

Manual de iniciación a PHP (II)

MANUAL DE INICIACIÓN A PHP (II)

INDICE

1. ESTRUCTURAS DE CONTROL	2
2. FUNCIONES.....	5
3. MANEJO DE CADENAS.....	7
4. FUNCIONES PARA MANEJO DE FECHAS	10
ANEXO 1. TIPOS DE ERRORES.....	11

1. ESTRUCTURAS DE CONTROL

- Condicionales: **if – elseif – else**

```
<?php
$nota = 5;
if ($nota<5) {
    echo "Vuelve a
intentarlo";
} elseif ($nota==5) {
    echo "Uff";
} else {
    echo "Bien hecho!";
}
?>
```

switch

```
<?php
$genero = 'F';
switch ($genero) {
    case 'M': echo
'Masculino';
                break;
    case 'F': echo
'Femenino';
                break;
    default: echo
'Neutro';
}
?>
```

- Bucles: **while**

```
<?php
$i = 0;
while ($i<10) {
    echo "contador = $i";
    $i++;
}
?>
```

```
<?php
$i = 0;
do {
    echo "contador = $i";
    $i++;
} while ($i<10);
?>
```

- Bucle: **for** (para arrays indexados)

```
<?php
for ($i=0; $i<10, $i++)
{
    echo "contador = $i";
}
?>
```

```
<?php
$países = array ('Italia', 'España',
'Francia');
$longitud = count ($países);
for ($i=0; $i<$longitud; $i++) {
    echo "país[$i] = ".$países[$i] <br>";
}
?>
```

← → ↺ ↻ 🏠 localhost/manual/for_array.php
📁 Favoritos

```
país[0] = Italia
país[1] = España
país[2] = Francia
```

- Bucle: **foreach** (solo para arrays asociativos, matrices y objetos)

Hay dos formas posibles de usar el bucle foreach:

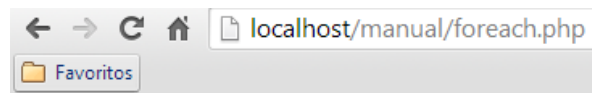
1. `foreach ($nombre_array as $valor) {`
 sentencias; `}`

Este bucle recorre el array indicado en \$nombre_array.

En cada iteración el elemento actual del array se guarda en \$valor y se incrementa el puntero interno del array.

```
<?php
    $países = array ('it'=>'Italia', 'es'=>'España',
'fr'=>'Francia');
    foreach ($países as $valor) {
        echo "$valor <br>";
    }
?>
```

[foreach1.php](#)



Italia España Francia

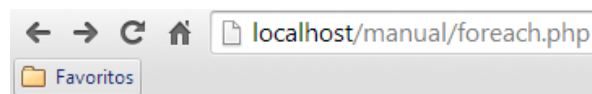
2. `foreach ($nombre_array as $clave=>$valor) {`
 sentencias; `}`

Este bucle recorre el array indicado en \$nombre_array.

En cada iteración el elemento actual del array se guarda en \$valor, la clave se guarda en \$clave y se incrementa el puntero interno del array.

```
<?php
    $dominios = array ('it'=>'Italia', 'es'=>'España',
'fr'=>'Francia');
    foreach ($dominios as $clave=>$valor) {
        echo "dominio[$clave] = ".$valor <br>";
    }
?>
```

[foreach2.php](#)



dominio[it] = Italia
dominio[es] = España
dominio[fr] = Francia

- **require y require_once**

Sirve para insertar en nuestro documento y en la posición exacta donde está [require](#), el código contenido en un archivo externo, antes de ser ejecutado por el servidor.

En caso de no encontrar el archivo especificado, se produce un FATAL ERROR que interrumpe la ejecución. (require_once solo inserta la primera vez aparece)

- **include e include_once**

Sirve para pegar en nuestro documento y en la posición exacta donde está [include](#), el código contenido en un archivo externo, antes de ser ejecutado por el servidor.

En caso de no encontrar el archivo especificado se envía un WARNING pero continúa la ejecución. (include_once solo inserta la primera vez que aparece)

```
<?php
echo    '<a href="/require.php">require</a> --
        <a href="/include.php">include</a> --
        <a href="/require_once.php">require_once</a> --
        <a href="/include_once.php">include_once</a> --';
?>
```

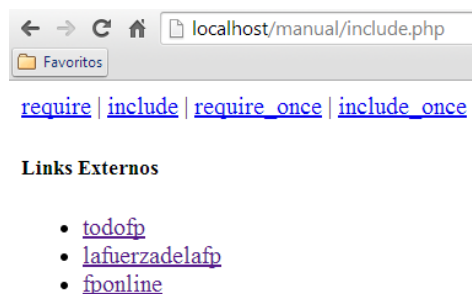
[header.php](#)

```
<div id="sectionLinks">
  <h2> Links Externos </h2>
  <ul>
    <li><a href="http://todofp.es/">todofp</a></li>
    <li><a href="http://lafuerzadelafp.es">lafuerzadelafp</a></li>
    <li><a
href="http://www.educacion.gob.es/fponline/">fponline</a></li>
  </ul>
</div>
```

