



PHP y MySQL

Bloque I

- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - Literales – Variables – Constantes
 - Operadores
 - Estructuras de control
 - Funciones
 - Arrays
 - Archivos

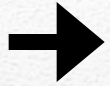
Contenido

Bloque II

- Formularios
- POO
- Acceso a bases de datos MySQL

Contenido

Bloque I



- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - Literales – Variables – Constantes
 - Operadores
 - Estructuras de control
 - Funciones
 - Arrays

Contenido

Arquitectura

Cliente - Servidor

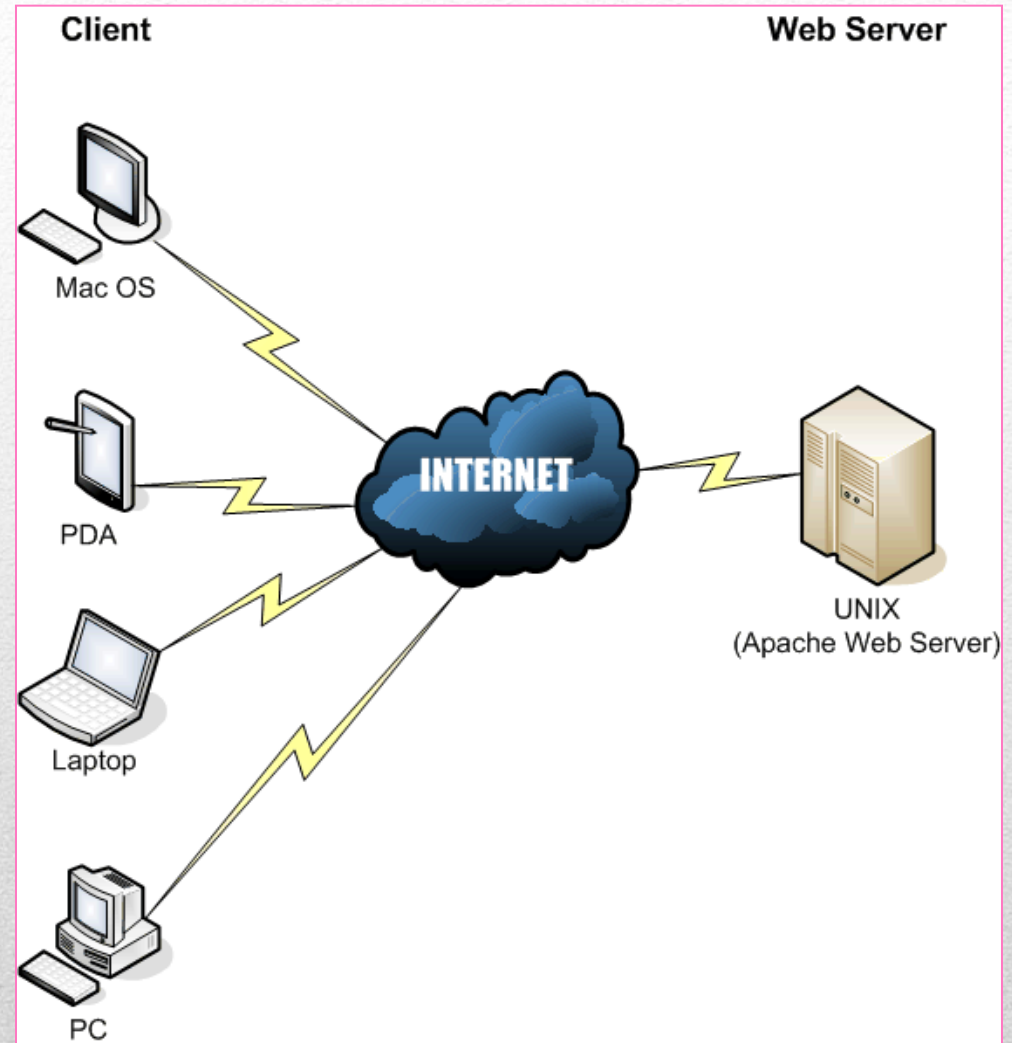
Cliente: Explorer, FireFox, Chrome, etc..

Servidor: Apache, IIS

Tecnología

Cliente: CSS, HTML, JavaScript

Servidor: Java, PHP, C#
Phyton



Bases del desarrollo web

- **Ciente**, navegador como **Internet Explorer, Chrome, Firefox, Opera, Safari, ...**

Los navegadores interactúan con el servidor a través de protocolos. Estos protocolos definen las reglas de intercambio de información entre el cliente y el servidor

- **Servidor**, software responsable de aceptar las solicitudes HTTP del cliente y de enviarle las respuestas (HTML, XML)

- **HTTP**, protocolo de transferencia de hipertexto.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. *¿Cómo sabe el servidor si la solicitud la hago yo o la hace otra persona?* ... (Cookies, Sesiones)

