

# UD2. Introducción al lenguaje PHP

Desarrollo Web en Entorno Servidor

Estructuras de control  
Operadores  
Arrays  
Funciones

# Estructuras de control (las clásicas)

---

- Condicionales:

- `if`
- `if-else`
- `if-elseif-else`
- `switch`

`if_basico.php`  
`If_else.php`  
`if_elseif.php`  
`switch.php`

- De repetición

- `for`
- `while`
- `do-while`

`bucle_for.php`  
`bucle_while.php`  
`bucle_dowhile.php`  
`bucle_diferencia_while_dowhile.php`

- Instrucciones de ruptura secuencial:

- `continue`
- `break`
- `return`


`continue.php`  
`return.php`  
`break.php`  
`break_anidado.php`  
`break_switch.php`

# Estructuras de control: instrucciones de ruptura

---

- **continue**: pasa a la siguiente iteración.
- **return**: si está en una función, la finaliza. Si está fuera de una función, finaliza el fichero.
- **break**: finaliza el bucle o la ejecución de un `switch`.
  - En PHP se puede indicar de cuántos bucles se quiere salir.

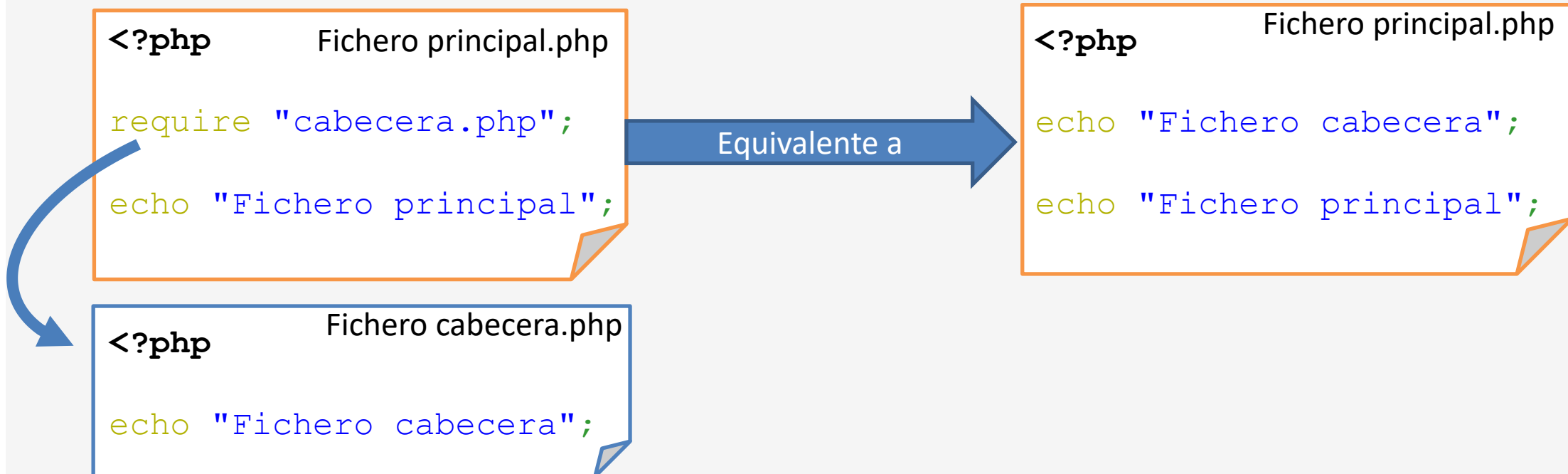
```
for($i=0; $i<14; $i++) {  
    for($j=0; $j<5; $j++) {  
        for($k=0; $k<4; $k++) {  
            break 2;  
        }  
    }  
}
```



# Estructuras de control: inclusión de ficheros

ejerequerido.php  
requerir.php

- Las instrucciones **require** e **include** incluyen el código de otro fichero en el fichero en que se escriben.
- Es decir:



- Las instrucciones **require** e **include** incluyen el código de otro fichero en el fichero.

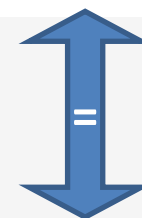
Si no se encuentra el fichero...