[menu.html](#)

```
<?php
require 'header.php';
include 'menu.html';
?>
```

[include.php](#)



2. FUNCIONES

Sintaxis:

```
function nombre_funcion ($parametro1, ..., $parametroN) {  
    sentencias;  
}
```

- Las funciones no se ejecutan inmediatamente al cargar la página php en el servidor, solo cuando se llaman.

- Las funciones pueden recibir varios valores mediante los parámetros.
PHP soporta paso de parámetros por valor, por referencia y por defecto.

- Las funciones pueden retornar un valor mediante return:

```
return $valor;
```

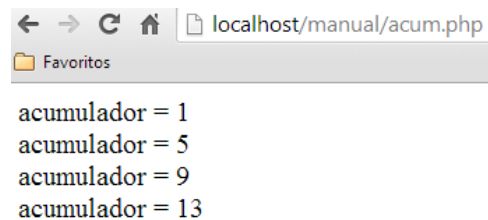
Ejemplo 1: **Paso de parámetros por valor**

```
<?php  
function sumar ($a1, $a2) {           // paso de parámetros por  
    valor  
    return $a1 + $a2;  
}  
$s = sumar (2, 4);  
echo "La suma es: $s";  
?>
```

[sumar.php](#)

Ejemplo 2: **Paso de parámetros por referencia** (añadiendo "&" al argumento)

```
<?php  
function acumular (&$a, $incremento)  
{  
    $a = $a + $incremento;  
}  
$acum = 1;  
echo "acumulador = $acum <br>";  
for ($i=1; $i<4; $i++){  
    acumular ($acum, 4);  
    echo "acumulador = $acum <br>";  
}
```



```
localhost/manual/acum.php  
Favoritos  
acumulador = 1  
acumulador = 5  
acumulador = 9  
acumulador = 13
```

[acum.php](#)

El paso de parámetros por referencia permite a una función cambiar el valor del parámetro. En el ejemplo 2 el parámetro \$acum es modificado por referencia dentro de la función, al cambiar el valor de \$a.

Es decir, \$acum → &\$a (\$acum apunta a la dirección de \$a).

Actividad:

Quita el símbolo “&” del primer argumento y comprueba como \$acum no varía.

Ejemplo 3: **paso de parámetros por defecto**

```
<?php
function soñar ($a = 'ser rico') { // paso de parámetro por
defecto
    return "Quiero $a";
}
echo soñar ('un Volvo V40 <br>');
echo soñar ();
?>
```

[defecto.php](#)

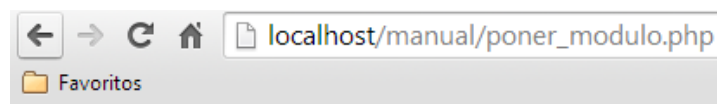
Si la llamada no tiene parámetros se usa el valor por defecto definido en la función.

Ejemplo 4: **cantidad variable de parámetros**

PHP dispone de las funciones `func_num_args ()`, `func_get_arg ()` y `func_get_args()` para el manejo de funciones con un número variable de parámetros.

```
<?php
function poner_modulo ( ) {
    for ($i=1; $i<func_num_args(); $i++) {
        echo func_get_arg($i);
    }
    echo "<br>";
}
poner_modulo ('DES'); // 1 parámetro
poner_modulo ('Desarrollo', 'Entorno', 'Servidor'); //
3 parámetros
?>
```

[poner_modulo.php](#)



DES
Desarrollo Entorno Servidor

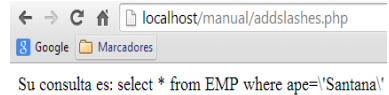
3. MANEJO DE CADENAS

Vamos a ver algunas funciones predefinidas en PHP para manejo de cadenas:

- Cuando hacemos consultas a una Base de Datos, usamos los caracteres especiales: Comillas simples ('), Comillas dobles ("), Barra invertida (\) y NULL

La función **addslashes** (string) añade un carácter de barra invertida (\) a los anteriores, para que el intérprete de PHP no lo tome como un carácter significativo.

```
<?php
$consulta1 = "select * from EMP where
ape='Santana' ";
echo 'Su consulta es: ' . addslashes ($consulta1);
?>
```



[addslashes.php](#)