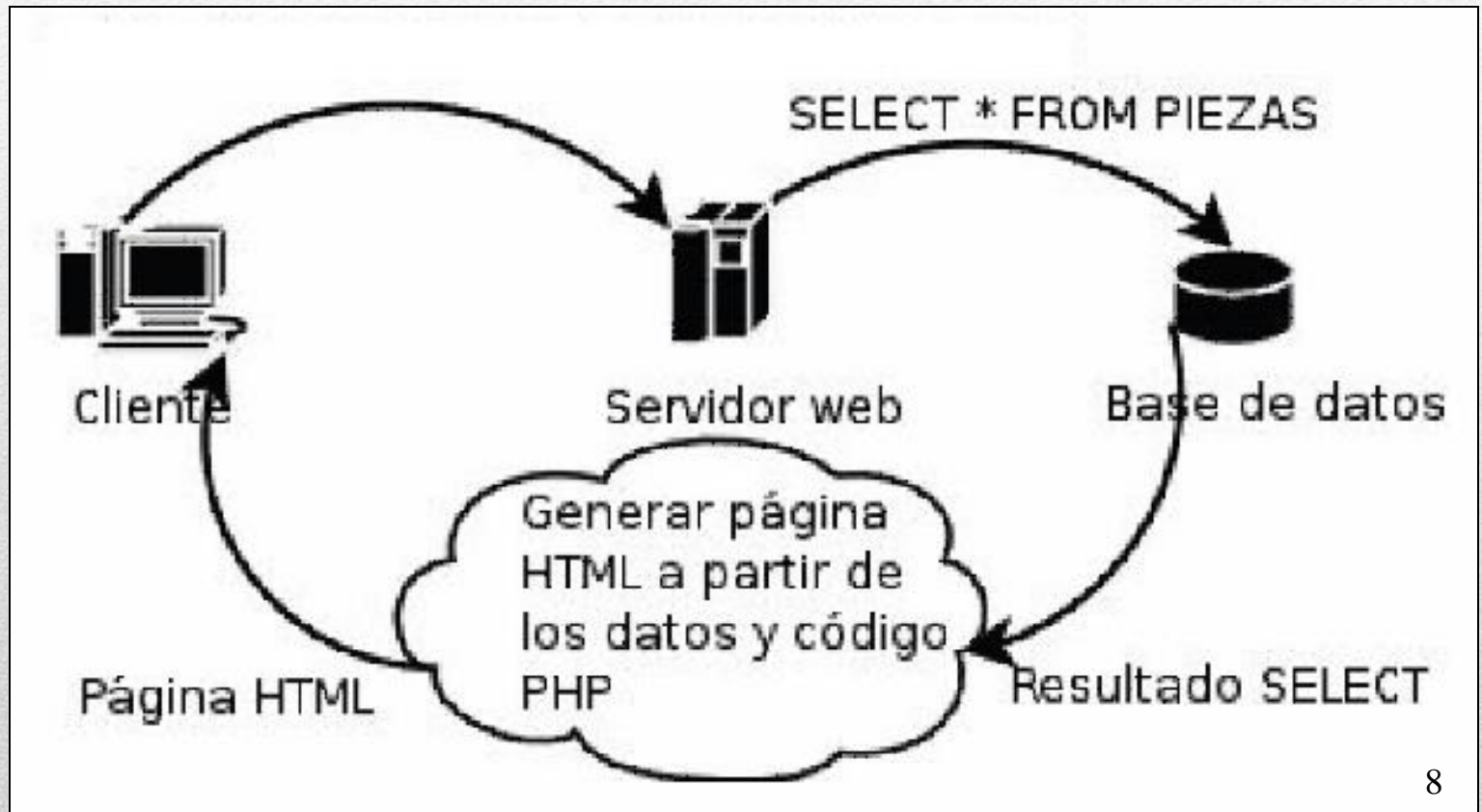
Bases del desarrollo web

- **Cookies y sesiones**

El protocolo HTTP es incapaz por sí solo de mantener el estado entre dos transacciones. El objetivo de las cookies y las sesiones en el servidor es precisamente identificar las solicitudes de un usuario y distinguirlas del resto

Bases del desarrollo web

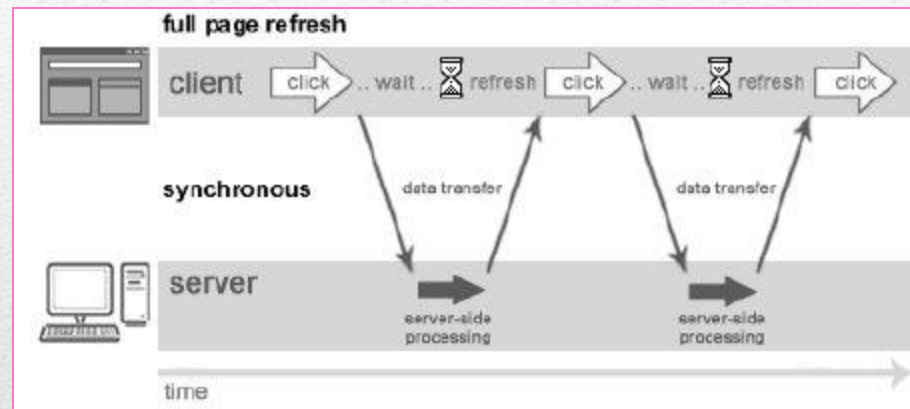
Diálogo entre cliente y servidor



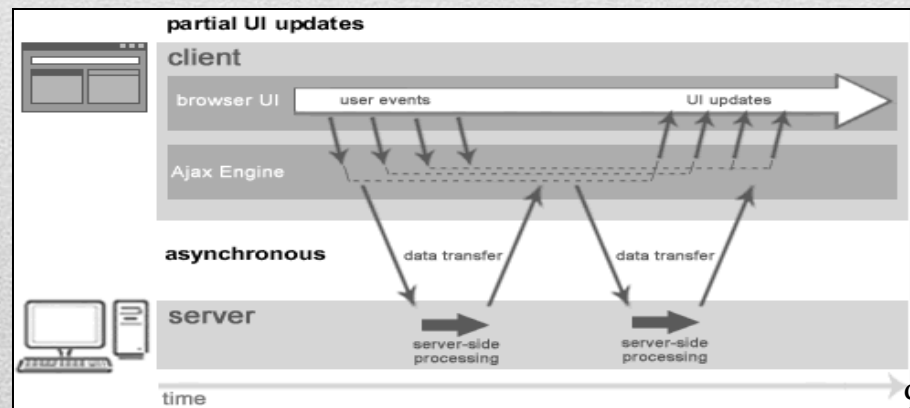
Bases del desarrollo web

Modelo Síncrono

Modelos de comunicación cliente



Modelo Asíncrono (Ajax)



Bloque I

- Bases del desarrollo web
 - Fundamentos de PHP
- ➔
- Qué es PHP
 - Literales – Variables – Constantes
 - Operadores
 - Estructuras de control
 - Funciones
 - Arrays

Contenido

- Lenguaje script, interpretado y de tipado débil
- Fue escrito por el danés [Rasmus Lerdorf](#) en 1994
- Es un lenguaje del lado del servidor
- Originalmente diseñado para producir webs

```
1
2  <html>
3    <head>
4      <title>Saluda en PHP</title>
5    </head>
6    <body>
7      <h3>Saluda en php</h3>
8      <?php echo "Hola mundo"; ?>
9      <hr>
10   </body>
11 </html>
```

Fundamentos de PHP

Qué es PHP

11

Ejercicio 0: Hacer un programa que muestre en pantalla información de PHP con la función `phpinfo()`.

