

UD3 - Programación baseada en linguaxes de marcas con código embebido



Características da linguaxe PHP (I)



Índice

3.1 Introdución

3.2 Condicionais

3.3 Bucles

3.4 Arrays

3.5 Funcións para tipos de datas e cadeas

3.6 Variables superglobais

3.7 Formularios web

3.8 Inclusión e redireccionamento de páxinas

3.9 Programando con algo de xeito





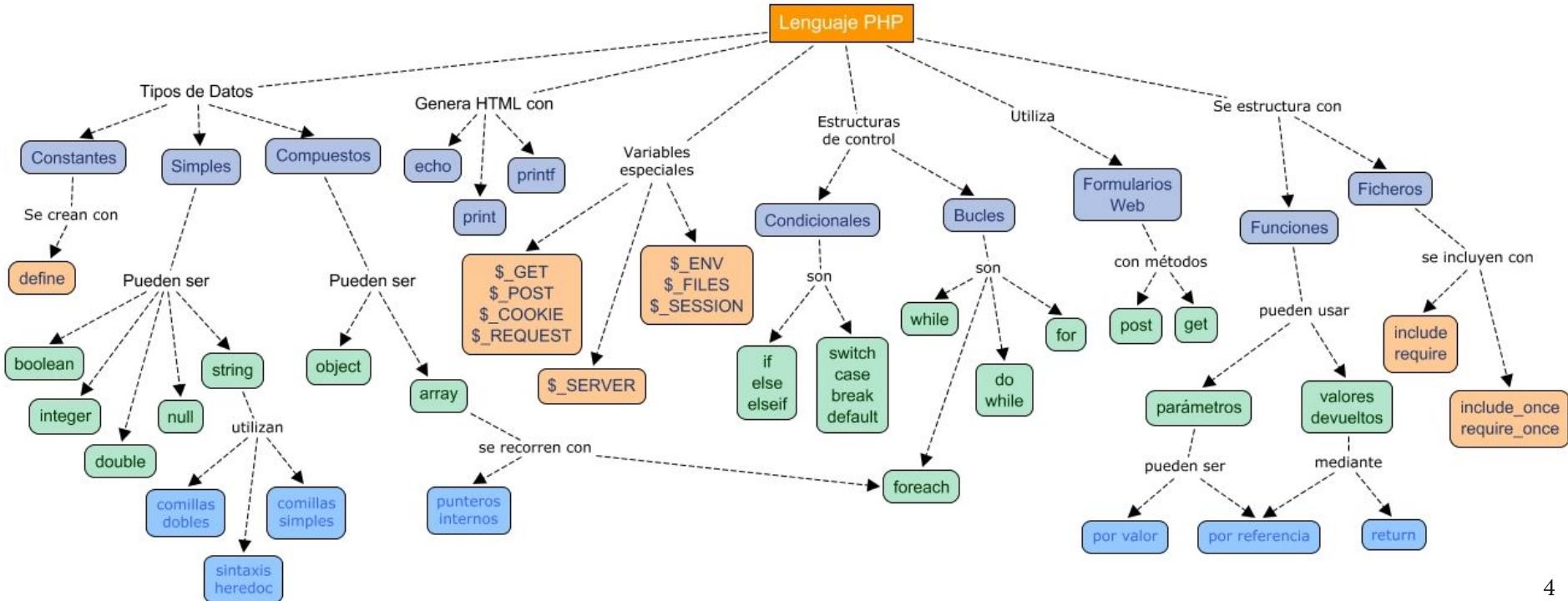
1

Introducción



3.1 Introducción

- Debido ás moitas similitudes coas linguaxes C e Java a curva de aprendizaxe de PHP faise moi dourada.
- Comezaremos polas características básicas da linguaxe: condicionais, bucles e estruturas compostas (Arrays).



2

Condicionais



3.2 Condicionais

- En PHP os guións constrúense en base a sentenzas. Empregando as chaves {}, pódense agrupar as sentenzas en **bloques** ou conjuntos de código que se comportan como se fosen unha única sentenza.
- As **sentenzas condicionais** definen as condicións sobre as que debe executarse unha sentenza ou bloco.

Sentenza if

- A sentenza if permite definir unha expresión para executar ou non o conxunto de sentenzas que vai a continuación.
- Se a expresión se avalía a **true / verdadeiro / 1** a sentenza execútase, e se se avalía a **false / falso / 0**, non se executará.

The screenshot shows a code editor window titled "index.php". The code contains the following PHP script:

```
$x = 20;  
if ($x > 10) {  
    Se a condición é certa  
    echo '$x é maior que 10';  
}  
Executa
```

The code is annotated with green underlines and text: "Se a condición é certa" is placed above the opening brace of the if block, and "Executa" is placed next to the closing brace. The output window below shows the result of the execution: "\$x é maior que 10".



3.2 Condicionais

index.php x

```
$x = 20;  
if ($x > 30) {  
    Se a condición é falsa... Ignora este código  
    echo '$x é maior que 30'; ←|  
}
```

● ● ● Preview x

Como a condición é falsa non se
executa e non se amosa nada

index.php x

```
if (true) {  
    Sempre executa o código dentro do if  
    echo 'A condición é true';  
}  
  
// Saída: A condición é true
```



3.2 Condicionais

Else

A cláusula `else` permite inserir código alternativo cando a condición é falsa.

index.php x

```
$x = 20;  
if ($x > 30) {  
    false  
    echo '$x é maior que 30';  
}  
if ($x > 10) {  
    true  
    echo '$x é maior que 10';  
}
```

← Execútase
este código

index.php x

```
$x = 20;  
if ($x == 30) {  
    false  
    echo '$x é 30';  
} else {  
    echo '$x non é 30';  
}  
  
// Saída: $x non é 30
```



3.2 Condicionais

Elseif

Coa cláusula `elseif` pódense engadir máis condicións.

```
index.php x  
$x = 20;  
if ($x < 30) {  
    echo '$x é menor que 30';  
} elseif ($x >= 20) {  
    echo '$x é maior que 20 ou $x é igual a 20';  
} else {  
    echo '$x é menor que 20';  
}  
// Saída: $x é menor de 30
```

Execútase este código
ao ser certa a condición

Aínda que a condición é certa
non se executa porque xa
se foi pola outra rama

Condicións múltiples

As condicións van encadeadas e poden combinarse con `else` para facer ramas similares a unha estrutura de árbore.

```
index.php x  
$x = 20;  
if ($x > 10 && $x < 30) {  
    echo '$x é maior que 10 e menor que 30';  
}  
if ($x < 10 || $x > 30){  
    echo '$x é menor que 10 e maior que 30';  
}  
// Saída: $x é maior que 10 e menor que 30
```

And

Or

¿Diferenza entre `elseif` e `else if`? → Ningunha



3.2 Condicionais

Negación de condicións e uso de chaves

The screenshot shows a code editor window titled "index.php". The code is as follows:

```
$x = 20;
if (!( $x == 30 )) {
    echo '$x non é igual a 30';
}
// Saída: $x non é igual a 30
```

Annotations in Spanish are present:

- "Non é igual a" and "(inverte a lóxica da condición)" are written next to the logical NOT operator `!`.
- "Devolve true" is written below the condition `(\$x == 30)`.

Cando, como sucede no exemplo, a sentenza `if` / `elseif` ou `else` actúe sobre unha única sentenza, non será necesario empregar chaves `{}`. Habería que usar **chaves** para formar un **conjunto de sentenzas** sempre que queiras que o condicional actúe sobre máis dunha sentenza, aínda que habendo unha única sentenza tamén poden empregarse, co cal pode facerse sempre como boa práctica.



3.2 Condicionais

Switch

A sentenza switch é similar a enlazar varias sentenzas if comparando unha mesma variable con diferentes valores. Cada valor vai nunha sentenza case. Cando se encontra unha coincidencia, comezan a executarse as sentenzas seguintes ata que acaba o bloque switch, ou ata que se encontra unha sentenza break. Se non existe coincidencia co valor de ningún case, execútanse as sentenzas do bloque default, no caso de que exista.

Control de fluxo empregando sentenzas if

```
index.php x

if ($coin == 0) {
    echo 'Cara';
} elseif ($coin == 1) {
    echo 'Cruz';
} else {
    echo 'Error';
}
```

Control de fluxo empregando a sentenza switch

```
index.php x

switch ($coin) {
    case 0:
        echo 'Cara';
        break;
    case 1:
        echo 'Cruz';
        break;
    default:
        echo 'Error';
        break;
}
```



3.2 Condicionais

Uso da sentenza **break**

index.php

```
// Cando $coin == 0
switch ($coin) {
    case 0:
        echo 'Cara'; // Executado
    [---] Non hai BREAK!
    case 1:
        echo 'Cruz'; // Executado
        break;           // Saída: CaraCruz
    default:
        echo 'Error';
        break;
}
```

No caso de que non haxa
break o seguinte case
execútase áinda que non se
cumpra a condición

index.php

```
// Cando $coin == 1
switch ($coin) {
    case 0:
        echo 'Cara';
        // break;
    case 1:
        echo 'Cruz'; // Executado
        break;
    default:
        echo 'Error';
        break;
}
```

break: o programa sae
do switch cando lle toca
// Saída: Cruz



3.2 Condicionais

Agrupar código nunha sentenza switch

```
<?php
// Código que amosa o número de días dun mes
$mes = 10;
switch ($mes) {
    case 2:
        echo 'O mes inserido ten 28 días'; break;
    case 1:
    case 4:
    case 6:
    case 9:
    case 11:
        echo 'O mes inserido ten 30 días'; break;
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        echo 'O mes inserido ten 31 días'; break;
    default:
        echo 'O mes inserido non é válido';
}
?>
```



3.2 Condicionais

Operador ternario

Modo simplificado para condicións: `condición ? valorTrue : valorFalse`

```
● ● ●  
<?php  
$formatoHora = 24;  
$hora = 5;  
$strHora = ($formatoHora == 24) ? $hora+12 : $hora;  
echo "Son as $strHora";  
?>
```

```
● ● ●  
// Exemplo 1: Asignación de valor a unha variable:  
$idade = 20;  
$estado = ($idade >= 18) ? "Maior de idade" : "Menor de idade";  
  
// Exemplo 2: Impresión dun valor:  
$admin = true;  
echo ($admin) ? "Benvido, Administrador" : "Benvido, Usuario";
```

Operador Null coalescing

O operador `??` permite comprobar se unha variable é nula e darlle un valor ou deixala se o ten. Operación moi habitual.

```
● ● ●  
<?php  
$nome = $_GET['nome'] ?? "(baleiro")  
?>
```

```
● ● ●  
<?php  
echo $nomecompleto ?? $nome ?? $apelidos ?? "Debe inserir un nome";  
?>
```

3

Buckles



3.3 Bucles

Bucles

Os bucles empréganse para facer algo repetidamente. Por exemplo, se queremos amosar por pantalla os números do 1 ao 100 consecutivamente. Empregando un bucle pódese reducir a lonxitude dos bloques de código eliminando código repetitivo, reducíndoo a unhas poucas liñas.

