PHP Static Methods and Properties

Summary: in this tutorial, you will learn about PHP static methods and static properties and understand the differences between the `$this` and `self`.

Introduction to PHP static methods and properties

So far, you have learned how to define a class (https://www.phptutorial.net/php-oop/php-objects/) that consists of methods and properties. To use the methods and properties of the class, you create an object and access these methods and properties via the object.

Since these methods and properties are bound to an instance of the class, they are called instance methods and properties.

PHP allows you to access the methods and properties in the context of a class rather than an object. Such methods and properties are class methods and properties.

Class methods and class properties are called static methods and properties.

Static methods

To define a static method, you place the `static` keyword in front of the `function` keyword as follows:

```
<?php

class MyClass
{
        public static function staticMethod()
        {
```

```
        }
}
```

Since a static method is bound to a class, not an individual instance of the class, you cannot access `$this` inside the method. However, you can access a special variable called `self`. The `self` variable means the current class.

The following shows how to call a static method from the inside of the class:

```
self::staticMethod(arguments);
```

To call a static method from the outside of the class, you use the following syntax:

```
className::staticMethod(arguments)
```

For example:

```
MyClass::staticMethod()
```

The following example defines the `HttpRequest` class that has a static method `uri()` that returns the URI of the current HTTP request:

```
class HttpRequest
{
        public static function uri(): string
        {
                return strtolower($_SERVER['REQUEST_URI']);
        }
}
```

The following calls the `uri()` static method of `HttpRequest` class:

```
echo HttpRequest::uri();
```

If the current HTTP request is `https://www.phptutorial.net/php-static-method/`, the `uri()` method will return `/php-static-method/`.

Static properties

To define a static property, you also use the `static` keyword:

```
public static $staticProperty;
```

For example:

```
class MyClass
{
        public static $staticProperty;

        public static function staticMethod()
        {
        }
}
```

To access a public static property outside of the class, you also use the class name with the `::` operator:

```
MyClass::$staticProperty;
```

Like the static methods, to access static properties from within the class, you use the `self` instead of `$this` as follows:

```
self::staticProperty
```

self vs. $this

The following table illustrates the differences between the `self` and `$this`:

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

PHP static methods and properties example

Suppose that you want to create an `App` class for your web application. And the `App` class should have one and only one instance during the lifecycle of the application. In other words, the `App` should be a singleton.

The following illustrates how to define the App class by using the static methods and properties:

```
<?php

class App
{
        private static $app = null;

        private function __construct()
        {
        }

        public static function get() : App
        {
                if (!self::$app) {
                        self::$app = new App();
                }

                return self::$app;
```

```
        }

        public function bootstrap(): void
        {
                echo 'App is bootstrapping...';
        }
}
```

How it works:

First, define a static property called $app and initialize its value to `null`:

```
private static $app = null;
```

Second, make the constructor (https://www.phptutorial.net/php-oop/php-constructors/) private so that the class cannot be instantiated from the outside:

```
private function __construct()
{
}
```

Third, define a static method called `get()` that returns an instance of the `App` class:

```
public static function get() : App
{
        if (!self::$app) {
                self::$app = new App();
        }

        return self::$app;
}
```

The `get()` method creates an instance of the `App` if it has not been created, otherwise, it just simply returns the App's instance. Notice that the `get()` method uses the `self` to access the `$app` static property.

Fourth, the `bootstrap()` method is just for demonstration purposes. In practice, you can place the code that bootstraps the application in this method.

The following shows how to use the `App` class:

```
<?php

$app = App::get();
$app->bootstrap();
```

In this code, we called the `get()` static method from the `App` class and invoked the `bootstrap()` method of the `App` 's instance.

Summary

- Static methods and properties are bound to a class, not individual objects of the class.
- Use the `static` keyword to define static methods and properties.
- Use the `self` keyword to access static methods and properties within the class.