Bloque I

- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - ➔ • Literales – Variables – Constantes
 - Operadores
 - Estructuras de control
 - Funciones
 - Arrays

Contenido

Variables

- En PHP **no es necesario declarar las variables**
- Las variables se crean al asignarles un valor
- Todas las variables en PHP empiezan por '\$' (\$productsList, \$title, \$connection)
- Es case-sensitive (\$quantity es distinto a \$Quantity)

```
2  
3  <?php  
4      $title      = "Curso de PHP";  
5      $quantity   = 3.2;  
6  ?>  
7
```

Fundamentos de PHP

Literales – Variables - Constantes

13

```

2
3 <?php
4     $username = "Alonso";
5     echo "Buenos días $username"; // Buenos días Alonso
6     echo 'Buenos días $username'; // Buenos días $username
7     ?>
8

```

Literales

- Dos formas de definir los literales : **comillas simples** y **comillas dobles**
- Las comillas dobles evalúan variables y expresiones dentro del literal; las comillas simples no

```

3 <?php
4     define('SUCCESSFUL', 0);
5     define('FAILED', 1);
6     define('PI', 3.141592);
7
8     echo 1 + PI;
9     ?>

```

Constantes

Fundamentos de PHP

Literales – Variables - Constantes

Variables. Tipos de datos

- **integer, double, string, boolean, array, object**
- PHP es un lenguaje de tipado débil
- El tipo de una variable vendrá determinado por el valor que se le asigne

```
$state = 0;           // integer
$state = "Successful"; // string
```
- Conversión de tipos explícita

```
$total_amount = (double) $quantity
```
- **Ejemplos** de variables de distintos tipos :

```
$variable_integer = 3;
$variable_double = 4.5;
$variable_boolean = true;           // (true, false)
$array[0] = 'value';
$array[1] = 23;
$person = new Person('Raul', 'López');
```
- PHP evalúa cualquier valor distinto de cero como true y cero como falso
- **Para mostrar el tipo y valor de una variable podemos utilizar la función `var_dump(expresion)`**

Fundamentos de PHP

Literales – Variables - Constantes

15

Literales – Variables - Constantes

Variables. Funciones relacionadas con los tipos

<code>getType(mixed variable)</code>	devuelve el tipo de la variable
<code>setType(mixed variable, string)</code>	establece el tipo de la variable
<code>is_array(variable)</code>	true si la variable es un array
<code>is_bool(variable)</code>	true si la variable es lógica
<code>is_double(variable)</code>	true si la variable es un double (<code>is_float</code> , <code>is_real</code>)
<code>is_int(variable)</code>	true si la variable es entera (<code>is_integer</code> , <code>is_long</code>)
<code>is_null(variable)</code>	true si la variable es null
<code>is_numeric(variable)</code>	true si la variable es un número o una cadena numérica
<code>is_object(variable)</code>	true si la variable es un objeto
<code>is_string(variable)</code>	true si la variable es una cadena de caracteres

Literales – Variables - Constantes

Variables. Otras funciones

- Las siguientes son funciones útiles, sobre todo, para comprobar si se enviaron las variables de formulario

<code>boolean isset(variable)</code>	true si la variable existe
<code>void unset(variable)</code>	suprime o elimina la variable
<code>boolean empty(variable)</code>	true si existe la variable y si contiene un valor <code>!= ""</code> y <code>!= 0</code>

Variables. Ámbito

- Global entre scripts

(la variable \$a estará disponible al interior del script incluido b.inc)

```
<?php  
$a = 1;  
include 'b.inc';  
?>
```

- Global a un script (definidas fuera de las funciones)

- Locales

IMPORTANTE limitar el ámbito de las variables todo lo posible

Literales – Variables - Constantes

18

Variables. Ámbito

- Para utilizar una variable global dentro de una función es necesario indicar antes de usar la variable, que nos referimos a una variable global, no a una local. Esto se hace con la palabra clave global

- Ejemplo

```
<?php
$a = 1;
$b = 2;
function Suma()
{
    global $a, $b;

    $b = $a + $b;
}
Suma();
echo $b;
?>
```

también sería posible

```
<?php
$a = 1;
$b = 2;

function Suma()
{
    $GLOBALS['b'] = $GLOB
ALS['a'] + $GLOBALS['b'];
}

Suma();
echo $b;
?>
```

Literales – Variables - Constantes

19

Variables predefinidas en PHP

\$_COOKIE	Array de cookies pasadas al script.
\$_ENV	Array de variables de entorno disponibles en php.
\$_FILES	Array de ficheros subidos al servidor vía http
\$_GET	Array de elementos de formulario enviados usando el método GET
\$_GLOBALS	Array que contiene todas las variables globales entre scripts
\$_POST	Array de elementos de formulario enviados usando el método POST
\$_REQUEST	Array que engloba elementos de \$_GET, \$_POST, y \$_COOKIE
\$_SERVER	Array de variables del servidor web en el que se ejecuta el script
\$_SESSION	Array de variables de sesión

Fundamentos de PHP

Literales – Variables - Constantes

20

Bloque I

- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - Literales – Variables – Constantes
 - • Operadores
 - Estructuras de control
 - Funciones
 - Arrays

Contenido

Operadores aritméticos

\$a + \$b	Suma
\$a - \$b	Resta
\$a * \$b	Multiplicación
\$a / \$b	División
\$a % \$b	Resto de la división de \$a por \$b
\$a++	Incrementa en 1 a \$a
\$a--	Resta 1 a \$a

Operadores de cadenas – Concatenación

```
2
3  <?php
4
5  $a = "Hola ";
6  $b = $a . "Mundo";    // Ahora $b contiene "Hola Mundo"
7
8  ?>
```

Ejercicio1: Concatena dos cadenas con el operador punto (.) e imprimir su resultado

Ejercicio 2: Hacer un programa que sume dos variables que almacenan dos números distintos.

