Apuntes PHP

Hay que entender que PHP es un lenguaje que se utiliza en la para el desarrollo web y se ejecuta en el servidor, no en la memoria de tu máquina. Para ejecutar este será necesario que se cree un servidor local, ya que sin este, nuestro navegador no puede interpretar el código PHP. Una herramienta para la creación de este es XAMPP.

Que puede hacer PHP:

- Genera contenido dinámico en una página.
- CRUD y cerrar archivos en el servidor.
- CRUD en la base de datos,
- Puedes mostrar imágenes, PDFs y cualquier tipo de texto
- …

Sintaxis básica

TODO código de **PHP empieza y acaba de la siguiente forma**

```php
<?php
// PHP code goes here
?>
```

Normalmente, un archivo PHP contiene código HTML, por eso la mayoría de archivos se ven de la siguiente forma:

```php
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo 'Hello World!';
?>

</body>
</html>
```

Case-insensitive

```
ECHO 'Hello World!<br>';

echo 'Hello World!<br>';
```

Estas dos líneas funcionan exactamente de la misma forma, ya que el lenguaje ignora las mayúsculas en funciones y clases.

PERO, las variables **SI** son case-sensitive!

Comentarios

Funcionan de forma similar que en Java:

```
// This is a single-line comment
# This is also a single-line comment

/* This is a
multi-line
comment */
```

Variables

Las variables en PHP no hace falta ponerles un tipo, cuando le añades el valor, esta automáticamente detecta el tipo. Una variable puede cambiar de tipo mediante se va ejecutando el script.

Que hay que tener en cuenta:

- El nombre de una variable debe empezar por $.
- La variable debe empezar por una letra o un '_'.
- No pueden empezar por un número.
- El nombre de una variable solo puede contener "A-Z", "0-9" o "_".
- SON CASE-SENSITIVE.

Tipos de variable ->

string (text values)
int (enteros)
float (decimales)
bool (true o false)
array (múltiples valores)
object (guardar como objetos)
null (variable vacía)
resource (enlaces externos)
mixed (cualquier valor)

Para obtener el valor de estas, utilizaremos var_dump().

Dentro de las variables tenemos las:

- local
  - Una variable declarada dentro de la función es una variable local, lo que hace que solo sea accesible desde esta.

```php
function myTest() {
  $x = 5; // local scope
  echo "Variable x inside function is: $x";
}
myTest();

// using x outside the function will not work
echo "Variable x outside function is: $x";
```

- static
  - Normalmente, las funciones después de ser ejecutadas, borran las variables, en este caso cuando declaramos una static, siempre tendrá el último valor añadido y no se borrara.

```php
static $x = 0; // static scope
```

- Global
  - Una variable declarada fuera de una función es una variable global, la cual no se puede acceder directamente dentro de la función.

```php
$x = 5; // global scope

function myTest() {
  // using x inside this function will not work
  echo "Variable x inside function is: $x";
}
myTest();

echo "Variable x outside function is: $x";
```

  - En caso de que queramos acceder a ellas, tendríamos que añadir la keyword global dentro de la función y llamarlas, suena confuso, pero aquí dejo una imagen:

```php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
```

  - También se puede acceder a ellas de la siguiente forma, haciendo esto podemos modificarlas desde cualquier función a nuestro parecer:

```php
$GLOBALS['y']
```

ECHO y PRINTS

La diferencia entre los echo y los prints, es que echo es un simple método que ejecuta un acción sin devolver nada, lo que hace que sea imposible de utilizar en expresiones lógicas o introducirlo en variables. Mientras que el print es un método que sí se puede utilizar en ternarias, expresiones lógicas y meter dentro de variables:

```php
<?php
    // ESTO FUNCIONA:
    // Asigna 1 a la variable $res y muestra "Hola" en pantalla
    $res = print "Hola";

    // ESTO DA ERROR FATAL:
    // No puedes asignar void a una variable
    // $res = echo "Hola";

    // Uso en expresión (raro pero posible):
    ($edad > 18) ? print "Es mayor" : print "Es menor";
?>
```

Formas mejores de concatenar con los echo y prints:

```php
$nombre = "Francisco";

// Con ECHO
echo "Hola $nombre, bienvenido."; // Salida: Hola Francisco, bienvenido.

// Con PRINT
print "Hola $nombre, bienvenido.";
```

Ejemplo de uso en aplicativo real en cada uno:

```php
echo "<h2>$txt1</h2>";

print "<h2>PHP is Fun!</h2>";
```