Imprimindo os números do 1 ao 100 con echo

```
index.php x  
  
echo 1;  
echo 2;  
echo 3;  
:  
echo 100;
```

```
<?php  
for ($cont=0, $cont<500, $cont++)  
{  
    echo "Non lanzarei avions de papel en clase";  
}  
?>  
  
ANÍMO (0-1)
```



Empregando un bucle

```
index.php x  
  
for ($i = 1; $i <= 100; $i++) {  
    echo $i;  
}
```



3.3 Bucles

Bucle for

Pódese iterar ou repetir código empregando o bucle for. O funcionamento consiste en asignar a unha variable un valor inicial. Despois diso actualízase e execútase repetidamente ata que a condición do bucle xa non se cumpre.





3.3 Bucles

Bucle while

O bucle while permite repetir código igual que o bucle for, con pequenas diferenzas. Cando a condición está especificada o código do bucle repítese mentres sexa `true` ou verdadeiro, áñda que neste caso as variable contador non o fai automaticamente senón que hai que especificalo manualmente.

Exemplo de bucle While

```
index.php ×  
$i = 1; ① Inicializa a variable  
while ($i <= 100) { ② Comproba a condición  
    echo $i; ③ Repite o código  
    $i++; ④ Actualiza a variable  
}
```

Variante do ... While

Este bucle ten unha variante para facer que se execute sempre coma mínimo unha vez o código:



```
<?php  
do {  
    // bloque-de-código  
} while($condicion);  
?>
```

Pódense aniñar calquera dos bucles anteriores en varios niveis. Tamén pódense usar as sentenzas `break`, para saír do bucle, e `continue`, para omitir a execución das sentenzas restantes e volver á comprobación da expresión respectivamente.

Nota: Non se aconsella o uso destas dúas sentenzas (`break` e `continue`) en bucles, para cumplir cos principios da programación estruturada.



3.3 Bucles

Bucles aniñados

Un bucle aniñado é un bucle que se encontra dentro do bloque de sentenzas doutro bucle. Os bucles poden ter calquera nivel de aniñamento (un bucle dentro doutro bucle dentro dun terceiro, etc.).

Nos bucles aniñados é importante empregar **variables de control** distintas, para non obter resultados inesperados. Aínda que os nomes das variables deben ser significativos nestes bucles acostúmanse a empregar as variables **\$i**, **\$j**, **\$k** e así sucesivamente.

Por exemplo, crear unha táboa dun número de filas e columnas especificado previamente con bucles aniñados:

Táboa				
1	2	3	4	
1	1-1	1-2	1-3	1-4
2	2-1	2-2	2-3	2-4
3	3-1	3-2	3-3	3-4

```
<?php
$columnas = 4;
$filas = 3;
?>

<table border="1">
<caption>Táboa</caption>
<tbody>
<tr> <!-- Abre a primeira fila (Cabeceira) -->
<th></th> <!-- Cela baleira -->

<?php
// Bucle 1. Execútase tantas veces como columnas teña a táboa.
for ($j = 1; $j <= $columnas; $j++) {
print " <th>" . $j . "</th>";
}
print " </tr>"; // Pecha a primeira fila

// Bucle 2. Xera o resto de filas da táboa.
for ($i = 1; $i <= $filas; $i++) {
print " <tr>"; // Abre unha nova fila
// Crea a primeira cela <th> de cada fila (con número)
print " <th>" . $i . "</th>";

// Bucle 3. Execútase tantas veces como columnas teña a táboa
for ($j = 1; $j <= $columnas; $j++) {
print " <td>" . $i-$j . "</td>";
// Crea o resto de celas <td> de cada fila (con números)
}
print " </tr>"; // Pecha cada fila
}
?>
</tbody>
</table>
```



3.3 Bucles

Bucles infinitos

No caso de esquecer actualizar o valor da variable empregada na condición ao final do bucle, isto crea un bucle infinito onde a condición sempre sería `true` ou verdadeiro e por tanto continuaría executándose. Os bucles infinitos bloquean o programa e impiden que continúe a súa secuencia e non habería máis remedio que terminalo a man. Hai que asegurarse de que a condición é `false` ou falsa nalgún momento durante a execución do bucle, é dicir, que vaia rematar nalgún momento.

Exemplo de bucle Infinito

```
index.php ×

$ i = 1;

while ($i <= 100) {
    echo $i;
}

O bucle repetírase eternamente xa que
a condición será sempre true
```

Os bucles infinitos son un erro de programación, áinda que en determinadas ocasións se incorporan sentenzas de saída (`break` ou `continue`) como resposta a eventos externos son solucións aceptables, de novo serían usos pouco recomendables.

4

Arrays



3.4 Arrays

Tipos de datos compostos

- Un tipo de datos composto é aquel que permite almacenar más dun valor na mesma referencia. nunha variable temos unha referencia e un único valor. En PHP temos **dous tipos de datos compostos: arrays e obxectos**. Neles podemos almacenar múltiples valores a través da referencia do nome do obxecto ou do array.
- Nun **array** pódese almacenar unha colección de múltiples valores. Viría sendo unha caixa dividida con separadores onde cada un deles almacenará un dato. Cada compartimento levará unha **clave asociada** (0, 1, 2, ...) que será o **índice do array**.
- Estas claves poden ser:
 - **Numéricas**. Unha secuencia de enteros: 0, 1, 2...
 - **Asociativas**. Cadeas de caracteres que identifican os espazos ("AB", "AC", "AD", ...).

Variable

'Patricio'



Variable
\$nome1

'Calamardo'



Variable
\$nome2

'Bob'

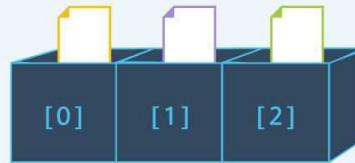


Variable
\$nome3

Manexan datos de xeito independente

Array

'Patricio' 'Calamardo' 'Bob'



Variable
\$nomes

Manexan todo o grupo de datos conjuntamente



3.4 Arrays

Arrays

- A sintaxe básica **para declarar ou crear un array** é indicando o nome do mesmo seguido duna serie de valores.

```
$nomeArray = array(valor1, valor2, ..., valorN);
```

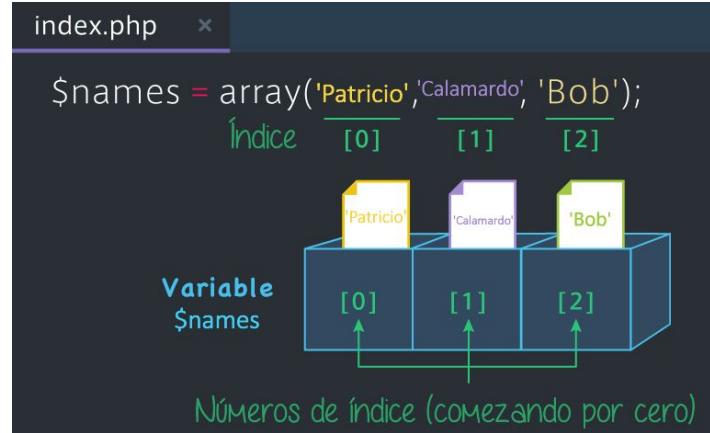
```
$nomeArray = [valor1, valor2, ..., valorN];
```

```
$nomeArray = []; // Array baleiro
```

```
$nomeArray = array(); // Array baleiro
```

- Nos numéricos, **por defecto** asígnase un **índice de 0** ao primeiro elemento.
- Para **recuperar un determinado valor** fariámolo indicando a posición entre corchetes [] dese valor no array:

```
$nomeArray [0];  
// Devolve o primeiro elemento do array
```



Accedendo aos elementos nun array

```
index.php
```

```
echo $names[0];
```

```
// Saída: Patricio
```

↑ Especificase un
número de índice

```
echo $names[1];
```

```
// Saída: Calamardo
```

↑ Especificase un
número de índice



3.4 Arrays

Engadir, actualizar e eliminar elementos

```
index.php  x

$names = array( 'Patricio', 'Calamardo', 'Bob' );
$names[] = 'Arenita';   ← Engade un elemento
                  ao final do array
echo $names[3]; // Saída: Arenita
$names[1] = 'Tyrion'; ← Actualiza o elemento
                      do índice 1
echo $names[1]; // Saída: Tyrion
```

- Para **eliminar elementos** recorreríamos á función **unset()**, por exemplo: `unset ($names[1]);`



3.4 Arrays

- Ao **eliminar un elemento** dun array, **pérdese o seu índice** tamén. No caso dos numéricos, deixarían de ser secuenciais.
- Coa **función array_values()** poderíamos volver a asignar un índice secuencial (se o precisamos).

```
$array = [0 => "cero", 1 => "un", 2 => "dous", 3 => "tres"];
unset($array[1]);

print_r($array); // Array ([0] => cero [2] => dous [3] => tres)

// Reindexar
$array = array_values($array);

print_r($array); // Array ([0] => cero [1] => dous [2] => tres)
```



3.4 Arrays

Arrays associativos

Nos arrays associativos accédese aos elementos a través de cadeas de texto denominadas claves, que son empregadas en lugar dos índices numéricos. Pódese especificar a clave para un valor empregando =>, por exemplo: `nomeArray = array('clave' => 'valor1', ...);`

Arrays associativos



index.php

```
$user = array(  
    Nome de variable  
    'nome' => 'Tyrion'  
    'idade' => 27  
    'casa' => 'Lannister'  
)
```

The code shows the definition of an associative array \$user. It consists of three key-value pairs: 'nome' is associated with the string 'Tyrion'; 'idade' is associated with the integer 27; and 'casa' is associated with the string 'Lannister'. The keys are labeled 'Nome de variable' and the values are labeled 'Valores'. A callout box with the text 'Separados por comas' points to the commas separating the elements in the array definition.