Fundamentos de PHP

Operadores

Operadores de comparación

<code>\$a < \$b</code>	<code>\$a</code> menor que <code>\$b</code>
<code>\$a > \$b</code>	<code>\$a</code> mayor que <code>\$b</code>
<code>\$a <= \$b</code>	<code>\$a</code> menor o igual que <code>\$b</code>
<code>\$a >= \$b</code>	<code>\$a</code> mayor o igual que <code>\$b</code>
<code>\$a == \$b</code>	<code>\$a</code> igual que <code>\$b</code>
<code>\$a != \$b</code>	<code>\$a</code> distinto que <code>\$b</code>

Operadores lógicos

<code>\$a AND \$b</code>	Verdadero si ambos son verdadero
<code>\$a && \$b</code>	Verdadero si ambos son verdadero
<code>\$a OR \$b</code>	Verdadero si alguno de los dos es verdadero
<code>\$a \$b</code>	Verdadero si alguno de los dos es verdadero
<code>\$a XOR \$b</code>	Verdadero si sólo uno de los dos es verdadero
<code>!\$a</code>	Verdadero si <code>\$a</code> es falso, y recíprocamente

Fundamentos de PHP

Operadores

Operadores de asignación

<code>\$a = \$b</code>	Asigna a \$a el contenido de \$b
<code>\$a += \$b</code>	Asigna a \$a la suma de \$b + \$a
<code>\$a -= \$b</code>	Asigna a \$a la resta de \$a - \$b
<code>\$a *= \$b</code>	Asigna a \$a la multiplicación de \$a por \$b
<code>\$a /= \$b</code>	Asigna a \$a la división de \$a entre \$b
<code>\$a .= \$b</code>	Asigna a \$a la concatenación de \$a seguida por \$b

Operador de referencia (&)

- Este operador permite obtener la referencia o dirección de memoria de una variable

```
2 <?php
3     $name1 = "Bob";
4     $name2 = & $name1;
5     $name2 .= " Martin";
6     echo $name1."<br>";
7     echo $name2."<br>";
8 ?>
```

\$name1 →

\$name2 →

Bob

Fundamentos de PHP

Operadores

Operadores de supresión de errores

- Este operador suprimirá el error de manera que no se muestre por pantalla

```
2  <?php
3      $a = @(57/0);
4      ?>
```

Fundamentos de PHP

Operadores

Bloque I

- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - Literales – Variables – Constantes
 - Operadores
 - • Estructuras de control
 - Funciones
 - Arrays

Contenido

if ... else ..., if ... elseif ... else

if (condición) { sentencias }

if (condición) { sentencias } else { sentencias }

if (condición) { sentencias } elseif (condición) { sentencias } ...

- Las llaves son necesarias cuando hay más de una sentencia

```
1
2  <?php
3      if ($quantity == 0)
4          echo "No ha solicitado nada en la página anterior";
5  ?>
6
```

```
2  <?php
3      if ($quantity == 0) {
4          echo "No ha solicitado nada en la página anterior";
5      }
6      elseif ($quantity > 0) {
7          echo "la cantidad solicitada es correcta";
8      }
9  ?>
```

Fundamentos de PHP

Estructuras de control

27

switch

```
switch ($variable) {  
    case Valor1: ...; break;  
    case Valor2: ...; break;  
    [default:   ...; break;]  
}
```

```
2  <?php  
3  switch ($season)  
4  {  
5      case Spring: echo "Es primavera"; break;  
6      case Summer: echo "Es Verano"; break;  
7      case Autumn: echo "Es Otoño"; break;  
8      case Winter: echo "Es Invierno"; break;  
9      default: echo "estación incorrecta";  
10 }  
11 ?>
```

Fundamentos de PHP

Estructuras de control

28

while

while (condición) sentencia;

while (condición) { sentencia1; sentencia2; ... sentencian; }

- Operadores **break** y **continue**;

```
3  <?php
4      $i = 1;
5      while($i <= 3) {
6          echo 'La variable $i vale: ' . $i ;
7          $i += 1;
8      }
9  ?>
```

Fundamentos de PHP

Estructuras de control

do while

```
1
2  <?php
3      $i = 5;
4      $n = 1;
5      do {
6          $n = $n * $i;
7          $i -= 1;
8      }while($i > 1);
9      echo "5! es igual a: " . $n
10  ?>
```

Fundamentos de PHP

Estructuras de control

30

for

for (var = valor_inicial; condición; var++) sentencia;

for (var = valor_inicial; condición; var++) { sentencias }

```
1
2  <?php
3      for($i = 1; $i <= 5; $i++)
4          echo $i . "<br/>";
5  ?>
6
```

Fundamentos de PHP

Estructuras de control

foreach

```
foreach(variable_array as $value) sentencia;  
foreach(variable_array as $value) { sentencias }  
foreach(variable_array as $key => $value) sentencia;  
foreach(variable_array as $key => $value) { sentencias }
```

- El bucle itera sobre la lista devolviendo un elemento de la lista en cada iteración

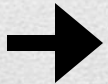
```
1  
2  <?php  
3      $numbers = array(1, 2, 3, 17);  
4      foreach($numbers as $number) {  
5          print "Valor actual de \$numbers: $number.\n";  
6      }  
7  ?>  
8
```

Fundamentos de PHP

Estructuras de control

Bloque I

- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - Literales – Variables – Constantes
 - Operadores
 - Estructuras de control
 - Funciones
 - Arrays



Contenido

Funciones

```
function nombre_funcion(param1,...paramn) {  
    sentencias  
    [return valor;]  
}
```

```
1  
2  <?php  
3  function connect($server, $user, $password = '') {  
4      $connection = mysql_pconnect('localhost', 'root');  
5      if (!isset($connection)) echo "<p>No se ha podido establecer la conexión  
6  }  
7  ?>
```

Ejercicio 3. Mostrar en pantalla una tabla de 10 por 10 con los números del 1 al 100

Ejercicio 4. Igual que el ejercicio 3, pero colorear las filas alternando gris y blanco. Además, el tamaño será una constante: define(TAM, 10)

Funciones

Paso de parámetros

- **Todos los parámetros son por valor** si no se especifica lo contrario
- Para pasar una variable por **referencia** se antepone **&**
- Se permiten **parámetros por defecto** (el parámetro por defecto tiene que estar a la derecha de cualquier parámetro sin valor)
- PHP permite un **número ilimitado de parámetros** (ninguna sintaxis especial)

`func_num_args():`

devuelve el nº de args pasados a la función

`func_get_arg(int num_arg):`

devuelve un arg de la lista

`func_get_args():`

devuelve un array copia de la lista de args

Funciones

Paso de parámetros por valor

```
2
3  ▢ function ejemplo_por_valor($a, $b) {
4      $a++;
5      $b++;
6      echo "Dentro de la función el valor de a es $a y el de b es $b";
7  }
```

Paso de parámetros por referencia

```
8
9  ▢ function ejemplo_por_referencia(& $a, & $b) {
10      $a++;
11      $b++;
12      echo "Dentro de la función el valor de a es $a y el de b es $b";
13  }
```