Strings y sus funciones

Acción	Java	PHP	Nota Importante
Longitud	`str.length()`	`strlen($str)`	
Posición	`str.indexOf("x")`	`strpos($str, "x")`	Ojo: devuelve `false` si no lo encuentra (no -1).
Cortar	`str.substring(0, 5)`	`substr($str, 0, 5)`	El 2º parámetro es la **longitud**, no el índice final.
Reemplazar	`str.replace("a", "b")`	`str_replace("a", "b", $str)`	El orden es: busca, reemplaza, dónde.
Minúsculas	`str.toLowerCase()`	`strtolower($str)`	
Mayúsculas	`str.toUpperCase()`	`strtoupper($str)`	
Limpiar	`str.trim()`	`trim($str)`	Elimina espacios al inicio y final.
Split	`str.split(",")`	`explode(",", $str)`	Convierte String en Array.
Join	`String.join(",", arr)`	`implode(",", $arr)`	Convierte Array en String.

Operadores

Operador	Nombre	Ejemplo	Resultado	JAVA
+	Adición	$x + $y	Suma de $x y $y	Igual que Java.
-	Sustracción	$x - $y	Diferencia de $x y $y	Igual que Java.
*	Multiplicación	$x * $y	Producto de $x y $y	Igual que Java.
/	División	$x / $y	Cociente de $x y $y	OJO! Devuelve float si no es exacto. No trunca como el int/int de Java.
%	Módulo	$x % $y	Resto de $x dividido por $y	Igual que Java (para enteros).
**	Exponenciación	$x ** $y	$x elevado a la potencia $y	En Java usas Math.pow(), aquí es un operador nativo.

Operador	Nombre	Ejemplo	Resultado	JAVA
==	Igual (Loose)	$x == $y	true si el valor es igual (aunque el tipo sea distinto).	⚠️ PELIGRO: "5" == 5 es true. PHP convierte el texto a número. En Java esto daría error de compilación.
===	Idéntico (Strict)	$x === $y	true si el valor es igual Y el tipo es el mismo.	✅ USAR SIEMPRE: Este es el equivalente lógico a comparar de forma segura. "5" === 5 es false.
!=	No igual	$x != $y	true si el valor no es igual.	Igual que ==, hace conversión de tipos.
<>	No igual	$x <> $y	true si el valor no es igual.	Alias de !=. Arcaico (herencia de SQL/BASIC). No se suele usar.
!==	No idéntico	$x !== $y	true si el valor no es igual O el tipo no es igual.	Es el opuesto estricto de ===.
>	Mayor que	$x > $y	true si $x es mayor que $y.	Igual que Java.
<	Menor que	$x < $y	true si $x es menor que $y.	Igual que Java.
>=	Mayor o igual	$x >= $y	true si $x es mayor o igual que $y.	Igual que Java.

<=	Menor o igual	$x <= $y	true si $x es menor o igual que $y.	Igual que Java.

Condicionales

Al igual que los operadores prácticamente iguales que java, tanto el if, elseif, else, switch.

Bucles

Tenemos 4 tipos de bucles pero obviamos el “do while”

- while:

```
while (condition) {
  // code to be executed repeatedly as long as condition is true
}
```

- for:

```
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
```

- foreach:

```
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
```

Arrays

En php tenemos tres tipos de arrays:

- Arrays indexados.(Index numérico)

```php
$cars = array("Volvo", "BMW", "Toyota");
```

- Arrays asociados (Index nominal)
  -
```php
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);
```
  -
```php
foreach ($car as $x => $y) {
  echo "$x: $y <br>";
}
```
  Muestra TODO

- Arrays multidimensionales (contienen uno o más arrays)
  -
```php
$cars = array (
  array("Volvo", 22, 18),
  array("BMW", 15, 13),
  array("Saab", 5, 2),
  array("Land Rover", 17, 15)
);
```
  -
```php
foreach ($cars as $row) {
  echo "<tr>";
  foreach ($row as $cell) {
    echo "<td>" . $cell . "</td>";
  }
  echo "</tr>";
}
```
  -
```php
echo "</table>";
```

Superglobals:

Estas son HashMaps que php llena con información del entorno o del servidor, que son visibles desde cualquier lugar de tu código.

Superglobal	Para qué sirve en tu App de Facturas?
`$_GET`	Para capturar datos de la URL. Ej: `ver_factura.php?id=5`. PHP guarda ese `5` en `$_GET['id']`.