3.4 Arrays

Arrays multidimensionais (I)

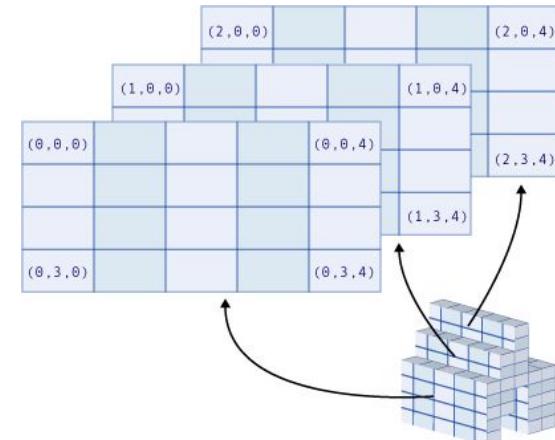
- Ata agora estivemos vendo arrays **unidimensionais**, é dicir un índice simple, que referencia os elementos nunha fila.

(0)	(1)	(2)	(3)	(4)
-----	-----	-----	-----	-----

- Se gardamos en cada compartimento un novo array temos un array de dous dimensóns ou **bidimensional**. Para facer referencia aos seus elementos indicarse fila e columna, e terá similitude cunha táboa de datos.

(0,0)				(0,4)
(1,0)				
(3,0)				(3,4)

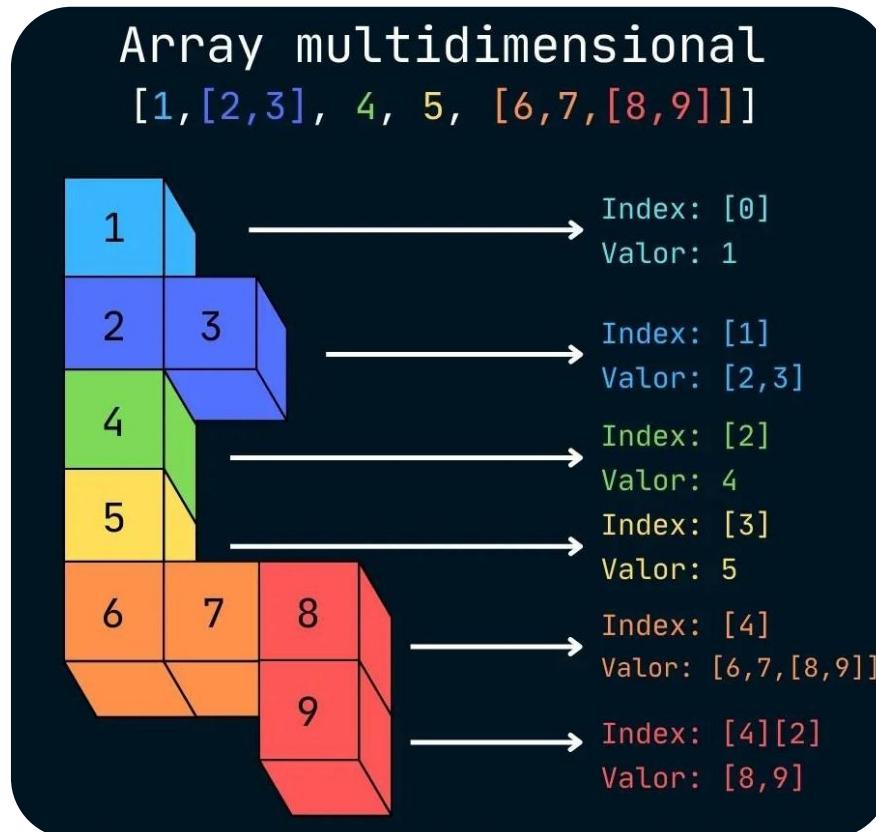
- Podemos expandir o array en calquera número de dimensóns ou **multidimensionais** sen máis que ir engadindo novos arrays en vez de valores. En PHP non é necesario indicar o tamaño antes de crear o array (igual que coa declaración doutros elementos). Simplemente se van engadindo novos elementos.





3.4 Arrays

Arrays multidimensionais (II)





3.4 Arrays

Arrays multidimensionais (III)



```
<?php
$personaxes = array(
    array("Gandalf", "Frodo", "Aragorn"),
    array("Tyrion", "Jon Snow", "Cersei"),
    array("Fafhrd", "Ratonero gris")
);

echo "Personaxes do primeiro libro: ";
echo $personaxes[0][0] . ", ";
echo $personaxes[0][1] . ", ";
echo $personaxes[0][2];
?>
```

Saída: Personaxes do primeiro libro: Gandalf, Frodo, Aragorn



```
<?php
$cidades = [
    'Galicia'  => ['Fene', 'Ferrol', 'A Coruña'],
    'Asturias'  => ['Llanes', 'Ribadesella', 'Arriondas'],
    'Cantabria' => ['Comillas', 'Laredo', 'Cabárceno']
];

?>
<h4>Array multidimensional de índice asociativo</h4>
<pre>
<?php
print_r($cidades); // Amosa a estrutura e o contido do array de xeito lexible

// Amósanse os elementos do array a partir da variable $índice
$índice = 'Galicia';
echo "\$índice = $índice<br> Elemento [0] -> " . $cidades[$índice][0];
echo "\$índice = $índice<br> Elemento [1] -> " . $cidades[$índice][1];
echo "\$índice = $índice<br> Elemento [2] -> " . $cidades[$índice][2];
?>
</pre>
```

Suxestión: A función **print_r()** amosa o contido do array de xeito lexible, sendo moi útil á hora de depurar ou corrixir erros.



3.4 Arrays

Percorrer arrays (I)

Para percorrer os elementos dun array existe un bucle específico (en PHP): foreach. Emprega unha variable temporal para recoller en cada iteración o valor para ese índice.

Accedendo aos valores nun array con foreach

Colle o valor do array
e o asigna a variable





3.4 Arrays

Percorrer arrays (II)

index.php x

```
$data = array('Tokyo', 'London', 'New York');
foreach ($data as $valor) {
    echo $value;
}

// Saída : Tokyo London New York
```

index.php x

```
foreach ($array as $valor) {
    Array associativo
    // Execútase outro código
}

foreach ($array as $clave => $valor) {
    Array associativo
    // Execútase outro código
}
```

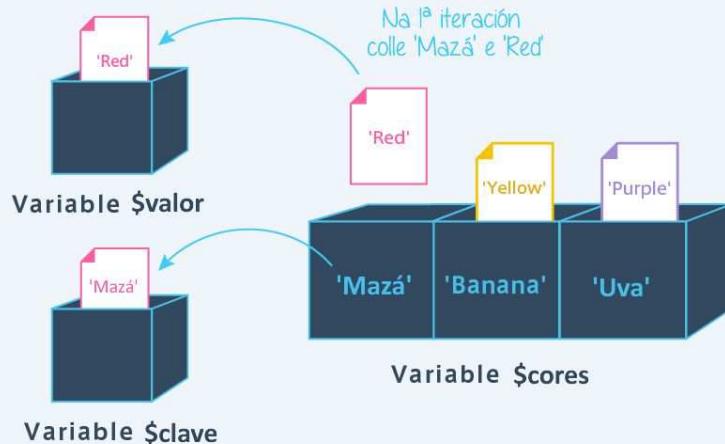
Pode omitirse



3.4 Arrays

Percorrer arrays (III)

Accedendo aos valores nun array asociativo



index.php

```
$cores = array(
```

```
    'Mazá' => 'Red',  
    'Banana' => 'Yellow',  
    'Uva' => 'Purple'  
)
```

```
foreach ( $cores as $clave => $valor ) {  
    echo $clave. ":" . $valor ;  
}
```

// Saída: Mazá:Red Banana:Yellow Uva:Purple

En cada iteración
recole un novo par
clave-valor



3.4 Arrays

Strings ou cadeas de texto como arrays

As **cadeas de texto ou Strings poden tratarse como arrays** nos que cada letra iría nunha posición seguindo o índice, sendo 0 o índice da primeira letra, 1 o da segunda e así ata chegar ao último carácter que tería como índice o tamaño da cadea menos un.

```
● ● ●  
<?php  
  
$cadea = "ABCDEFG";  
  
echo "Posición1: " . $cadea[0];  
echo " || Posición4: " . $cadea[3];  
  
// Saída: Posicion1: A || Posicion4: D  
  
?>
```

```
● ● ●  
  
$cadea = "ABCDEFG";  
  
// Obter a lonxitude da cadea  
$lonxitude = strlen($cadea);  
  
// Percorrer a cadea e amosar cada carácter  
for ($i = 0; $i < $lonxitude; $i++) {  
    echo $cadea[$i] . "<br>";  
}
```

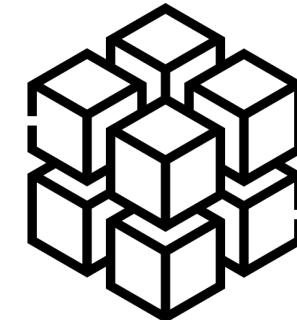


3.4 Arrays

Funcións de arrays (I)

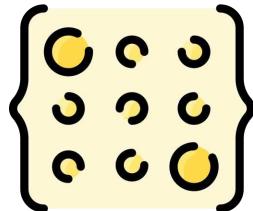
Tal e como acontecía con outros tipos de datos como Strings ou Datas, os arrays dispoñen dun conxunto de funcións predefinidas nas librerías que se instalan por defecto con PHP. Estas funcións permiten realizar as operacións más básicas ou habituais, como por exemplo:

- **Percorrer arrays.** Funcións como: each(), next(), prev(), reset(), end(), etc...
- **Funcións de busca.**
 - in_array(). Busca un valor nun array.
- **Traballo con arrays.**
 - count(). Indica o número de elementos dun array.
 - array_push(). Insire novos elementos no array.
 - shuffle(). Mestura de xeito aleatorio os elementos dun array.
 - array_diff(). Compara dous arrays e devolve os elementos que non coincidan.
 - array_merge(). Combina dous arrays xuntando os seus elementos.
 - range(). Crea un array de elementos dentro do rango especificado.