Funciones

Parámetros ilimitados

```
3  function media() {
4      $result = 0;
5      $num_args = func_num_args();
6      if ($num_args == 0) return 0;
7
8      foreach (func_get_args() as $arg) {
9          $result += $arg;
10     }
11     return $result / $num_args;
12 }
13
14 echo "la media de 1 + 2 + 3 + 4 es: ".media(1,2,3,4);
15
```

Parámetros por defecto

```
2
3  function makecoffee($type = "cappucino")
4  {
5      return "Hacer una taza de $type.\n";
6  }
7  echo makecoffee ();
8  echo makecoffee ("espresso");
9
```

Funciones

return

- La palabra reservada ***return*** permite devolver valores de las funciones
- Si lo que se quiere es devolver una referencia se tiene que usar **&** tanto en la **declaración de la función** como en la **asignación del valor de retorno a una variable**

```
function & returns_reference() {  
    return $someref;  
}
```

```
$newref = & returns_reference();
```

Ejercicio 4: mostrar una tabla de 4 por 4 que muestre las primeras 4 potencias de los números del uno 1 al 4 (hacer una función que las calcule invocando la función pow). Recuerda en PHP las funciones hay que definir las antes de invocarlas.

Funciones

include, require

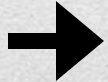
- Permiten cargar funciones y variables declaradas en otros scripts (reutilización)
- Similares a #include de C
- Se pueden incluir en cualquier parte del script
- Ambas tiene la misma funcionalidad, con pequeñas diferencias:
 - **include**, si no puede cargar el fichero, no provocará ningún error
`<? include ("archivo") ?>`
 - **require**, si no puede cargar el fichero, provocará un error fatal
`<? require("archivo") ?>`

Ejercicio 5: Igual que el 4, pero define la función de la potencia en un archivo a parte y utiliza require para poder usar dicha función en otro archivo.

Funciones

Bloque I

- Bases del desarrollo web
- Fundamentos de PHP
 - Qué es PHP
 - Literales – Variables – Constantes
 - Operadores
 - Estructuras de control
 - Funciones
 - Arrays



Contenido

- Conocidos como **matrices** o **arrays** en PHP
- Muy utilizados (estructuras de datos, opciones de configuración, idiomas)
- En PHP existen dos tipos de arrays:
 - arrays escalares, cuyo **índice** es un entero
 - arrays asociativos, cuyo **índice** es una string

Array escalar

	amigos[0]	amigos[1]	amigos[2]
índice	0	1	2
valor	Ana	Juan	Pepe

Array asociativo

	temp['Junio']	temp['Julio']	temp['Agosto']
índice	Junio	Julio	Agosto
valor	28	29	30

Arrays

Creación e inicialización de arrays

Operador []

```
2  
3  $friends[];  
4
```

Array vacío!

```
3  
4  $friends[0] = 'Javier';  
5  $friends[1] = 'Agustín';  
6  $friends[2] = 'Santiago';  
7  $friends[3] = 'Jorge';
```

```
9  $temperatures['Junio'] = 25;  
10 $temperatures['Julio'] = 28;  
11 $temperatures['Agosto'] = 30;  
12
```

```
13 $values[] = 'cero';  
14 $values[] = 1,618;  
15 $values[] = 7;  
16
```

42

¡ Débilmente tipado !

Arrays

Creación e inicialización de arrays

Constructor array

```
2  
3     $accounts = array();  
4
```

Array vacío!

```
5  
6     $friends  = array('Agustin',  
7                       'Santiago',  
8                       'Jorge');  
9
```

```
10  
11     $temperatures = array('Junio' => 25,  
12                           'Julio' => 28,  
13                           'Agosto' => 30);  
14
```

```
15  
16     $values = array('cero', 1, 618, 7);  
17
```

¡ Débilmente tipado !

Arrays

Arrays multidimensionales

```
18
19 $board[0][0] = 1;
20 $board[0][1] = 0;
21 $board[0][2] = 0;
22
23 $temperatures = array(
24     "spring"    => array(2006 => 20, 2007 => 24),
25     "summer"   => array(2006 => 31, 2007 => 30),
26     "winter"    => array(2006 => 18, 2007 => 20),
27 );
28 $temperatures['spring'][2006];
29 $temperatures['winter'][2007];
30
```

Arrays

Recorrido

```
for (var = valor_inicial; condición; var++) sentencia;  
for (var = valor_inicial; condición; var++) { sentencias }
```

```
31  
32 $seasons = array('Spring', 'Summer', 'Autumn', 'Winter');  
33  
34 // for -- count: devuelve el número de elementos del array  
35 for ($i = 0; $i < count($seasons); $i++) {  
36     echo "$seasons[$i]<br>";  
37 }  
38  
39 >> Spring  
40 >> Summer  
41 >> Autumn  
42 >> Winter
```

Arrays

Recorrido

```
foreach(variable_array as $value) sentencia;  
foreach(variable_array as $value) { sentencias }  
foreach(variable_array as $key => $value) sentencia;  
foreach(variable_array as $key => $value) { sentencias }
```

```
44  
45 // foreach  
46 foreach ($seasons as $season) {  
47     echo "$season<br>";  
48 }
```



```
49  
50 >> Spring  
51 >> Summer  
52 >> Autumn  
53 >> Winter  
54
```

```
55  
56 foreach ($seasons as $key => $value) {  
57     echo "[$key] => $value<br>";  
58 }
```



```
55  
56 [0] => Spring  
57 [1] => Summer  
58 [2] => Autumn  
59 [3] => Winter  
60
```

Arrays

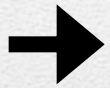
Funciones relacionadas con los arrays

<code>unset(\$temperatures['Junio'])</code>	elimina 'Junio' del array
<code>unset(\$temperatures)</code>	elimina todo el array
<code>list(\$primero, \$segundo) = \$numbers;</code>	asignación de varias variables en una línea

Ejercicio 6: Crea un array con los días de la semana, un array asociativo con los meses del año y su respectivo número de días(no bisiestos), utilizando estos arrays y conociendo que el 1 de enero del 2017 será domingo muestra el calendario de marzo del 2017

Arrays

Bloque II



- Formularios
- POO
- Acceso a bases de datos MySQL

Contenido

Formularios HTML

```
<form action="" method="">
```

```
...
```

```
</form>
```

- **action** define el tipo de acción a llevar a cabo con el formulario. Existen dos posibilidades:
 - el formulario es enviado a una dirección de correo electrónico
 - el formulario es enviado a un programa o script que procesa su contenido
- **method** se encarga de especificar la forma en la que el formulario es enviado. Los dos valores posibles que puede tomar este atributo son *post* y *get*