- **require** implica obligación → si no existe se genera error fatal y se detiene.
- **include** no implica obligación → genera aviso y se continúa.
- Las versiones **require\_once** e **include\_once** son análogas, pero solo incorporan el código si NO se había hecho antes.

# Estructuras de control: código HTML embebido

- Dentro de los bloques de las estructuras de control, es posible “escapar” el código PHP para inyectar HTML.
- Nos ahorramos los echo.
- A veces es mejor, a veces peor.

```
<?php if($usuario == "admin") { ?>
    <h1>Hola admin!</h1>
<?php } else { ?>
    <h1>No autorizado.</h1>
<?php } ?>
```



```
<?php if($usuario == "admin") {
    echo "<h1>Hola admin!</h1>";
} else {
    echo "<h1>No autorizado.</h1>";
}
```

# Operadores

---

Similares a Java:

- **De comparación:** `==`, `!=`, `>=`, `<=`, `<`, `>`,...
- **Aritméticos:** `%` (módulo), `**` (potencia), `+` (suma), `-` (resta),...
- **Lógicos:** `&&`, `and`, `||`, `or`, `!`, `xor`,...
- **De bit:** `&`, `|`, `>>`, `<<`,...
- **Asignación:** `=`, `&=`, `+=`, `-=`, `*=`, `/=`, ...
- **Otros:** `++$var`, `$var++`, `--$var`, `$var--`, `$var1 . $var2`,...

Lista completa: <https://www.php.net/manual/es/language.operators.php>

# Operadores: igualdad y desigualdad

---

Idéntico.php

Dos formas de comparar:

- Operadores **idéntico** (===) y **no idéntico** (!==).
- Operadores **igual** (==) y **desigual** (!=).



# Operadores: igualdad y desigualdad

---

Idéntico.php

Dos formas de comparar:

- Operadores **idéntico** (===) y **no idéntico** (!==).
- Operadores **igual** (==) y **desigual** (!=).

El operador igual compara solo los valores.

El operador idéntico compara los valores y los tipos.

Son asociativos, más flexibles que otros lenguajes.

De tamaño variable.

Tipos de datos no homogéneos.

Las claves son números o cadenas.

# Arrays unidimensionales

arrays\_sin\_claves.php  
arrays1.php

## Creación:

```
$varA = array(225, "patata", 25);  
$varB = [225, "patata", 25];
```



Sintaxis antigua

## Acceso a un elemento:

```
$varA[1];           // Devuelve "patata"
```

## Añadir o modificar elemento:

```
$varA[6] = "cebolla";  
$varA[0] = 10;  
$varA[] = 12;           // Se añadirá al final
```

# Arrays multidimensionales

---

## Creación:

```
$varA = array( array(3,4), array(5,5), array(1,2) );  
$varB =      [ [3,4],      [5,5],      [1,2] ];
```



Sintaxis antigua

## Acceso a un elemento:

```
$varA[2][0];           // Devuelve 1
```

## Añadir o modificar elemento:

```
$varA[6][6] = "cebolla";  
$varA[0][1] = 10;
```

# Arrays asociativos

---

Por defecto, el índice de un array es 0,1,2,...

Pero podemos indicar el índice que queramos, incluso poner un string.

## Creación:

```
$var1 = array("clave1" => 0, "clave2" => 1);  
$var2 = ["clave1" => 0, "clave2" => 1];
```



Sintaxis antigua

## Acceso a un elemento:

```
$var1["clave1"]; // Devuelve "0"
```

## Añadir o modificar elemento:

```
$var1["palabra"] = "valor";
```

# Arrays: recorrido (I)

---

## Varias opciones:

- Utilizando un bucle `for` limitando con `count()`.

```
$mi_array = [10, 11, 12];  
for($i = 0; $i < count($mi_array); $i++) {  
    echo $mi_array[$i] . "<br>";  
}
```

- Utilizando un `foreach` y sacando el valor

```
$mi_array = [10, 11, 12];  
foreach($mi_array as $elemento_del_array) {  
    echo "El valor es $elemento_del_array <br>";  
}
```

# Arrays: recorrido (II)

---

## **Varias opciones:**

- Utilizando un `foreach` y sacando el valor y la clave:

```
$mi_array = ["Luis" => 10, "Ana" => 11, "Leo" => 12];  
foreach($mi_array as $alumno => $nota) {  
    echo "La nota de $alumno es $nota <br>";  
}
```

