Programación

PHP

Vigésimo tercera semana

Abril 2022

Cruz García, Iago

Introducción

Esta semana vamos a ver como utilizar clases y objetos en PHP, similar a JavaScript y otros lenguajes, y a "conectar" PHP y JavaScript.

Recomendación: A partir de ahora es necesario que a la hora de realizar ejercicios nos acostumbremos a buscar información en la documentación oficial. En caso de no poder resolver las dudas con la documentación ofrecida, el siguiente paso será preguntar las dudas en el foro de clase.

Documentación oficial de JavaScript

Foro de la asignatura

XAMPP para utilizar PHP

Página oficial de PHP

Clases y objetos

Creación e instanciación

Como ya vimos en anteriores unidades, las clases nos permiten crear "variables" que pueden almacenar atributos y funciones mediante la instanciación en objetos. En PHP encontraremos las mismas funciones, con una sintaxis similar:

index.php

```
<html>
<body>
<?php
  class Vehiculo{
    public $ruedas;

    function __construct($ruedas){
          $this->ruedas=$ruedas;
    }

    function set_ruedas($ruedas){
          $this->ruedas=$ruedas;
    }

    function get_ruedas(){
          return $this->ruedas;
    }
}
$coche = new Vehiculo(2);
echo $coche->get_ruedas();
```

```
?>
</body>
</html>
```

Como vemos, encontramos las mismas estructuras:

- **public $*variable*:** Permite crear un atributo. Aquí utilizamos public, que indica que la variable es accesible públicamente, es decir, no haría falta llamar al método "get", pero podríamos utilizar **private**, que bloquearía el acceso inmediato, necesitando una función intermedia, como el "get".
- **__construct(*parametros)*:** Es el constructor de la clase, lo que nos permite instanciar el objeto con unos valores iniciales.
- **$this->variable**: Permite acceder a la variable de la clase.
- **function *función()*:** Permite el mismo funcionamiento que en JavaScript, crear funciones con parámetros de entrada o devolver variables según necesidad.
- **new *nombreClase()*:** Identico a otros lenguajes orientados a objetos, permite instanciar un objeto de la clase especificada y acceder así a sus valores.

Herencia

Otro de los puntos fuertes del uso de clases es la capacidad de heredar comportamientos de "padres" a "hijos". Para ello, usaremos la siguiente sintaxis:

index.php

```
<html>
<body>
<?php
  class Vehiculo{
    public $ruedas;

    function __construct($ruedas){
          $this->ruedas=$ruedas;
    }

    function set_ruedas($ruedas){
          $this->ruedas=$ruedas;
    }

    function get_ruedas(){
          return $this->ruedas;
    }
}

  class Avion extends Vehiculo{
    public $alas;

    function __construct($ruedas,$alas){
          parent::__construct($ruedas);
          $this->alas = $alas;
    }

    function get_alas(){
          return $this->alas;
    }

    function volar(){
          echo "<br>Estoy volando";
    }
```

```
}

$ryanair = new Avion(4,2);
echo $ryanair->get_ruedas();
echo "<br>";
echo $ryanair->get_alas();
$ryanair->volar();
?>
</body>
</html>
```

Observamos que funciona de forma similar a cómo lo hace en JavaScript. Podremos heredar los métodos y atributos del "padre" y declarar nuevos en el "hijo". Como curiosidad, si en JavaScript utilizamos **super()** para referirnos al "padre" de una clase, en PHP utilizaremos **parent::**

Conexiones

Una pieza interesante de la programación en PHP es la conectividad con HTML y JavaScript. Ya vimos que un archivo *.php nos permite introducir lenguaje HTML sin ningún tipo de complicación, pero con JavaScript será diferente, pues habrá que tener en cuenta, que las funciones de uno u otro no se ven entre sí, pues son dos lenguajes de programación diferente.

Pero, si bien no podremos usar dichas funciones directamente, si podremos realizar conexiones entre ambos lenguajes utilizando REST. Es muy común encontrarnos con formularios en HTML cuyas especificaciones y limitaciones se establecen en JavaScript pero es el documento PHP el que recibe los valores, los confirma y realiza las acciones requeridas.

Formularios

Vamos a comprobar cómo utilizar formularios creados directamente en HTML para enviar información a PHP:

index.html

```
<html>
<head>
<script src="https://code.jquery.com/jquery-3.6.0.js"
integrity="sha256-H+K7U5CnXl1h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk="
crossorigin="anonymous"></script>
</head>
<body>
```

```
    <form action="main.php" method="POST">
        Nombre: <input type="text" name="nombre"><br>
        Edad: <input type="text" name="edad"><br>
        <input type="submit">
    </form>
</body>
<script src="main.js"></script>
</html>
```

main.php

```
<html>
<body>
<?php
    $nombre = $_POST["nombre"];
    $edad = $_POST["edad"];
    echo "Hola $nombre <br>";
    echo "Tu edad es $edad<br>";
?>
</body>
</html>
```

Vemos que el formulario (index.html) tiene dos atributos, **action** que indica que página o acción debe realizar el formulario al recibir la señal **submit** y **method** que indica que método se usará al enviar el mismo formulario. Por otra parte, el documento PHP (main.php) recoge los valores "nombre" y "edad" (dados por el atributo **name** en el formulario), los almacena en variables y los muestra en el documento.

AJAX

Podemos utilizar AJAX con JQuery para enviar información y recibirla. Figuremos que en el siguiente código, index.html tiene una etiqueta 'p' con el identificador "respuesta":

main.js

```
function __main__(){
var nombre = prompt("Introduce tu nombre");
var edad = prompt("Introduce tu edad");
$.ajax({
    data:{"nombre":nombre,"edad":edad},
    url:'main.php',
    type:'post',
    success:function(response){
            $("#respuesta").html(response);
            }
    });
}

__main__();
```

main.php

```
<?php
    $nombre = $_POST["nombre"];
    $edad = $_POST["edad"];
    echo "Hola $nombre <br>";
    echo "Tu edad es $edad<br>";
?>
```

Similar a como teníamos en HTML, ahora JavaScript realizará la llamada. Esta vez, no cargará una página nueva, si no que el contenido de la respuesta es el código HTML que responde **main.php** usando **echo.**

Para enviar elementos como una API, podremos usar JSON y enviarlo utilizando **json_encode(*objeto*).** El código sería:

main.js

```
function __main__(){
var nombre = prompt("Introduce tu nombre");
var edad = prompt("Introduce tu edad");
$.ajax({
    data:{"nombre":nombre,"edad":edad},
    url:'main.php',
    type:'post',
    success:function(response){
            var respuesta = JSON.parse(response);
            console.log(respuesta);
            $("#respuesta").html("Hola "+respuesta.nombre+", tienes "+
respuesta.edad+ " años");
            }
    });
}

__main__();
```

main.php

```
<?php
class Usuario{
public $nombre;
public $edad;

function __construct($nombre, $edad){
    $this->nombre = $nombre;
    $this->edad = $edad;
}
}
```

```
    $user = new Usuario($_POST["nombre"],$_POST["edad"]);
    echo json_encode($user);
?>
```