Formularios

Elementos de formulario

```
<input name="nombre" type="text">
```

```
<input name="nombre" type="password">
```

```
<textarea name="nombre" rows=n_filas cols=n_columnas></textarea>
```

```
<select name="nombre">  
  <option>opción1</option>  
  ...  
  <option>opciónn</option>  
</select>
```

```
<input name="nombre" type="radio" value="valor">Texto
```

```
<input name="nombre" type="checkbox">Texto
```

Formularios

Envío y borrado en formularios HTML

```
<input type="submit" value="Enviar">
```

```
<input type="reset" value="Borrar">
```

- **Ejemplo** de formulario:

```
<form action="login.php" method="post" name="login">
```

```
  Usuario: <input name="user" type="text">
```

```
  Contraseña: <input name="pass" type="password">
```

```
  <input type="submit" value="Login">
```

```
</form>
```

Formularios

Usuario: Contraseña:

Formularios

52

Recepción de las variables de formulario

`$_GET['variable']`

`$_POST['variable']`

- Estos arrays asociativos(con el name del input como indice) contienen las variables transferidas de una página a otra a través de un formulario. Dependiendo del método utilizado (get o post) en el formulario, las variables estarán en uno u otro

Ejercicio 7: Modifica el ejercicio 6, de tal forma que el usuario pueda seleccionar a través de un formulario, cual es el mes de 2017 que desea ver en el calendario.

Ejercicio 8: Realiza un formulario de registro con los siguientes campos: nombre de usuario, email, contraseña, edad, sexo e intereses. Comprueba que todos los campos estén correctamente rellenos. Si todo esta bien muestra que el usuario se ha registrado correctamente y todos los valores del registro.

Los valores correctos de la contraseña será que tenga mas de 6 caracteres, la edad debe ser un digito del 1 al 120, el email debe contener un@. El sexo será un select, los intereses serán al menos 4 checkboxes a vuestra elección. Investiga en <http://php.net/manual/es> las funciones que necesites que aun no hayamos visto.

Formularios

- Html permite cargar archivos en el servidor.
- PHP permite trabajar con estos archivos en el servidor
- Para enviar archivos al servidor necesitamos un formulario de type multiple.

Archivos

- EL ATRIBUTO ENCTYPE DEL ELEMENTO HTML <form>:
- Sintaxis

<form

ENCTYPE=“multipart/form-data”

ACTION=“nombre.php” method=“POST”>

Enviar este archivo: <INPUT NAME=“archivo”
TYPE=“file”>

</form>

PASO PARA CREAR UN FORMULARIO MULTIPLE

- Esta es una matriz super global, eso quiere decir que puede ser usada en cualquier parte del código..
- Cuando usamos controles de archivos no necesitamos las matrices `$_POST`, `$_GET` Y `$_REQUEST`

La matriz `$_FILES`

Subida de archivos al servidor

php.ini

```
;;;;;;;;;;  
; File Uploads ;  
;;;;;;;;;;  
; Whether to allow HTTP file uploads.  
file_uploads = On  
  
; Temporary directory for HTTP uploaded files (will use  
; system default if not specified).  
;upload_tmp_dir =  
  
; Maximum allowed size for uploaded files.  
upload_max_filesize = 2M
```

formulario

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE"  
VALUE='1024000'>(en bytes)  
<INPUT TYPE="FILE" NAME="archivo">
```

servidor

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="102400">
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```

- La variable `$_FILES` contiene toda la información del archivo subido:
 - `$_FILES['imagen']['name']`
Nombre original del archivo en la máquina cliente
 - `$_FILES['imagen']['type']`
Tipo [mime](#) del archivo. Por ejemplo, "image/gif"
 - `$_FILES['imagen']['size']`
Tamaño en bytes del archivo subido
 - `$_FILES['imagen']['tmp_name']`
Nombre del archivo temporal en el que se almacena el archivo subido en el servidor
 - `$_FILES['imagen']['error']`
Código de error asociado al archivo subido

- `<?php`
// En versiones de PHP anteriores a la 4.1.0, debería utilizarse `$HTTP_POST_FILE`
S en lugar
// de `$_FILES`.

`$dir_subida = './uploads/';`
`$fichero_subido = $dir_subida . $_FILES['fichero_usuario']['name'];`

`echo '<pre>';` //<pre> para que se vean los saltos de linea y tabulaciones en html

`if (move_uploaded_file($_FILES['fichero_usuario']['tmp_name'], $fichero_subido)) {`
 `echo "El fichero es válido y se subió con éxito.\n";`
`} else {`
 `echo "¡Posible ataque de subida de ficheros!\n";`
`}`
`echo 'Más información de depuración:';`
`print_r($_FILES);` //print_r — Imprime información legible para humanos sobre una variable
`print "</pre>";`

`?>`

Subida de archivos en php con \$FILES

59

Ejercicio 9: igual que el ejercicio 8, pero añadiendo al formulario la opción de subir una imagen de perfil del usuario que ocupe un máximo de 1 MB, en formato jpg, y que quede guardada en la carpeta de perfiles (que debes crear dentro de htdocs). Cuando el usuario se registre correctamente también debes mostrar su foto de perfil.

- Veamos otras funciones básicas de archivos, abrir (fopen), cerrar (fclose), leer (fgets) y escribir (fputs).
- Estas cuatro nos solventaran la mayoría de problemas que nos surgan con respecto al acceso a archivos.

Funciones de acceso a archivos

- Con esta función abrimos un archivo, bien sea local o una dirección de internet (http:// o ftp://).
- La función **fopen** nos devuelve un valor numérico (indicador de archivo) de tipo **integer** que nos servirá para hacer referencia al archivo abierto.

fopen (archivo, modo)

- **r** solo lectura
- **r+** lectura y escritura
- **w** solo escritura. Sino existe el archivo lo crea, si ya existe lo re-escribe
- **w+** lectura y escritura. Sino existe el archivo lo crea, si ya existe le re-escribe
- **a** solo lectura. Sino existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.
- **a+** lectura y escritura. Sino existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.

Con fopen podemos abrir un archivo de los siguientes modos

```
<?PHP
```

```
if ( ! fopen("http://www.ciberaula.com/", "r")) {  
echo "El archivo no se puede abrir\n";  
exit;  
}  
?>
```

Abriendo un archivo
con el protocolo
http.

Los modos **r**, **r+**, **w**, **w+** colocan el puntero de lectura/escritura a principio del archivo, los modos **a**, **a+** lo colocan al final.

Ejemplo de fopen()

- La función **fgets** nos devuelve una cadena con la longitud especificada del archivo al que apunta el indicador de archivo.