- Si una cadena en PHP va a ser utilizada para consultar una URI en HTML, necesitamos que se mantengan los espacios y los caracteres alfanuméricos. La función **urlencode** (string) reemplaza los caracteres alfanuméricos por el símbolo % y dos dígitos que representan el carácter, y los espacios por el símbolo +.

```
<?php
$uri = "http://www.salude.es";
echo "Su dirección es: $uri";
echo "Su dirección codificada es: " . urlencode
($uri);
?>
```

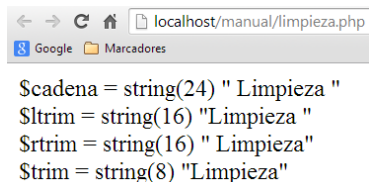


[uri.php](#)

- Para limpieza de cadenas.
 - **rtrim** (string) elimina caracteres predefinidos por la derecha de la cadena
 - **ltrim** (string) elimina caracteres predefinidos por la izquierda de la cadena
 - **trim** (string) elimina caracteres predefinidos por ambos lados de la cadena
 - **strip_tags** (string) elimina etiquetas HTML de la cadena

```
<?php
$cadena = " \0 \r \n Limpieza \n \r \0 ";
$ltrim = ltrim($cadena);
$rtrim = rtrim ($cadena);
$trim = trim ($cadena);

echo '$cadena = '; var_dump ($cadena); echo
"<br>";
echo '$ltrim = '; var_dump ($ltrim); echo
"<br>";
echo '$rtrim = '; var_dump ($rtrim); echo
"<br>";
echo '$trim = '; var_dump ($trim); echo "<br>";
?>
```



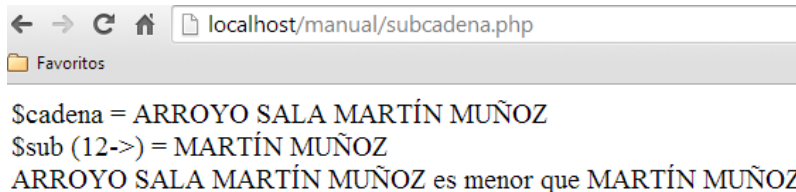
[limpieza.php](#)

- Para manejo de subcadenas
 - **strlen** (string) proporciona el n° de caracteres (longitud) de la cadena
 - **substr** (string, inicio, [n]) proporciona la subcadena que empieza en la posición “inicio” y tiene longitud “n” (parámetro opcional).
 - **strcmp** (string1, string2) devuelve un entero negativo si string1 es menor que string2, positivo si string1 es mayor que string2 y 0 si son iguales.
La comparación se realiza carácter a carácter empezando por la izquierda y en case sensitive (distingue mayúsculas de minúsculas).

```
<?php
$cadena = "ARROYO SALA MARTÍN MUÑOZ";
$sub = substr ($cadena, 12);
$cmp = strcmp ($cadena, $sub);

echo "\$cadena = $cadena <br>";
echo "\$sub (12->) = $sub <br>";
if ($cmp>0)
    echo "$cadena es mayor que $sub";
else if ($cmp<0)
    echo "$cadena es menor que $sub";
else
    echo "Las cadenas son iguales";
?>
```

[subcadena.php](#)



```
← → ↻ 🏠 📄 localhost/manual/subcadena.php
Favoritos

$cadena = ARROYO SALA MARTÍN MUÑOZ
$sub (12->) = MARTÍN MUÑOZ
ARROYO SALA MARTÍN MUÑOZ es menor que MARTÍN MUÑOZ
```

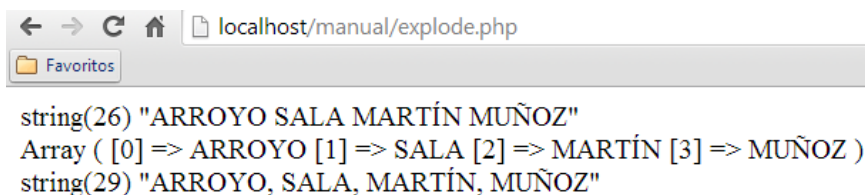
- Conversión de cadenas en arrays
 - **explode** (separador, string, [límite]) divide una cadena de caracteres según un “separador” y la convierte en un array con un número “límite” de elementos.
 - **implode** (separador, string) convierte un array de varios elementos en una sola cadena separada por el “separador”.

```
<?php
$cadena = 'ARROYO SALA MARTÍN MUÑOZ';
$array = explode (' ', $cadena);
$cadena2 = implode (' ', $array);

var_dump ($cadena); echo '<br>';
print_r ($array); echo '<br>';
var_dump ($cadena2);

?>
```

[explode.php](#)



```
string(26) "ARROYO SALA MARTÍN MUÑOZ"
Array ( [0] => ARROYO [1] => SALA [2] => MARTÍN [3] => MUÑOZ )
string(29) "ARROYO, SALA, MARTÍN, MUÑOZ"
```