3.4 Arrays



Funcións de arrays (II)

- **Ordenación.**
 - sort(), rsort(). Ordenan os elementos do array.
 - array_values(). Reindexa un array, útil cando se fan unsets para recrear os índices. Devolve o array modificado.
- **Conversión de String a array e viceversa.**
 - explode(). Converte un String a partir dun separador , ; ou un espazo en branco no String correspondente. A función implode() realizaría a operación inversa, converter o array nun String dividido por un separador.

```
<?php
$strDatos = "Tarta de Chocolate, Tarta de Mazá, Membrillo";
// aplicamos a función explode() aos datos separados por comas da variable $strDatos
$postres = explode(", ", $strDatos);
echo $postres[0]; // Amosa o primeiro elemento "Tarta de chocolate"
echo $postres[1]; // Amosa o segundo elemento "Tarta de mazá"
echo $postres[2]; // Amosa o terceiro elemento "Membrillo"
?>
```

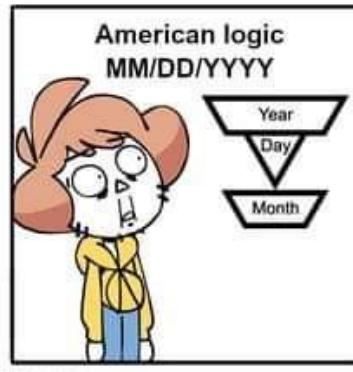
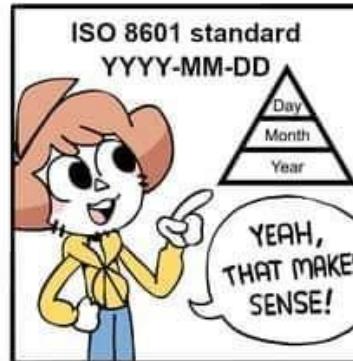
5

Funcións para tipos de datos e cadeas



3.5 Funcións para tipos de datos e cadeas

- PHP non ten un tipo de datos específico para datas e horas, como si fan as bases de datos cos tipos `date` e `datetime` por exemplo.
- A información de data e hora **gárdase coma un número enteiro** e para facilitar o traballo con datas e horas disporemos dunha ampla colección de funcións.
- **Timestamp** → A data represéntase como `timestamp`, unha marca temporal de tipo enteiro que é o valor devolto pola función `time()`. Representa o momento actual medido coma segundos dende a época Unix (1 de xaneiro de 1970 00:00:00 GMT), por exemplo: **1663576503** (algo máis dun billón e medio de segundos).
- **Conversión** → Ao traballar con outros sistemas como as bases de datos (MYSQL usaría AAAA-MM-DD) e ter que adaptar as datas segundo as necesidades dunha aplicación verémonos obrigados a estalas convertendo continuamente. Así mesmo haberá diferenzas entre o noso formato dd/mm/aaaa e o americano mm/dd/aaaa, e co ISO, que é o que terán os componentes HTML (AAAA-MM-DD).





3.5 Funcións para tipos de datos e cadeas

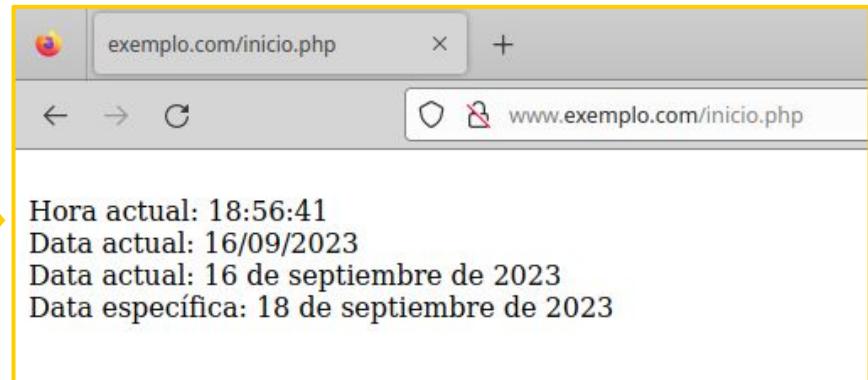
Funcións para Datas e Horas (I)

- A función `date()` permite obter unha cadea de texto a partires dunha data e hora con diferentes formatos.
- A función recibe o formato e o enteiro coa data e hora (`timestamp`) a formatar e devolve a cadea de texto formatada.
- Se non se indica a data en formato enteiro collerá a data e hora actuais.

```
date(string $format, int $timestamp = time()): string
```

Para aplicar o formato coa función `date()` hai que empregar unha serie de caracteres, e opcionalmente pasarlle un `timestamp` coa data se non queremos que colla a actual por exemplo:

```
<?php
$horaActual = date("H:i:s");
echo "Hora actual: " . $horaActual;
$dataActual = date("d/m/Y");
echo "Data actual: " . $dataActual;
$dataActual2 = date("j F Y");
echo "Data actual: " . $dataActual2;
$timestamp = strtotime("2023-09-18");
$dataEspecifica = date("d F Y", $timestamp);
?>
```





3.5 Funcións para tipos de datos e cadeas

Funcións para Datas e Horas (II)

Para que un sistema poida proporcionar información sobre a data e hora correcta, debes indicarlle a túa zona horaria, senón o máis probable é que aparezca unha advertencia ou *warning*. Para establecer a **zona horaria** en España peninsular habería que indicar:

```
date_default_timezone_set('Europe/Madrid');
```

Por outra banda no tocante aos **textos en cada idioma**, a función setlocale() permite definir o idioma local, para español sería o código: 'es_ES'. Posteriormente poderase imprimir a data segundo ese localismo coa función strftime().

```
setlocale(LC_TIME, 'es_ES');
```

```
setlocale(LC_TIME, 'gl_ES');
```

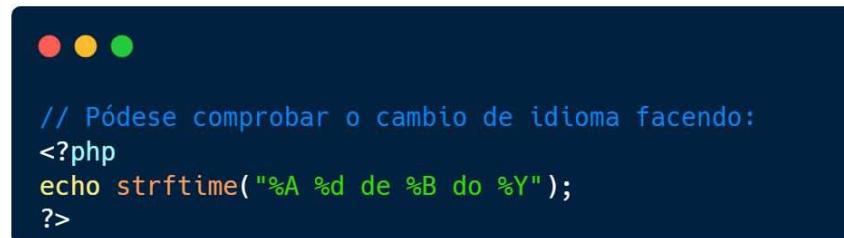
```
setlocale(LC_TIME, 'es_ES.UTF-8');
```

```
setlocale(LC_TIME, 'gl_ES.UTF-8');
```

Nota: Para isto ese localismo ten que estar instalado no servidor, en Linux/Xubuntu:

```
$ sudo locale-gen es_ES
```

```
$ sudo locale-gen es_ES.UTF-8
```



```
// Pódese comprobar o cambio de idioma facendo:  
<?php  
echo strftime("%A %d de %B do %Y");  
?>
```



3.5 Funcións para tipos de datos e cadeas

Funcións para Datas e Horas (III)

Referencia rápida de funcións para traballar con datas:

- time(). Devolve a data actual como **timestamp**.
- date(). Da formato á data actual ou a un **timestamp** que se lle pase como parámetro.
- mktime(). Obtén o **timestamp** dunha data.
- getdate(). Obtén un array con toda a información do **timestamp** que se lle pase como parámetro.
- strtotime(). Converte unha cadea de texto cunha data en formato inglés a **timestamp**.
- strftime(). Da formato a un **timestamp** para convertelo a cadea de texto.

Nota: PHP dispón do obxecto DateTime para facilitar o traballo con datas, que veremos máis adiante dentro da unidade de POO.



```
<?php
$semanaSeguinte = time() + (7 * 24 * 60 * 60);
// 7 días; 24 horas; 60 minutos; 60 segundos
echo '<br>Agora: ' . date('Y-m-d') . "\n";
echo '<br>Semana Seguinte: ' . date('Y-m-d', $semanaSeguinte);
// ou usando strtotime():
echo '<br>Semana Siguiente: ' . date('Y-m-d', strtotime('+1 week'));
?>
```



```
<?php
echo strtotime("now"), "\n";
echo strtotime("10 September 2000"), "\n";
echo strtotime("+1 day"), "\n";
echo strtotime("+1 week"), "\n";
echo strtotime("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime("next Thursday"), "\n";
echo strtotime("last Monday"), "\n";
?>
```



3.5 Funcións para tipos de datos e cadeas

Funcións para Datas e Horas (IV)

Exemplo todo nun, partindo de dúas datas MYSQL/HTML (AAAA-MM-DD):

```
<?php

// Dúas datas no formato YYYY-MM-DD
$data1 = '2024-09-25';
$data2 = '2024-09-10';

// Converter ambas datas en timestamp (segundos dende o 1 de xaneiro de 1970)
$timestamp1 = strtotime($data1);
$timestamp2 = strtotime($data2);

// Amosar as datas completas formateadas para nós
echo "Data 1: " . strftime('%A, %d de %B de %Y', $timestamp1) . "<br>";
// Mércores, 25 de Setembro de 2024
echo "Data 2: " . strftime('%A, %d de %B de %Y', $timestamp2) . "<br><br>";
// Martes, 10 de Setembro de 2024

// Calcular a diferenza en segundos entre ambas
$diferencia_segundos = $timestamp1 - $timestamp2;

// Converter a diferenza en días (un día = 86400 segundos)
$diferencia_dias = $diferencia_segundos / 86400;

// Amosar a diferenza en días
echo "A diferenza entre $data1 e $data2 é de " . abs($diferencia_dias) . " días.";
?>
```

Axuda de strftime() en [php.net](https://www.php.net/manual/en/function.strftime.php)

format	Descripción	Ejemplo de valores devueltos
Día	---	---
%a	Una representación textual abreviada del día	dom hasta sáb
%A	Una representación textual completa del día	domingo hasta sábado
%d	El día del mes con dos dígitos (con ceros iniciales)	01 a 31
%e	El día del mes, con un espacio precediendo a los dígitos simples. No está implementado como está descrito en Windows. Véase más abajo para más información.	1 a 31
%j	Día del año, tres dígitos con ceros iniciales	001 a 366
%u	Representación numérica del día de la semana del ISO-8601	1 (para lunes) hasta 7 (para domingo)
%w	Representación numérica del día de la semana	0 (para domingo) hasta 6 (para sábado)



3.5 Funcións para tipos de datos e cadeas

Funcións para cadeas de texto (I)

Algunhas funcións de uso habitual serían:

- `strlen($cadea)`. Devolve o tamaño dun string.
- `substr($cadea, $posición, $offset)`. Devolve a parte do string indicada entre dúas posicións.
- `str_replace($busca, $reemplaza, $cadea)`. Substitúe unha cadea por outra nun string.
- `str_pad(...)`. Enche un string con caracteres ata o tamaño indicado.
- `strpos($cadea, $busca)`. Devolve a posición dunha cadea dentro de outra.
- `strip_tags($cadea)`. Función de seguridade que elimina etiquetas HTML e PHP dunha cadea.

```
<?php
$text = '<p>Parágrafo de proba.</p><!-- Comment -->
<a href="#fragment">Outro texto</a>';
// Elimina todo (exemplo1)
echo strip_tags($text);
// Permite <p> e <a> (exemplo2)
echo strip_tags($text, '<p><a>');

/* O resultado do exemplo sería:
exemplo1 >>> Parágrafo de proba. Outro texto
exemplo2 >>> <p>Parágrafo de proba.</p> <a
href="#fragment">Outro texto</a>
*/
?>
```