```
<?PHP
```

```
$archivo = fopen("data.txt" , "r");
```

```
if ($archivo) {
```

```
while (!feof($archivo)) {
```

```
$linea = fgets($archivo, 255);
```

```
echo $linea;
```

```
}
```

```
}
```

```
fclose ($archivo)
```

```
?>
```

La función **feof** devuelve TRUE si el puntero de lectura/escritura se encuentra al final del archivo, y FALSE en caso contrario.

fgets (indicador_archivo, longitud)

- La función **fputs** escribe una cadena en el archivo indicado. Para escribir en un archivo este debe haber sido previamente abierto. La función **fputs** devuelve TRUE si se ha escrito con éxito, en caso contrario devuelve FALSE.

```
<?PHP
$archivo = fopen("./data.txt" , "w");
if ($archivo) {
    fputs ($archivo, "Hola Mundo");
}
fclose ($archivo);
?>
```

fputs (indicador_archivo, cadena)

- Esta función devuelve TRUE si el archivo especificado existe, y FALSE en caso contrario.

```
<?PHP
```

```
if (file_exists("data.txt")) {  
    echo "El archivo existe";  
} else {  
    echo "El archivo NO existe";  
}  
?>
```

file_exists (archivo)



- La función copy : copia un archivo de un lugar (origen) a otro (destino), devuelve TRUE si la copia a tenido éxito y FALSE en caso contrario.
<?PHP
if (copy("./data.txt", "./tmp/data.txt")) {
echo "El archivo ha sido copiado con éxito";
} else {
echo "El archivo NO se copio" style="margin-left: 50">echo "El archivo NO se ha
podido copiar";
}
?>

copy (origen, destino)

- Para trabajar con directorios las funciones principales que tenemos son:
- `Opendir()` -> abre el directorio
- `Closedir()` -> cierra el directorio
- `Readdir()` -> lee los archivos del directorio secuencialmente
- `Rewindir()` -> devuelve el puntero de lectura al principio del directorio

Directorios

68

- `resource opendir (string $path [, resource $context])`
- Abre un gestor de directorio para ser usado con llamadas posteriores.
- Devuelve el gestor de directorio, necesario para el resto de llamadas.

Opendir (ruta, [contexto])

69

- Cierra el gestor de directorio que recibe por parametros

closedir(directorio)

70

- Devuelve el nombre de la siguiente entrada del directorio. Las entradas son devueltas en el orden en que fueron almacenadas por el sistema de ficheros.
- Retorna el nombre de la entrada en caso de éxito o FALSE en caso de error.

readdir(directorio)

71

- ```
if ($gestor = opendir('/path/to/files')) {
 echo "Gestor de directorio: $gestor\n";
 echo "Entradas:\n";

 /* Esta es la forma correcta de iterar sobre el directorio. */
 while (false !== ($entrada = readdir($gestor))) {
 echo "$entrada\n";
 }
 closedir($gestor);
}
```


Ejercicio10: siguiendo el ejercicio nueve, crea una página donde se visualice una pagina con una cuadrícula de cuatro columnas que muestre todas las fotos y nombres de usuario de todos los perfiles registrados con al aplicación

# Directorios, ejemplo

72



## Bloque II

- 
- Formularios
  - POO
  - Acceso a bases de datos MySQL

# Contenido

- Desde la versión de PHP 5.0 se ha ido mejorando progresivamente la implementación del uso de Clases y Objetos con el fin de permitir el desarrollo de páginas y aplicaciones web mediante **Programación Orientada a Objetos (OOP** en inglés).
- El crear una página o aplicación web haciendo uso de una correcta Programación Orientada a Objetos nos facilitará en gran medida la organización y reutilización del código y la depuración de errores, con lo cual nuestros proyectos web podrán ser mantenidos y ampliados de forma mucho más sencilla.

# PHP Orientación a Objetos

74



- A diferencia de JS, en PHP existen Clases.
- Las clases definen la estructura de los objetos.
- Los objetos ya no serán objetos sin más con propiedades y funciones que se definan de forma dinámica, si no que objetos de una determinada clase siempre tendrán las propiedades, constantes y metodos definidos en esa clase.

# PHP Orientación a Objetos

75

- Definir clases en PHP.

- La definición básica de una clase comienza con la palabra reservada *class*, seguida de un nombre de clase, y continuando con un par de llaves que encierran las definiciones de las propiedades y métodos pertenecientes a dicha clase.

- Ejemplo

- ```
<?php
class ClaseSencilla
{
    // Declaración de una propiedad, con visibilidad pública
    public $var = 'un valor predeterminado';

    // Declaración de un método con visibilidad pública
    public function mostrarVar() {
        echo $this->var; //Uso de // $this para referirse al propio objeto y acceder a sus prop
                        // y funciones
    }
}
?>
```

Definición Clases, propiedades y funciones

76

- Dentro de la clase se define el constructor o destructor de los objetos de esa clase:
- Ejemplo

// Constructor:

```
function __construct() { //Dos guiones bajos
    echo "<p>En el Constructor de la Clase</p>";
}
```

/*

// El constructor también podía hacerse así (con el mismo nombre del Clase) pero esta //forma ha sido deprecada en PHP 7. no se aconseja su uso.

