protected $y = 0;

/*
* This function is the reason we are going to use the
* instanceof operator below.
*/
public function setLocation($x, $y)
{
$this->x = $x;
$this->y = $y;
}

public function __toString()
{
return 'Point [x=' . $this->x . ', y=' . $this->y . ']';
}
}

?>
```

Now instantiate a few instances of these types. Note, I will put them in an array (collection) so we can iterate through them quickly.

```
<?php

$myCollection = array(123, 'abc', 'Hello World!',
new Circle(), new Circle(), new Circle(),
new Point(), new Point(), new Point());

$i = 0;
foreach($myCollection AS $item)
{
/*
* The setRadius() function is written in the Circle class
* definition above, so make sure $item is an instance of
* type Circle BEFORE calling it AND to avoid PHP PMS!
*/
if($item instanceof Circle)
{
$item->setRadius($i);
}

/*
* The setLocation() function is written in the Point class
* definition above, so make sure $item is an instance of
* type Point BEFORE calling it AND to stay out of the ER!
*/
if($item instanceof Point)
{
$item->setLocation($i, $i);
}

echo '$myCollection[' . $i++ . '] = ' . $item . '<br>';
}

?>
```

```
$myCollection[0] = 123
$myCollection[1] = abc
$myCollection[2] = Hello World!
$myCollection[3] = Circle [radius=3]
$myCollection[4] = Circle [radius=4]
$myCollection[5] = Circle [radius=5]
$myCollection[6] = Point [x=6, y=6]
$myCollection[7] = Point [x=7, y=7]
$myCollection[8] = Point [x=8, y=8]
```

up
down
4

Hayley Watson ¶

6 years ago

If you want to use "$foo instanceof $bar" to determine if two objects are the same class, remember that "instanceof" will also evaluate to true if $foo is an instance of a _subclass_ of $bar's class.

If you really want to see if they are the _same_ class, then they both have to be instances of each other's class. That is:

```
<?php

($foo instanceof $bar && $bar instanceof $foo)

?>
```

Consider it an alternative to "get_class($bar) == get_class($foo)" that avoids the detour through to string lookups and comparisons.

up
down
5

julien plee using g mail dot com ¶

16 years ago

Response to vinyanov at poczta dot onet dot pl:

You mentionned "the instanceof operator will not accept a string as its first operand". However, this behavior is absolutely right and therefore, you're misleading the meaning of an instance.

<?php 'ClassA' instanceof 'ClassB'; ?> means "the class named ClassA is an instance of the class named ClassB". This is a nonsense sentence because when you instanciate a class, you ALWAYS obtain an object. Consequently, you only can ask if an object is an instance of a class.

I believe asking if "a ClassA belongs to a ClassB" (or "a ClassA is a class of (type) ClassB") or even "a ClassA is (also) a ClassB" is more appropriate. But the first is not implemented and the second only works with objects, just like the instanceof operator.

Plus, I just have tested your code and it does absolutely NOT do the same as instanceof (extended to classes)! I can't advise anyone to reuse it. The use of <?php is_instance_of ($instanceOfA, 'ClassB'); ?> raises a warning "include_once(Object id #1.php) …" when using __autoload (trying to look for $instanceOfA as if it was a class name).

Finally, here is a fast (to me) sample function code to verify if an object or class:

```
<?php
function kind_of (&$object_or_class, $class)
{
return is_object ($object_or_class) ?
$object_or_class instanceof $class
: (is_subclass_of ($object_or_class $class)
|| strtolower ($object_or_class) == strtolower ($class));
}
?>
```

up

down
3
ejohnson82 at gmail dot com ¶
16 years ago
The PHP parser generates a parse error on either of the two lines that are commented out here.
Apparently the 'instanceof' construct will take a string variable in the second spot, but it will NOT take a string...
lame

```
class Bar {}
$b = new Bar;
$b_class = "Bar";
var_export($b instanceof Bar); // this is ok
var_export($b instanceof $b_class); // this is ok
//var_export($f instanceof "Bar"); // this is syntactically illegal
//var_export($f instanceof 'Bar'); // this is syntactically illegal
```

up
down
1
jeanyves dot terrien at orange-ftgroup dot com ¶
16 years ago
Cross version function even if you are working in php4
(instanceof is an undefined operator for php4)

```
function isMemberOf($classename) {
$ver = floor(phpversion());
if($ver > 4) {
$instanceof = create_function ('$obj,$classname','return $obj instanceof $classname;');
return $instanceof($this,$classname);
} else {
// Php4 uses lowercase for classname.
return is_a($this, strtolower($classname));
}
} // end function isMemberOf
```

up
down
2
soletan at toxa dot de ¶
16 years ago
Please note: != is a separate operator with separate semantics. Thinking about language grammar it's kind of ridicilous to negate an operator. Of course, it's possible to negate the result of a function (like is_a()), since it isn't negating the function itself or its semantics.

instanceof is a binary operator, and so used in binary terms like this

```
terma instanceof termb
```

while ! (negation) is a unary operator and so may be applied to a single term like this

```
!term
```

And a term never consists of an operator, only! There is no such construct in any language (please correct me!). However, instanceof doesn't finally support nested terms in every operand position ("terma" or "termb" above) as negation does:

```
!!!!!!!!!!!!!!term == term
```

So back again, did you ever write

```
a !!!!!!!!!!!= b
```

to test equivalence?
＋add a note

- Copyright © 2001-2024 The PHP Group
- My PHP.net
- Contact
- Other PHP.net sites
- Privacy policy