3.5 Funcións para tipos de datos e cadeas

Funcións para cadeas de texto (II)

- [htmlspecialchars\(\)](#). Función de seguridade que converte caracteres especiais a entidades HTML. Especialmente útil como mecanismo de protección fronte ataques XSS para evitar que os datos que achega o usuario sexan interpretados como etiquetas validas en lugar de texto.



```
<?php
$cadeaCodificada = htmlspecialchars("<a href='test'>Proba</a>", ENT_QUOTES);
echo $cadeaCodificada; // &lt;a href=&#039;test&#039;&gt;Proba&lt;/a&gt;
?>
```

- [trim\(\\$cadea\)](#). Elimina os espazos, tabuladores e outros caracteres a ambos extremos da cadea. Útil para validacións e seguridade.
- [ucwords\(\)](#), [strtolower\(\)](#), [strtoupper\(\)](#), etc. Modificar maiúsculas e minúsculas.
- [Ver más...](#)



6

Variables superglobais



3.6 Variables superglobais

- En PHP dispoñemos de variables internas predefinidas que poden usarse en calquera ámbito (local ou global), polo que reciben o nome de variables **superglobais**.
- Cada unha delas é un array ou matriz cun conxunto de valores, que PHP nos da por defecto.
- Principais variables:
 - **`$_GET`, `$_POST`, `$_COOKIE` e `$_REQUEST`** conteñen as variables que se pasaron ao guión actual utilizando respectivamente os métodos GET (parámetros na URL), HTTP POST (ocultos), Cookies HTTP ou todos os métodos xuntos (Request). Estes métodos veranse más polo miúdo ao traballar con formularios, agás as cookies.
 - **`$_ENV`** contén variables que se poidan ter pasado a PHP dende o contorno en que se está a executar.
 - **`$_FILES`** contén os ficheiros que se poidan ter subido ao servidor co método POST.
 - **`$_SESSION`** contén as variables de sesión disponibles para o guión ou script actual.
 - **`$_SERVER`** contén información sobre o contorno de servidor web e de execución.





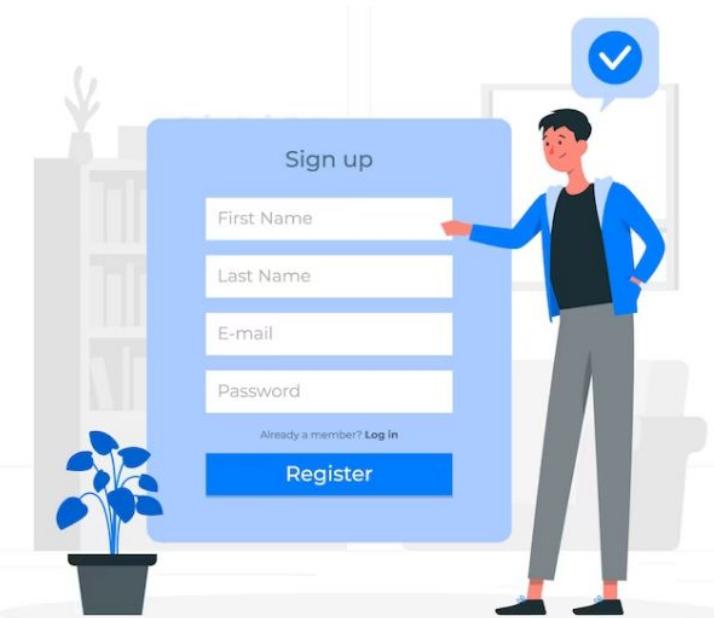
7

Formularios web



3.7 Formularios web

- Nas aplicacíons web o elemento fundamental para **recoller información** do usuario/a serán os **formularios web**.
- Os formularios servirán para **interactuar co usuario** e fazer que os programas se comporten de xeito distinto segundo os **parámetros de entrada**.
- Teñen a particularidade de que se definen empregando **unicamente HTML**. Van no cliente xa que é a quen están dirixidos, pero o seu procesamento terá que realizarse dalgún xeito no servidor para poder executar o programa PHP.
- Os formularios HTML van enmarcados sempre entre as etiquetas **<FORM> </FORM>**.
- Dentro dun formulario inclúense os elementos sobre os que pode actuar o usuario, principalmente usando as etiquetas **<INPUT>**, **<SELECT>**, **<TEXTAREA>**, **<CHECKBOX>**, **<RADIO>**, **<FILE>** e **<BUTTON>**.





3.7 Formularios web

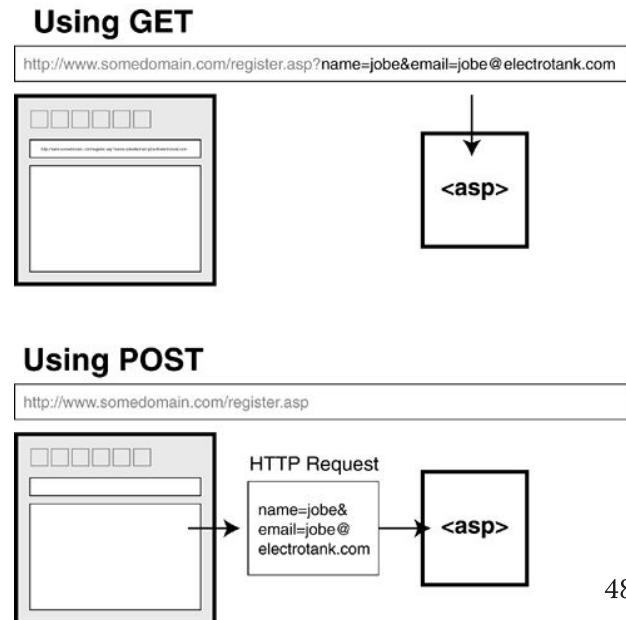
- O atributo **action** do elemento **FORM** indica a páxina á que se lle enviarán os datos do formulario.
- O atributo **method** especifica o método empregado para enviar a información. Este atributo pode ter dous valores:
 - **GET**. Con este método os datos do formulario agréganse á URI empregando un signo de interrogación "?" como separador.
 - **POST**. Con este método os datos inclúense no corpo do formulario e se envían utilizando o protocolo HTTP.

Preparando un formulario

The code in the editor shows:

```
index.php
```

A onde se van
enviar os datos
↓
<form action="url" method="post">
Atributo action Atributo post
// Código do formulario
</form>





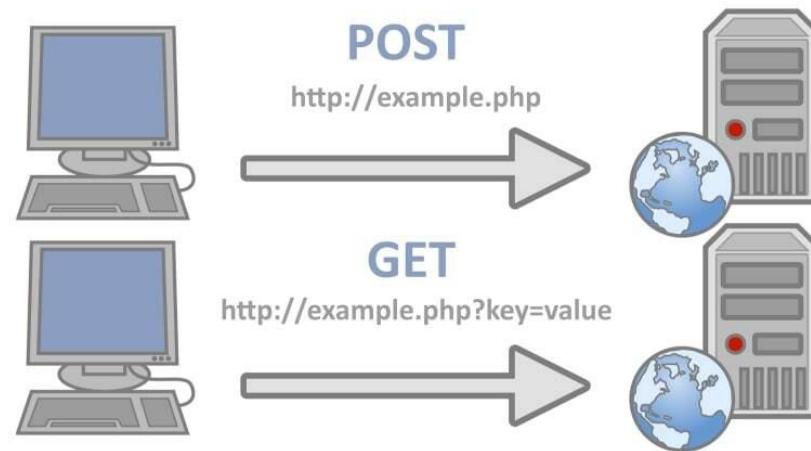
3.7 Formularios web

Método GET (I)

Neste caso os datos son enviados na URL e pasan a páxina indicada polo etiqueta **action** do formulario no seguinte formato:

http://paxina.php?parametro1=valor1¶metro2=valor2&...

Os datos que aparecen despois do signo **?** son os parámetros, e serán pares parámetro a valor, coa forma parámetro seguido do símbolo **=** e do valor e cada parámetro separado polo símbolo **&**.





3.7 Formularios web

Método GET (II)

Desvantaxes

- **Calquera usuario pode ver os valores** das variables **na barra do navegador**, polo que dito método pode resultar inseguro, non se debe enviar información sensible como contrasinais.
- **O usuario pode cambiar o valor** de ditas variables de forma cómoda simplemente escribindo uns novos no navegador polo que pode suceder que o sitio amose ou faga algo diferente do permitido. Por POST sería más complicado, áñada que tamén posible capturando a petición HTTP ou clonando o formulario.
- O **tamaño** dos parámetros que se poden pasar nunha URL está **limitado** (dependendo do navegador pode oscilar entre 2KB-8KB, é dicir 2000-8000 caracteres).
- Emprega o **método URL encoding** polo que determinados caracteres dos valores como os espazos en branco e os caracteres non alfanuméricos poden sufrir transformacións.
- **Non se poden enviar datos binarios** como arquivos, imaxes, etc.

Vantaxes

- A través da URL **poden enviarse datos** dunha páxina a outra **sen necesidade de emplegar formularios**.



3.7 Formularios web

Método GET (III)

Recuperar os datos enviados e xerar ligazóns con datos

- As variables que se envían desta forma poden ser referenciadas en PHP a través do **array asociativo global \$_GET[]**.
- Faríase da forma: **\$_GET['Parametro1']**.



```
<html>
<body>
<form action="ud2.1.php" method="get">
    Nome: <input type="text" name="nome"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" value="Enviar"><br>
</form>
Ola <?php isset($_GET["nome"]) ? print $_GET["nome"] : ""; ?>!<br>
Xa teño o teu email: <?php isset($_GET["email"]) ? print $_GET["email"] : ""; ?>
</body>
</html>
```

Este formulario sinxelo recolle os valores ao facer *submit* sobre si mesmo, amosándoos por pantalla. Na URL aparecerá algo como:

ud2.1.php?nome=pepe&email=pepe%40gmail.com

Xa que substitúe a @ pola codificación %40.