# Arrays: operadores

arrays\_union.php

Operador		Descripción
<code>\$a1 === \$a2</code>	Idéntico	Tienen las mismas claves y mismos valores (en el mismo orden y tipo)
<code>\$a1 == \$a2</code>	Igual	Tienen las claves y valores iguales (aunque sean de diferentes tipos)
<code>\$a1 !== \$a2</code>	No idéntico	El opuesto de idéntico
<code>\$a1 != \$a2</code>	No igual	El opuesto de igual
<code>\$a1 + \$a2</code>	Unión	Devuelve un array con los elementos de ambos



# Arrays: especiales de PHP

---

En cada página que se ejecute:

- El módulo de PHP me pone a disposición ciertas variables especiales.
- Con información sobre el servidor, el cliente, la petición, etc.
- La mayoría visibles cualquier punto del código (son **globales**).

NOMBRE	DESCRIPCIÓN
\$GLOBALS	Variables globales definidas en la aplicación
\$_SERVER	Información sobre el servidor
\$_GET	Parámetros enviados con el método GET (en la URL)
\$_POST	Parámetros enviados con el método POST (formularios)
\$_FILES	Ficheros subidos al servidor
\$_COOKIE	Cookies enviadas por el cliente
\$_SESSION	Información de sesión
\$_REQUEST	Contiene la información de \$_GET, \$_POST y \$_COOKIE
\$_ENV	Variables de entorno

# Arrays: el array \$\_GET

---

- Si el usuario escribe en el navegador...

`http://URL?parametro1=valor1&parametro2=valor2`

- Entonces, en el servidor podemos recuperar lo que nos envía:

```
<?php
    echo $_GET["parametro1"]; // Devuelve "valor1"
?>
```

- Pero ojo, tenemos que comprobar por seguridad que existe con `isset`.

Trozo de código que se puede ejecutar cuando se le llama por su nombre:

- Puede devolver un valor (o ninguno).
- Puede recibir varios argumentos (o ninguno).
  - Opcionalmente se pueden indicar valores por defecto para ellos (ver ejemplo).

## **Definición:**

```
function nombre($var1, $var2) {  
    ...  
    return $res;  
}
```

## **Llamada:**

```
nombre($var1, $var2);
```

# Funciones: paso de argumentos

---

[duplicar.php](#)

Los argumentos de una función se pasan por **copia**.

Para pasarlos por **referencia** hace falta usar el operador &

Los argumentos de una función se pasan por **copia**.

Es decir, se pasa una copia del valor, pero **NO** la variable original.

Para pasarlos por **referencia** hace falta usar el operador &

Es decir, se pasa a la función un alias a la variable original.

Es otro mecanismo para devolver variables.

# Funciones predefinidas

## FUNCIONES DE VARIABLES

isset(\$var)	TRUE si la variable está inicializada y no es NULL
is_null(\$var)	TRUE si la variable es NULL
empty(\$var)	TRUE si la variable no está inicializada o su valor es FALSE
is_int(\$var), is_float(\$var), is_bool(\$var), is_array(\$var)	Para comprobar el tipo de dato de \$var
intval(\$var), floatval(\$var), boolval(\$var), strval(\$var)	Para obtener el valor de \$var como otro tipo de dato

**OJO:** PHP trae multitud de funciones de serie, consulta siempre la documentación.

## FUNCIONES DE CADENAS

strlen(\$cad)	Devuelve la longitud de \$cad
explode(\$cad, \$token)	Parte una cadena utilizando \$tok como separador. Devuelve un array de cadenas
implode(\$token, \$array)	Crea una cadena larga a partir de un array de cadenas, entre cadena y cadena se introduce \$token
strcmp(\$cad1, \$cad2)	Compara las dos cadenas. Devuelve 0 si son iguales, -1 si \$cad1 es menor y 1 si \$cad1 es mayor
strtolower(\$cad), strtoupper(\$cad)	Devuelve \$cad en mayúsculas o minúsculas, respectivamente
str(\$cad1, \$cad2)	Busca la primera ocurrencia de \$cad2 en \$cad1. Si no aparece devuelve FALSE, si aparece devuelve \$cad1 desde donde comienza la ocurrencia.