```
function Persona() {
    echo "<p>En el Constructor de la Clase</p>";
}
```

*/

// Destructor:

```
function __destruct() {
    echo "<p>En el Destructor de la Clase</p>";
}
```

Constructores

77

- Es posible definir valores constantes en función de cada clase manteniéndola invariable. Las constantes se diferencian de las variables comunes en que no utilizan el símbolo \$ al declararlas o emplearlas. La visibilidad predeterminada de las constantes de clase es *public*.
- El valor debe ser una expresión constante, no (por ejemplo) una variable, una propiedad o una llamada a una función.
- Ejemplo:
- ```
<?php
class MiClase
{
 const CONSTANTE = 'valor constante';
}

?>
```

# Constantes

78



- Un objeto es una instancia de una clase
- Cada clase se suele definir en un archivo independiente, por lo que primero es:

```
<?php
```

```
require_once("Persona.php");
```

- Para crear el objeto debemos invocar al constructor de la clase:

```
$objPersona = new Persona();
```

- Para acceder a las propiedades y funciones, se utiliza ->:

```
$objPersona->setNombre("MARTINA"); //función
```

```
$objPersona->nombre // propiedad
```

# Creación de Objetos

79

- Para acceder a las constantes, se utiliza el operador :: y hay varias opciones.

- 1ª acceder a través del nombre de la clase:

```
echo MiClase::CONSTANTE . "\n";
```

- 2ª Con una variable que tenga el nombre de la clase como valor:

```
$nombreclase = "MiClase";
echo $nombreclase::CONSTANTE . "\n"; // A partir de PHP 5.3.0
```

- 3ª Con una instancia de la clase(objeto):

```
$clase = new MiClase();
echo $clase::CONSTANTE . "\n"; // A partir de PHP 5.3.0
```

- Si queremos acceder a la constante dentro de la propia clase donde se define utilizamos la palabra clave self:

```
<?php class MiClase {
 const CONSTANTE = 'valor constante';
 function mostrarConstante() {
 echo self::CONSTANTE . "\n";
 }
}?>
```

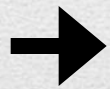
- Con los métodos y atributos staticos de una clase se accede de la misma forma que a sus constantes



- Ejercicio 11:
  - Encapsula toda la información de los datos del ejercicio 9 en una clase denominada usuario que debes almacenar en un usuario.php. Cambia la propiedad edad por fecha de nacimiento.(Investiga el uso de la clase DateTime en php para trabajar con fechas)
  - Además de las propiedades del formulario, añade los siguientes métodos:
    - toString(): esta función devuelve una cadena de caracteres con todos los datos del usuario
    - Edad(): Devuelve la edad del usuario si este es mayor de edad, sino devuelve false.
    - Login(\$pass): Recibe una contraseña por parametros, la compara con la del usuario sin tener en cuenta Mayusculas o minusculas y devuelve true si coinciden y false si no
  - También añade las siguientes constantes:
    - HOMBRE = h, MUJER=m ,MAYORIA\_EDAD=18
  - Modifica tu código de control de formulario del ejercicio 10, para utilizar la clase usuario que acabas de crear.(crear un objeto usuario y mostrar todas las propiedades y salida de sus funciones)

## Bloque II

- Formularios
- POO



- Acceso a bases de datos MySQL

# Contenido



## MySQL y phpMyAdmin

- MySQL es un sistema gestor de bases de datos relacionales multiusuario
- phpMyAdmin es una herramienta para la administración de MySQL

# Acceso a bases de datos MySQL

---

MySQLi es realmente MySQL Improved Extension.

Implementa funcionalidad para conectarse a bases de datos MySQL 4.1+

Las antiguas funciones `mysql_XXX()`, ahora serán orientadas a objetos.

# MySQLi



## Conéctandonos a una base de datos

```
$mysqli = new mysqli("localhost", "user", "password", "db_name");
```

## Comprobando el estado de la conexión

```
if ($mysqli->connect_errno()) {

 printf("Error en la conexión: %s\n", $mysqli->connect_err());
 exit();

}
```

## Cerrando la conexión con una base de datos

```
$mysqli = new mysqli("localhost", "user", "password", "db_name");
```

```
$mysqli->close();
```

## Ejecutando una petición contra la base de datos

```
$query = "SELECT * FROM Ciudad";
```

```
$result = $mysqli->query($query);
```

```
while($row = $result->fetch_array())
{
 echo $row['codigo_ciudad'];
}
```



## Utilizando prepared staments

```
$sql = 'INSERT INTO tablename VALUES (?, ?)';
$stmt = $mysqli->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param("is", 1, "test");
$stmt->execute();
$stmt->close();
```

El primer parámetro de bind\_param especifica el tipo de valor de los siguientes con un carácter por cada uno

| i | la variable correspondiente es de tipo entero                 |
|---|---------------------------------------------------------------|
| d | la variable correspondiente es de tipo double                 |
| s | la variable correspondiente es de tipo string                 |
| b | la variable correspondiente es un blob y se envía en paquetes |

# Acceso a bases de datos MySQL

MySQLi no soporta transacciones, en su lugar debemos confiar en COMMIT y ROLLBACK de la base de datos.

MySQLi por defecto implementa auto-commit, despues de cualquier consulta de hace automáticamente un COMMIT a la base de datos.

Para desactivar este comportamiento

```
$mysqli->autocommit(FALSE);
```

Para forzar COMMIT y  
ROLLBACK

```
$mysqli->commit();
```

```
$mysqli->rollback();
```

# Acceso a base de datos MySQL

88



- Cada vez que creamos una conexión, estamos creando un nuevo objeto mysql para utilizar la base de datos.
- Cuando servimos a numerosos clientes se crean números objetos mysql para servir lo que necesiten en la base de datos.
- Solamente una conexión puede acceder a la base de datos en un determinado tiempo, por lo que no es necesario ocupar espacio en memoria de más.
- Existe un patrón creado para este tipo de problemas de la orientación a objetos, este patrón se asegura de que solamente haya una instancia de un determinado objeto en toda la ejecución de la aplicación.
- Para llegar a este objetivo se basa en el uso de statics

## **Patron SINGLETON para las conexiones** 89

- Estático significa que referencia siempre a la misma dirección de memoria. Una vez que creamos una variable o un método estático no podemos crearlo de nuevo, aunque si se puede cambiar su valor.
- Declarar propiedades o métodos de clases como estáticos los hacen accesibles sin la necesidad de instanciar la clase.
- Debido a que los métodos estáticos se pueden invocar sin tener creada una instancia del objeto, la seudovariable `$this` no está disponible dentro de los métodos declarados como estáticos.

# statics

90



- [Esta](#) es una clase que envuelve una conexión según el patron singleton en php
- Se usaría de la siguiente forma:  

```
$db = Database::getInstance();
$mysqli = $db->getConnection();
$sql_query = "SELECT foo FROM";
$result = $mysqli->query($sql_query);
```

### Ejercicio 12:

- Actualiza el ejercicio 11 almacenando todos los usuarios en base de datos.
- Actualiza el ejercicio 10 accediendo a la información de las imágenes de perfil de usuario desde la base de datos en lugar de examinando la carpeta de uploads.
- Utiliza el patrón singleton