Ademais pode empregarse para **enviar datos en ligazóns** creándoas a partires de PHP, por exemplo:

```
<a href="datos.php?id=<?php echo $id; ?>&nome=<?php echo $nome; ?>">Ver datos</a>
```



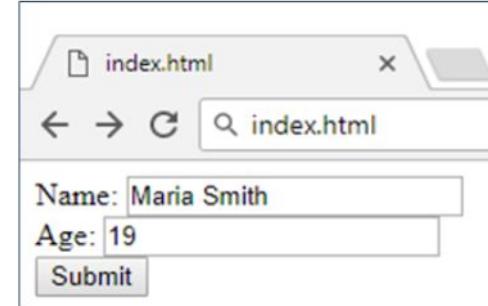
3.7 Formularios web

Método POST (I)

- No método HTTP POST os datos van dentro da petición HTTP e polo tanto non son visibles para o usuario na URL como acontecía co método GET. Ademais, permite o envío de datos binarios (arquivos ou imaxes).
- Límite de datos a enviar, maior que co método GET pero tamén ten un límite, por defecto será o do PHP, cun tamaño de 8Mb.
- Para modificar o límite de envío, pódese:
 - Modificar as directivas `post_max_size` e `upload_max_filesize` no `php.ini` e reiniciando o Apache.
 - Engadir un arquivo `.htaccess` no cartafol onde está a páxina php engadindo unha directiva `php_value_post_max_size`, que redefine dende a raíz do documento o límite de tamaño de arquivos.

```
<form method="post">
  Name: <input type="text" name="name"/> <br/>
  Age: <input type="text" name="age"/> <br/>
  <input type="submit" />
</form>

POST /index.html HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
          URL-encoded form data
name=Maria+Smith&age=19
```





3.7 Formularios web

Método POST (II)

- As variables que se envían desta forma poden ser referenciadas en PHP a través do **array asociativo global \$_POST[]**. Faríase da forma: **\$_POST['Parametro1']**.

```
● ● ●  
  
<html>  
<body>  
  <form action="ud2.1php" method="post">  
    Nome: <input type="text" name="nome"><br>  
    Email: <input type="text" name="email"><br>  
    <input type="submit" value="Enviar"><br>  
  </form>  
  Ola <?php isset($_POST["nome"]) ? print $_POST["nome"] : ""; ?>!<br>  
  Xa teño o teu email: <?php isset($_POST["email"]) ? print $_POST["email"] : ""; ?>  
  </body>  
</html>
```

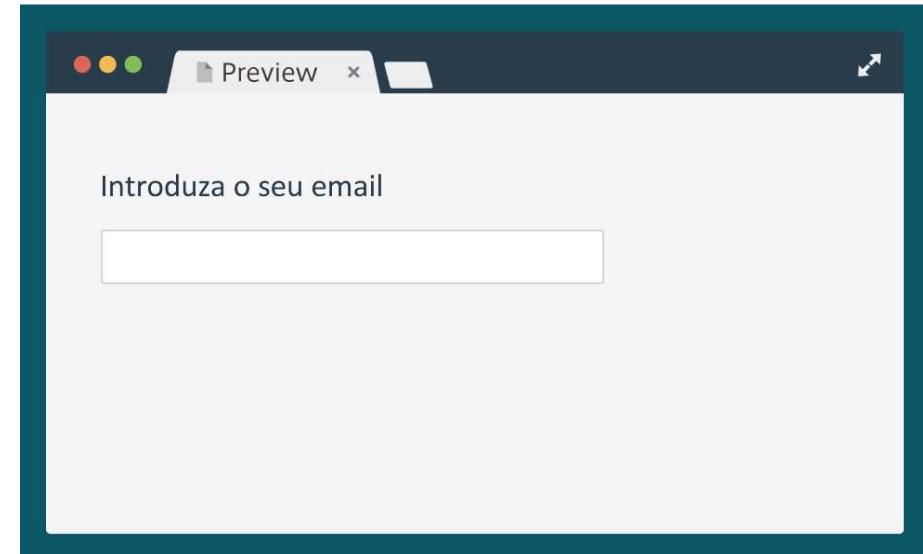
- Nota: Non se recomenda empregar **\$_REQUEST** por claridade no código e problemas de rendemento e seguridade.



3.7 Formularios web

Elemento INPUT

```
index.php  ✎  
  
<form action="enviar.php" method="post">  
    Introduza o seu email  
    <input type="text" name="email">  
        Etiqueta para este input  
</form>
```



Nota: Ademais de indicar o nome coa etiqueta `name` é recomendable facelo tamén coa etiqueta `id`.

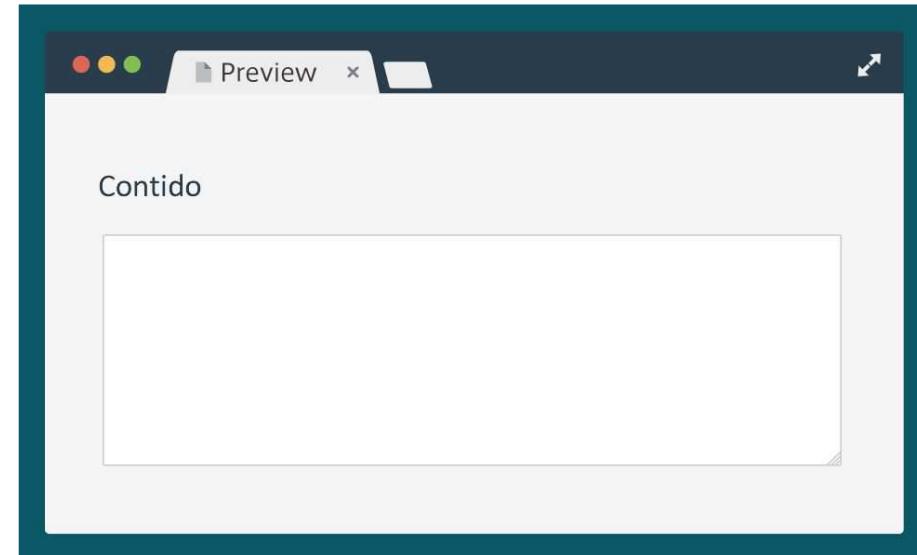


3.7 Formularios web

Elemento TEXTAREA

index.php x

```
<form action="enviar.php" method="post">  
    Contido  
    <textarea name="content"></textarea>  
        Etiqueta para o elemento      Require etiqueta  
                                    de peche  
</form>
```



Nota: Ademais de indicar o nome coa etiqueta `name` é recomendable facelo tamén coa etiqueta `id`.



3.7 Formularios web

Botón ENVIAR

index.php x

```
<form action="enviar.php" method="post">
    URL

    Introduza o seu email

    <input type="text" name="email" value="">

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

Texto que se vai
mostrar no botón

The screenshot shows a 'Preview' window with a dark header bar. Inside, there is a form with the placeholder text 'Introduza o seu email' above an empty input field. Below the input field is a green rectangular button with the word 'Enviar' in white. The overall interface is clean and modern.

Nota: Lembra tamén que un botón de tipo **reset**, limparía todos os datos inseridos no formulario.



3.7 Formularios web

Procesamento da información devolta por un formulario web (I)

- Os valores de cada elemento do formulario envíanse á páxina ou script PHP que se indique no **action**. Unha vez alí haberá que recollelos.
- Se foron enviados polo método **POST** coa variable **\$_POST** e se foron enviados con **GET** empregando a variable **\$_GET**.
- Por outra banda, sexa cal sexa o método de envío tamén se poden recoller empregando a variable **\$_REQUEST**.
- **¿Por que non empregar sempre \$_REQUEST?** Porque pode darnos conflitos ao non poder diferenciar se unha variable se recibe polos dous métodos á vez (a través dun formulario e a URL) e leva o mesmo nome. Tamén empeora o rendemento e a seguridade.

The screenshot shows a browser window titled "Preview" and a code editor window titled "index.php". The browser preview shows two input fields: "Nome" containing "Paco o ninja" and "Email" containing "paco@ninjasunidos.com". The code editor shows the HTML form structure:

```
<form action="enviar.php" ...>
    <input ... name="name">
    ...
    : O nome do atributo é name
    <input ... name="email">
    ...
    : O nome do atributo é email
</form>
```

Annotations with green arrows point from the browser inputs to their corresponding code elements. The annotation for the "Nome" input includes the note "O nome do atributo é name". The annotation for the "Email" input includes the note "O nome do atributo é email".

Nota: O máis **recomendable** para os formularios sería tratar todo con **POST**, xa que quedan ocultos ao usuario/a e se evita o problema da dobre entrada.



3.7 Formularios web

Procesamento da información devolta por un formulario web (II)

Preparando un formulario

The screenshot shows a code editor window titled "enviar.php". The code is as follows:

```
Enviar × Especifica o nome do elemento para obter o valor
echo $_POST['name']; // Saída: Paco o ninja
echo $_POST['email']; // Saída: paco@ninjasunidos.com
```

A callout box highlights the line `$_POST['name']`. A green arrow points from the text "Especifica o nome do elemento para obter o valor" to the opening bracket of the array reference. Another callout box highlights the line `array(` and contains the following text:

`$_POST` é un array asociativo

array(
'name' => 'Paco o ninja',	
Clave	Valor
'email' => 'paco@ninjasunidos.com'	
)	

Nota: Pódese substituír `$_POST` por `$_REQUEST` directamente, ou por `$_GET` cambiando o método do formulario.



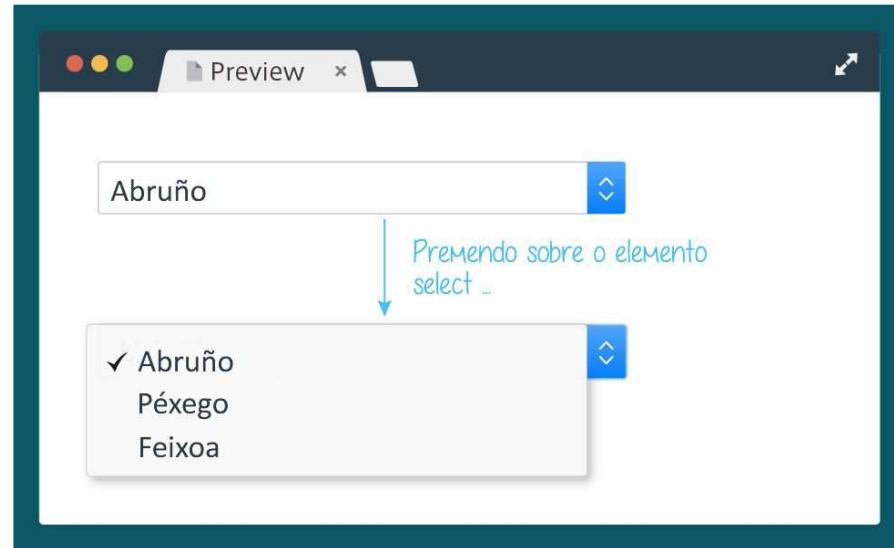
3.7 Formularios web

Elemento SELECT (I)

index.php