$_POST	Para recibir los datos de tu formulario de "Crear factura". Todo lo que el usuario escriba en los <input> viaja aquí de forma invisible.
$_SERVER	Información técnica. La usarás para preguntar: if ($_SERVER['REQUEST_METHOD'] == 'POST'). Es decir, "¿Me están enviando datos para guardar o solo visitando la página?".

Superglobal	Descripción	Consejo / Nota
$_REQUEST	Mezcla "sucia" que contiene todo lo que hay en $_GET, $_POST y $_COOKIE.	EVITAR. Es inseguro y poco claro. Sé explícito: si esperas un POST, usa $_POST.
$_SESSION	Mantiene información del usuario (persistencia) mientras navega entre páginas.	Vital en producción. Para tu prueba, úsala si decides hacer un sistema de login simple.
$_FILES	Contiene los datos de archivos binarios subidos al servidor (PDFs, Imágenes).	Requiere que tu formulario HTML tenga enctype="multipart/form-data".

$_COOKIE	Array asociativo con las cookies guardadas en el navegador del usuario.	Útil para recordar preferencias (como "modo oscuro") sin iniciar sesión.
$_ENV	Variables del entorno del Sistema Operativo donde corre el servidor.	Contiene cosas como el PATH o credenciales del servidor.
$GLOBALS	La "madre" de todas. Contiene referencias a todas las variables globales del script.	Rara vez la usarás directamente. PHP la usa internamente.

PHP Forms

$_GET y $_POST.
Estas dos superglobales se utilizan para recoger información de un formulario.
Ambas funcionan de forma similar, pero el $_GET es utilizado para tratar información menos sensible y que no nos importa que sea vista por todos, ya que esta se pasa vía URL y se muestra en esta. Mientras que el $_POST lo envía a través del HTTP Post method, lo cual hace que esta información no sea visible para todos.

Ejemplo de $_POST

- Formulario HTML

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

- Cuerpo PHP (welome.php)

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Ejemplo de $_GET

- Formulario HTML:

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

- Cuerpo PHP (welcome_get.php):

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

Estos son solo los ejemplos, lo recomendado y práctico es hacerlo todo en un mismo archivo:

```
PHP
// 1. LÓGICA PHP AL PRINCIPIO
$mensaje = "";
$error = "";

// Preguntamos: ¿El usuario acaba de pulsar el botón Enviar?
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Recibimos datos
    $concepto = $_POST['concepto'];

    // Validamos
    if (empty($concepto)) {
        $error = "¡El concepto es obligatorio!";
    } else {
        // AQUÍ CONECTARÍAS CON MYSQL Y GUARDARÍAS
        // ... código SQL ...
        $mensaje = "Factura guardada correctamente";
    }
}
?>

<!DOCTYPE html>
<html>
<body>
    <?php
        if ($mensaje) echo "<p style='color:green'>$mensaje</p>";
        if ($error) echo "<p style='color:red'>$error</p>";
    ?>

    <form method="post" action="">
        Concepto: <input type="text" name="concepto">
        <input type="submit" value="Guardar">
    </form>
</body>
</html>
```

Cabe tener en cuenta que todas las entradas de texto deben pasar un filtro, es decir, hay que crear una función la cual limpie el texto “sucio” para obtener un mayor nivel de protección e integridad. El hecho de utilizar el $_GET[“name”] sin pasarle un filtro de texto es una práctica errónea, a la hora de un aplicativo real.

Date and Time

date() - Sirve para mostrar la fecha y/o hora local.
time() - Devuelve la hora y fecha exacta actual:
date_default_timezone_set() - Sirve para seleccionar una zona horaria por defecto.
date_default_timezone_get() - Sirve para devolver la zona horaria que tienes establecida.

Formato más utilizado:

```php
// Esto genera: "2025-12-30 15:30:00" (Fecha y hora exacta)
$fecha_hora_mysql = date("Y-m-d H:i:s");
```

PHP Include Files

Comando	Significado	¿Qué pasa si falla?	Comportamiento
require	"Lo necesito obligatoriamente"	FATAL ERROR	El script se muere al instante. Se detiene toda la ejecución.
include	"Inclúyelo si puedes"	WARNING	El script sigue funcionando. Muestra un aviso feo, pero el resto de la página carga.

PHP connect to MySQL

Con las siguientes líneas conectamos con la base de datos para poder manipularla.

- En caso de ser una base de datos ya existente:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

- En caso de no serlo:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

Creación de una tabla:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Insertar valores:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Seleccionar datos:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>"
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Con un where:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Eliminar

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```