- Sintaxis heredoc

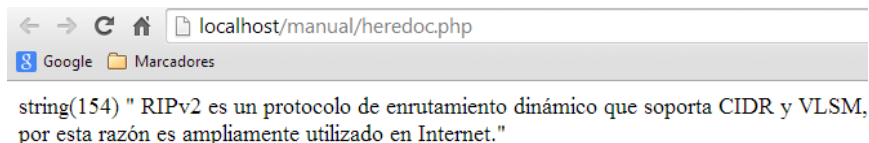
Otra forma de declarar variables de tipo string es utilizar la sintaxis heredoc y nowdoc, y resulta muy útil cuando el texto es largo. Se usa el operador <<< seguido de un delimitador, después la cadena y por último el mismo delimitador;

```
<?php
$cadena = <<<EOF
    RIPv2 es un protocolo de enrutamiento dinámico
    que soporta CIDR y VLSM, <br> por esta razón es
    ampliamente utilizado en Internet.
EOF;
var_dump ($cadena);

?>
```

[heredoc.php](#)

Observación: el delimitador final no puede llevar otros caracteres de ningún tipo, ni sangría, excepto el carácter punto y coma “;”



```
string(154) "RIPv2 es un protocolo de enrutamiento dinámico que soporta CIDR y VLSM,
por esta razón es ampliamente utilizado en Internet."
```

4. FUNCIONES PARA MANEJO DE FECHAS

En PHP no existe el tipo de datos fecha, se trabaja con cadenas de caracteres y funciones que extraen la fecha y hora del sistema.

- **date** (formato, [timestamp]) devuelve una cadena con el formato especificado y que contiene la fecha indicada por el entero “timestamp”.

“timestamp” es opcional y si se omite la función toma la fecha/hora del sistema.

- **strtotime** (string) devuelve un entero que representa la fecha indicada en la cadena

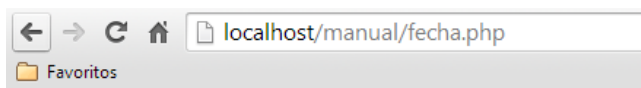
- **mktime** (h, i, sa, m, d, Y) devuelve un entero que representa la fecha indicada por los argumentos

(h=horas, i=minutos, sa=segundos, m=mes, d=día, Y=año)

```
<?php
$fechaSis = date (" l d/m/Y");
$cadena = "11:40am April 16 2011";
$fecha1=strtotime ($cadena);
$fecha2=mktime(14, 0, 0, 7, 10, 2014);

echo 'Hoy es'. $fechaSis. "<br>";
echo 'La hora es '. date("h:i:sa").'<br>';
echo '<br> Formato completo RFC: <br>';
echo date (DATE_RFC2822);
echo '<br><br>Joel nació el '.date("d/m/Y h:i:sa", $fecha1);
echo '<br><br>Las vacaciones comienzan el'.date("d/m/Y h:i:sa",
$fecha2);
?>
```

[fecha.php](#)



Hoy es Friday 30/05/2014

La hora es 05:34:55pm

Formato completo RFC:

Fri, 30 May 2014 17:34:55 +0200

Joel nació el 16/04/2011 11:40:00am

Las vacaciones comienzan el 10/07/2014 02:00:00pm

ANEXO 1. TIPOS DE ERRORES

Este es un listado de los errores más comunes:

Archivo	Tipo de error	Nivel PHP	Descripción
error01.php	Error sintáctico	E_PARSE	Falta un ;
error02.php	Error lógico	E_NOTICE	Se envía a la salida estándar una variable no inicializada
error03.php	Error semántico	E_WARNING	División por 0
error05.php	Error fatal	E_ERROR	Se ha usado () para acceder a elementos de un array
error07.php	Error conceptual	E_STRICT	Un método no estático es llamado estáticamente

```
<?php
echo
"Hola"
echo
"Adiós"
?>
```

[error01.php](#)

localhost/manual/error01.php

Parse error: syntax error, unexpected 'echo' (T_ECHO), expecting ';' or ':' in C:\xampp\htdocs>manual\error01.php on line 3

```
<?php
echo "$a";
?>
```

[error02.php](#)

localhost/manual/error2.php

Notice: Undefined variable: a in C:\xampp\htdocs>manual\error2.php on line 2

```
<?php
$a = 1; $b =
0;
$c = $a /
$b;
echo "$c";
?>
```

[error03.php](#)

localhost/manual/error3.php

Warning: Division by zero in C:\xampp\htdocs>manual\error03.php on line 4

```
<?php
$lista = [1, 2, 3];
for ($i = 0; $i <= 2; $i++)
{
    echo $lista($i);
}
?>
```

[error05.php](#)

localhost/manual/error05.php

Fatal error: Function name must be a string in C:\xampp\htdocs>manual\error05.php on line 4