```
<select>
    <option> Abruño </option>
    <option> Péxego </option>
    <option> Feixoa </option>
</select>
```

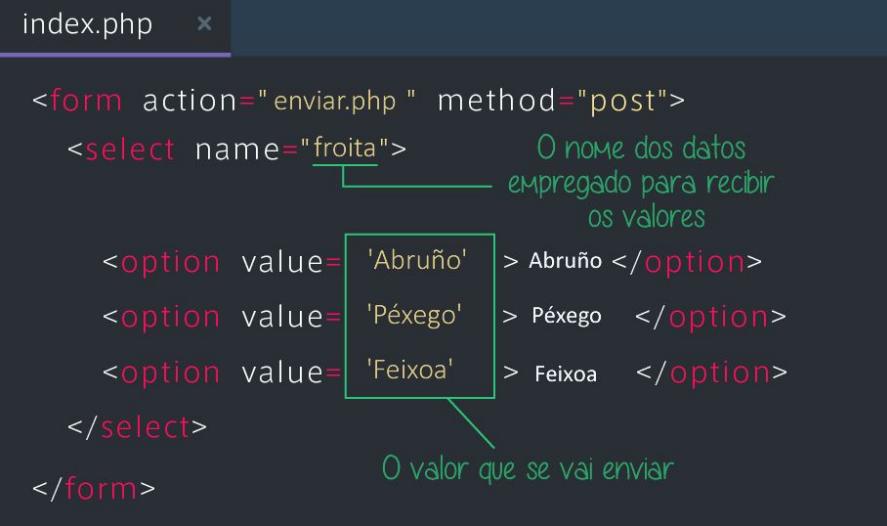
Os valores amósanse nunha lista despregable





3.7 Formularios web

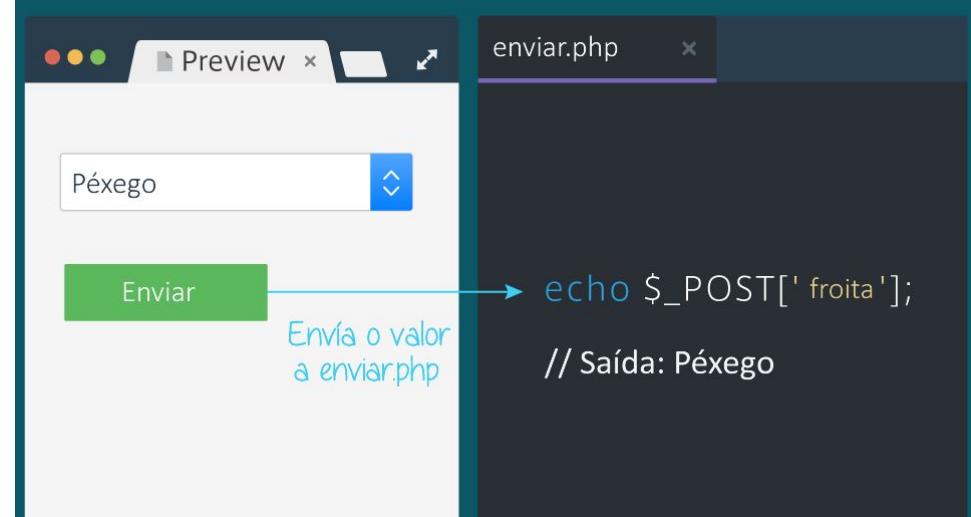
Elemento SELECT (II)



```
index.php x
<form action="enviar.php" method="post">
    <select name="froita">          O nome dos datos
        <option value='Abruño'> Abruño </option>
        <option value='Péxego'> Péxego </option>
        <option value='Feixoa'> Feixoa </option>
    </select>
</form>
```

O nome dos datos empregado para recibir os valores

O valor que se vai enviar



Preview x

Péxego

Enviar

enviar.php x

```
echo $_POST['froita'];
// Saída: Péxego
```

Envía o valor a enviar.php



3.7 Formularios web

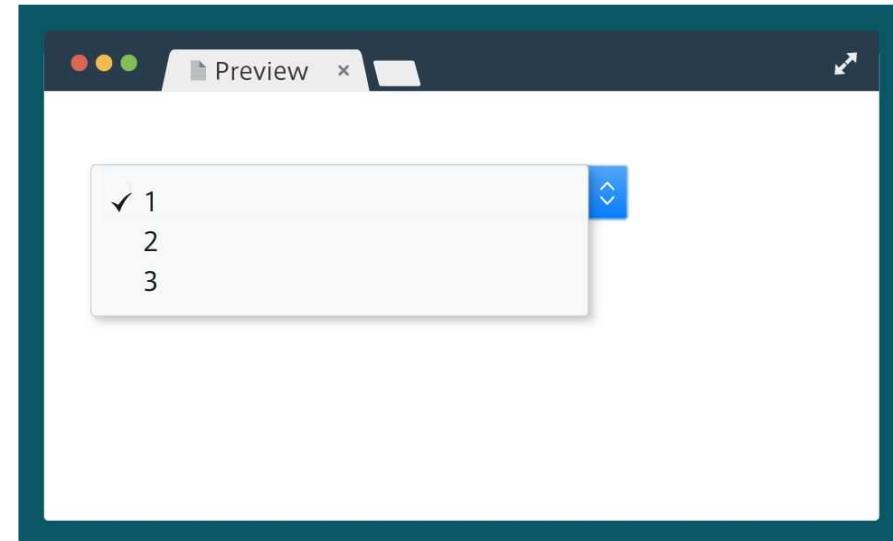
Elemento SELECT (III)

En calquera componente HTML podemos facer substitucións con código PHP para aforrar traballo. Por exemplo:

```
index.php  x

for ($i = 1; $i < 4; $i++) {
    echo "<option value='{$i}'>{$i}</option>";
}
Substitución por variables

— Sería equivalente a facelo só con HTML:
echo "<option value='1'>1</option>";
echo "<option value='2'>2</option>";
echo "<option value='3'>3</option>";
```





3.7 Formularios web

Checkbox

A caixa de verificación ou **checkbox** amósase como unha caixa cadrada que está marcada (marcada) cando está activada. As caixas de verificación úsanse para que un usuario poida seleccionar unha ou máis opcións dun número limitado de opcións.

Podemos empregar un identificador diferente para cada unha, por exemplo:

```
<?php  
// Verifica se a casilla de verificación "vehiculo1" está marcada  
if (isset($_POST['vehiculo1'])) {  
    echo "Tes unha bicicleta.";  
}  
  
// Verifica se a casilla de verificación "vehiculo2" está marcada  
if (isset($_POST['vehiculo2'])) {  
    echo "Tes unha piragua.";  
}  
?>
```

```
<form action="datos.php" method="POST">  
    <input type="checkbox" id="vehiculo1" name="vehiculo1" value="Bici">  
    <label for="vehiculo1">Teño bici</label><br>  
    <input type="checkbox" id="vehiculo2" name="vehiculo2" value="Piragua">  
    <label for="vehiculo2">Teño piragua</label><br>  
    <input type="submit" value="Enviar">  
</form>
```

Teño bici
 Teño piragua



3.7 Formularios web

Array de checkboxes (I)

Tamén se poden emplegar de maneira agrupada como unha lista de caixas definíndoos como un array:

Grupos musicais:

- Los Chikos del Maíz
- Son da rúa
- Dois ke te Crew
- Kase O

Enviar

```
<h1>Grupos musicais:</h1>
<form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
    <input type="checkbox" value="1" name="musica[]" /><label>Los Chikos del Maíz</label><br/>
    <input type="checkbox" value="2" name="musica[]" /><label>Son da rúa</label><br/>
    <input type="checkbox" value="3" name="musica[]" /><label>Dois ke te Crew</label><br/>
    <input type="checkbox" value="4" name="musica[]" /><label>Kase O</label><br/>
    <br><br><br>
    <input type="submit" value="Enviar" name="enviar" id="enviar">
</form>
```

O que recibiremos como `$_POST['musica']` será un array, onde cada elemento marcado se incluirá co valor que temos no `value` e se non o marcamos non se incluirá.



3.7 Formularios web

Array de checkboxes (II)

Podemos traballar con ese elemento como un array, para verificar que caixas foron marcadas, validar se é obligatorio enviar cando menos un, ou calquera outra operación:

```
echo "Validando campos.....<br>";
var_dump($_POST);
if (isset($_POST['enviar'])) {
    if (is_array($_POST['musica'])) {
        $seleccionados = '';
        $numGrupos = count($_POST['musica']);
        $contador = 0;
        foreach ($_POST['musica'] as $clave => $valor) {
            if ($contador != $numGrupos - 1)
                $seleccionados .= $valor.', ';
            else
                $seleccionados .= $valor.'.';
            $contador++;
        }
    }
    else {
        $seleccionados = 'Debes seleccionar cando menos un grupo.';
    }
    echo '<div>Seleccionados: '.$seleccionados.'</div>';
}
```

No exemplo, obtense o número de elementos marcados, e percórrese o array para amosar os seus valores no string `$seleccionados`.



3.7 Formularios web

Array na lista de selección múltiple

Así mesmo, cando empregamos listas de selección múltiple temos que definir tamén como array o nome dese elemento, xa que enviamos máis dun valor (logo o procesaremos como antes):

Idiomas que coñece:

Galego	▲
Inglés	
portugués	▼

