PHP Interfaces

In PHP, interface is block which declares set of functions to be defined with a class to implement this interface. A class can extend more than one interface, there by, we can simulate  multiple inheritance in PHP. Let us see all about it, in detail with this article.

## PHP class implements Interface

While implementing an interface with a PHP class, we must define all the functions declared with that interface as it is, without changing arguments, return type, access modifiers or any thing.  
For example, the following code has deals with PHP interface and class to show how to implement an interface for a class.

interface UserInterface {

public function setMenu($menu\_array);

public function getContent($page);

}

class WebApp implements UserInterface {

public function setMenu($menu\_array) {

$menu\_bar="";

foreach($menu\_array as $k=>$v) {

$menu\_bar .="<a href='" . $menu\_array[$k]["menu\_link"] . "'>" . $menu\_array[$k]["menu\_caption"] . "</a>  ";

}

return $menu\_bar;

}

public function getContent($page) {

$content = "<h1>Welcome to the " . ucwords($page) . " page</h1>";

return $content;

}

}

In the above example, we have created an PHP interface named as UserInterface, and this is implemented by a class WebApp using implements keyword. We can implement this interface, wherever we want the same set of user interface components that is menu and content for our web application.

While implementation of the above interface, two functions declared at UserInterfacehas been defined in the class which implements it.  
  
While defining, if any one of those functions, or any signature, like arguments, return type are changed, then it will cause some [PHP error](http://phppot.com/php/php-errors/). For example, if the WebApp class is not having getContent() definition, then, the error will be,  
  
Fatal error: Class WebApp contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (UserInterface::getContent) in ...

## Interface extends Interface

In PHP, an interface can extends another interface and there by it inherits all functions declared with this interface. For that, we should use extendskeyword, as we used for inheriting classes.

For example, let us add one more interface named Widget, with the above program, in which getPopularTags() function is declared. Now, we are going to extend this Widgetinterface for UserInterface.

interface Widget {

public function getPopularTags();

}

interface UserInterface extends Widget {

public function setMenu($menu\_array);

public function getContent($page);

}

By extending Widget interface, meaning that, the UserInterface is extended with one more function declaration for getPopularTags(). Now, for implementing UserInterface, a PHP class must define getPopularTags() also.

### Multiple Inheritance in PHP

As we have seen in [PHP inheritance](http://phppot.com/php/php-inheritance/) article, PHP supports only single inheritance. But, using this property of extending PHP interfaces, we can simulate multiple inheritance.

An interface extends two interfaces, as like as a class inherits two parent classes where multiple inheritance is supported. For example, in the following program, UserInterfaceextends both interface named Advertisement, Widget to inherit the functions declared with them.

interface Widget {

public function getPopularTags();

}

interface Advertisement {

public function getAdsTile();

}

interface UserInterface extends Widget, Advertisement{

public function setMenu($menu\_array);

public function getContent($page);

}

If any PHP class which implements Usernterface must have definition for all the functions declared in Usernterface and also in other interfaces, that is, Widget,Advertisement,  it extends with.

## PHP Interface Constants

In PHP, the constants are defined inside interface block, by using constkeyword. For example, a PHP Interface constant can be defined as follows.

const MAX\_LENGTH = "100";

We can access the interface constants by using interface name, or by using the name of the class, and also the name of the class instance, which implements the interface.

For example, if we define MAX\_LENGTHfor Usernterfaceof our first example, then this constant can be accessed by using the following possible number of ways.

* *Usernterface ::MAX\_LENGTH*
* *WebApp ::MAX\_LENGTH*
* Instance of *WebApp (*if any).
* *$this::MAX\_LENGTH (*within class definition).

**Note:**

* All functions declared in PHP interfaces should be specified with public modifier to allow its visibility on defining them for classes which implements this interface
* We cannot extend class with an interface. For example,

class-name extends interface-name

will cause error, like,

Fatal error: Class ... cannot extend from interface UserInterface in...

* Similarly, we can not implement an interface into another interface,  which will cause *T\_IMPLEMENTS* error.

Interface:-

1).we can’t create object of interface

**Class :-**

->Class is the collection of beehive and attributes

->Class is the collection of Member function(functions) and Data members(variables)

->Feature of **class** is it is reusable purpose

**Object :-**

->Object is a copy of class

->Object is a blue print of a class or instance of a class

->With the help of Object we can access member function and data member of a class

**Static :-**

-> We can’t call a **normal variable** inside a **static function**

**-> S**tatic variable will preserve its previous value

-> Normal variable will get initialize every time whenever you creation an object