Introduction to PHP static methods and static properties

Sometimes, it is useful if we can access methods and properties in the context of a [class](http://www.zentut.com/php-tutorial/php-objects-and-classes/)rather than an object. To do this, you can use static keyword.

To add a static method to a class, you use the *static* keyword as follows:

|  |  |
| --- | --- |
| 1  2  3 | public static function static\_method(){      // method implementation  } |

You can put the static keyword before or after the method’s visibility. However, by convention, the visibility is declared first.

To add a static property to a class, you also use the static keyword as the following syntax:

|  |  |
| --- | --- |
| 1 | private static $static\_property; |

The static methods and static properties are not linked to any particular object of the class but the class itself.

To call a static method outside the class, you use the :: operator as follows:

|  |  |
| --- | --- |
| 1 | MyClass::static\_method(); |

To access a public static property outside the class, you also use the :: operator:

|  |  |
| --- | --- |
| 1 | MyClass::$static\_property; |

However to access static methods and static properties from within an instance of the class, you use  selfinstead of  $thisas follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php  class MyClass{  private static $static\_property;    public static function static\_method(){  //...  }    public function method(){  self::$static\_property;  self::static\_method();  }  } |

Developers, who a new to PHP object-oriented programming, are sometimes confused between  self and $this. Let’s make the concepts clear.

*self* vs. *$this*

| **$this** | **self** |
| --- | --- |
| Represents an instance of the class or object | Represents a class |
| Always begin with dollar ($) sign | Never begin with dollar($) sign |
| Is followed by the -> operator | Is followed by the :: operator |
| The property name after the -> operator does not take dollar ($) sign, e.g., $this->property. | The property name after the :: operator always take the dollar ($) sign. |

PHP static methods and properties example

The following example illustrates how to use static method and property:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <?php  class Counter{  private $count;  public static $instance;    public function \_\_construct($count = 0){  $this->count = $count;  self::$instance++;  }    public function count(){  $this->count++;  return $this;    }    public function \_\_toString(){  return (string)$this->count;  }    public static function countInstance(){  return self::$instance;  }  }    $c1 = new Counter();  $c1->count()     ->count();  echo 'Counter 1 value: '. $c1 . '<br/>';    $c2 = new Counter();  $c2->count()     ->count()     ->count();    echo 'Counter 2 value: '. $c2 . '<br/>';    echo 'The number of counter instances:'  . Counter::countInstance()      . '<br/>'; |

How it works.

* We declared a new class named Counter that has two properties: $count*and*$instance, where  $instance is a static property*.*Each time a new counter object is created, we increased the value of the  $instance static property by 1. Notice that we accessed the  $instanceproperty by using the self and the :: operator.
* Besides the regular methods, we declared a static method named  countInstance()that is used to return the number of objects instantiated from the Counter class.
* In the test script section below the class, we created two counter objects and displayed the number of instances by calling the countInstance() static method.

When to define static methods

Before defining a static method, ask yourself a question: “Does it make sense to call the method without instantiating a new object?” . If a logic that can be shared by multiple instances of the class, you can extract the code and put it into a static method.

PHP static methods are often used in *utility* classes of  PHP frameworks. A utility class is a class that contains only static methods.

In this tutorial, we have introduced you PHP static methods and static properties of a class. We then shown you the differences between *$this* and *self ,*and discussed when we can use static methods.