```
<form>
    <p>Idiomas que coñece:</p>
    <select name="idioma[]" size="3" multiple="multiple">
        <option selected="selected" value="g">Galego</option>
        <option value="i">Inglés</option>
        <option value="p">Portugués</option>
    </select>
    <p>
        <input type="submit" value="rexistrar">
        <input type="reset" value="borrar">
    </p>
</form>
```



8

Inclusión e redirecciónamento de páxinas



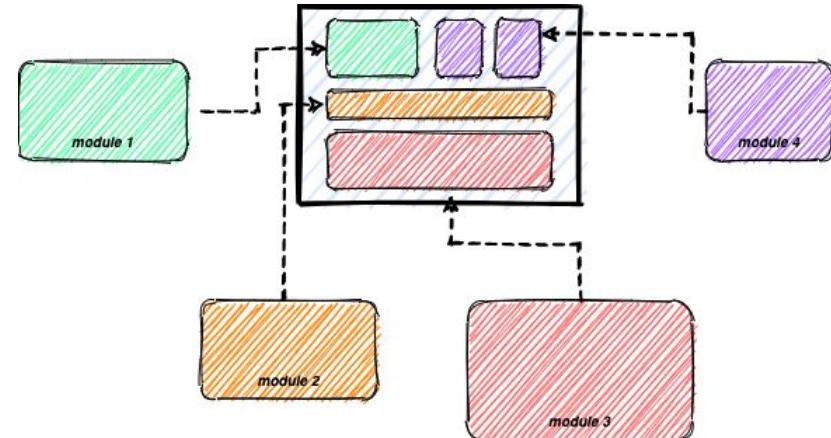
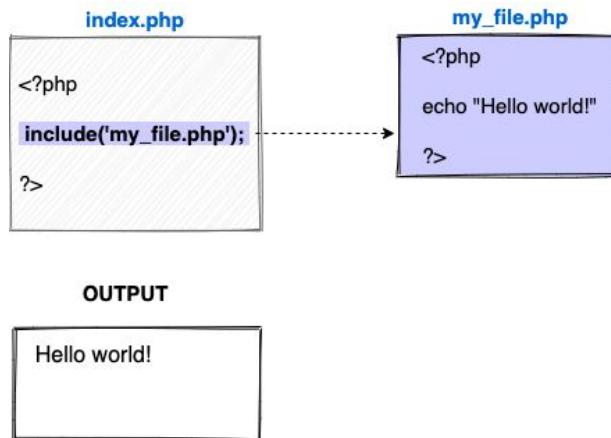
3.8 Inclusión e redirecciónamento de páxinas

Conforme medran os programas resulta útil agrupar o código en diferentes ficheiros, o mesmo que se queremos reutilizar código en ficheiros externos. PHP permite facer referencia a eses ficheiros para incluílos no programa actual. PHP dispón de varias sentenzas para incluír ficheiros no código:

- **include()**. Avalía o contido do ficheiro que se indica e o inclúe como parte do ficheiro actual, no mesmo punto en que se realiza a chamada. A localización do ficheiro pode especificarse utilizando unha ruta absoluta, pero o más usual é cunha ruta relativa. Neste caso, tómase como base a ruta que se especifica na directiva `include_path` do ficheiro `php.ini`. Se non se atopa nesa localización, buscarase tamén no directorio do guión actual, e no directorio de execución.
- **include_once()**. Se por equivocación se inclúe máis dunha vez un mesmo ficheiro, o normal é obter algún tipo de erro (por exemplo, ao repetir unha definición de función). `include_once()` funciona exactamente igual que `include()`, pero só inclúe aqueles ficheiros que ainda non se incluíron no programa. **Evita incluír o mesmo arquivo dúas veces.**
- **require()**. Se o ficheiro que queremos incluír non se atopa, cando usamos `include` da un aviso e continúa a execución do guión. A diferenza máis importante ao usar `require()` é que nese caso, **cando non se pode incluír o ficheiro, se detén a execución do guión.**
- **require_once()**. Asegura a inclusión do ficheiro indicado só unha vez, e xera un erro se non se pode levar a cabo.



3.8 Inclusión e redirecciónamento de páxinas



- **¿Por que non empregar sempre as funcións `_once()`?** → Porque son más pesadas e consumen más recursos.
- Cando se comeza a avaliar o contido do ficheiro externo, abandónase de forma automática o modo PHP e o seu contido se trata en principio como etiquetas HTML. Por este motivo, é necesario delimitar o código PHP que conteña o noso arquivo externo utilizando dentro do mesmo os delimitadores `<?php` e `?>`.
- **Suxestión** → Unha técnica de programación para que o código resulte más claro consiste en emplegar a extensión `.inc.php` para aqueles ficheiros en linguaxe PHP que se crean para ser incluídos dentro doutros, e nunca terían que executarse independentes.



3.8 Inclusión e redirecciónamento de páxinas



```
<!DOCTYPE html>
<html>
<head>
    <title>Páxina de inicio</title>
</head>
<body>
    <!-- Incluímos o menú -->
    <?php include 'menu.php'; ?>

    <h1>Benvido á nosa páxina de inicio</h1>
    <p>Esta é a páxina principal do noso sitio web.</p>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
    <title>Páxina de contacto</title>
</head>
<body>
    <!-- Incluímos o menú -->
    <?php include 'menu.php'; ?>

    <h1>Contacto</h1>
    <form ... >
        </form>
    </body>
</html>
```



```
<?php // Script menu.php ?>
<nav>
    <ul>
        <li><a href="inicio.php">Inicio</a></li>
        <li><a href="acerca.php">Sobre nós</a></li>
        <li><a href="contacto.php">Contacto</a></li>
    </ul>
</nav>
```



3.8 Inclusión e redirecciónamento de páxinas

- É posible **redirixir ao usuario a outra páxina ou script**.
- Se por exemplo unha vez completado un formulario de rexistro se valida correctamente o seguinte paso sería indicarle que o rexistro se realizou e que ten que confirmar un email ou algo similar, amosando outra páxina. A función **header()**, que permite enviar encabezados HTTP pódese empregar para facer este tipo de redireccións:

```
<?php
    header( 'Location: http://www.ninjasunidos.com' );
    exit;
?>
```

- Cómpre facer uso do encabezado **Location**, que será o que provoque que se cargue a nova páxina ou script.
- Vai **sempre acompañada de exit** (tamén serviría **die**) para evitar que se execute calquera outro código interno a partires da redirección. Ademais, estas funcións melloran o rendemento evitando que se envíe o resto do contido da páxina.



3.8 Inclusión e redireccionamento de páxinas

- Debe chamarse antes de amosar HTML por pantalla, aínda que sexa un único espazo en branco ou salto de liña, ou poderían producirse erros.
- É dicir, NON se poden usar echo, print, printf ou etiquetas HTML antes da función header().
- Hai que ter especial coidado cando se inclúen outras páxinas no script para evitar que insiran elas o HTML.



```
<html>
<?php
/* Esto producirá un error. Fíxate no <html> que vai antes da chamada a header() */
header('Location: http://www.ninjasunidos.com/');
exit;
?>
```

- Noutros usos, pode empregarse para lanzar erros personalizados. Por exemplo, un erro 404:



```
header("HTTP/1.1 404 Non atopado");
include("erro404.html"); // Amosa a páxina de erro personalizada
exit;
```



3.8 Inclusión e redirecccionamento de páxinas

- Os scripts PHP normalmente xeran contido dinámico que non debe ser posto na caché polo navegador do cliente ou proxy. Outro uso desta función permite **forzar a desactivación da caché** con:

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT"); // Unha data pasada
?>
```

- Ademais, tamén permite **abrir diálogos de descarga**:

```
<?php
// Imos amosar un PDF
header('Content-Type: application/pdf');

// Darémoslle o nome documento.pdf ao PDF descargado
header('Content-Disposition: attachment; filename="documento.pdf"');

// Documento no servidor
readfile('documento-no-servidor.pdf');
?>
```

9

Programando con algo de xeito



3.9 Programando con algo de xeito

- **Cabeceira do arquivo** → Sempre é importante que todos os archivos `.php` inicien cunha cabeceira específica que indique quien foi o último que modificou ese documento, en que data o fixo, etc. A nivel profesional, cada equipo de desenvolvemento establece se estes campos deben ampliarse con más información para favorecer o traballo en equipo.

```
<?php
/**
 *  @Título: Tarefa 1-1 Comezando con PHP
 *  @versión: 1.0.0 (Irá incrementando en cada cambio)
 *  @autor: Frodo Bolsón, de Bolsón Cerrado.
 *  @data últ. modificación: 20/09/2023
 */
?>
```

- **Comentarios** → **Sempre** hai que comentar o código. Calquera función, calquera detalle que se saia do básico, agrupamentos de seccións do código etc, deben documentarse para axudar a entender o código.
 - Deben ser **significativos e relevantes**, aportar algo a maiores que o propio nome de variables, funcións ou tipos devoltoos.
 - Ollo, os comentarios non deberían contradicir o código xamais, deberían estar **actualizados**. Se facemos un cambio posterior, hai que cambiar tamén o comentario.



3.9 Programando con algo de xeito

- **Facer novos arquivos e cartafoles cando sexa necesario** → Sempre se poden crear novos arquivos e directorios tentando que o nome que se lles dea sexa o máis significativo posible. Por exemplo, para as librerías crear unha carpeta **include**, para a base de datos unha carpeta **/bd** ou **/db**, etc. Crear un arquivo de configuración, un para cada clase, outro para as funcións por categorías, etc.
- **Facer o código doadoo de ler**
 - **Identar o código** con espazos ou tabuladores (o máis recomendable sería só os primeiros), respectando a xerarquía e a profundidade á hora de anidar os diferentes elementos.
 - Establecer un **tamaño máximo de liña** de arredor de 150-200 caracteres.
 - Indicar **espazos** entre os diferentes **operadores**:



```
<?php  
// Non se recomenda, difícil de ler  
$a=0;  
for($i=5;$i<=$j;$i++)  
?>
```



```
<?php  
// Recomendación  
$a = 0;  
for ($i = 5;$i <= $j; $i++)  
?>
```



3.9 Programando con algo de xeito

- Os **nomes de variables** deberían comenzar por minúscula (despois do \$), e a primeira letra da seguinte palabra se a teñen por maiúscula. Estilo CamelCase. Por exemplo: `$importeTotal`
- Á hora de empregar **chaves** facelo sempre colocándoas **en liña**, para saber cal se corresponde con cal.

```
<?php
if ($condicion) {
    for ($i = 0; $i <= 10; $i++) {
        // Bloque de código
    }
}

while ($condicion) {
    funcion();
}
?>
```



Documentación complementaria

- [Manual oficial de PHP.](#)
- [Wiki CIFP Rodolfo Ucha: Sintaxe básica de PHP.](#)
- [Wiki San Clemente: PHP.](#)
- [Funcións para datas e horas en PHP.](#)
- [Vídeotutorial: Manexo de formularios en PHP.](#)





Fin!

¿Algunha pregunta?

Lembra que tamén podes preguntar:

- ◉ Nos foros da aula virtual.
- ◉ Pola mensaxería da aula virtual.