

Recursosinformáticos

# Aprender a desarrollar un sitio Web con PHP y MySQL

Ejercicios prácticos y corregidos

2<sup>a</sup> edición

Olivier ROLLET

Descarga   
[www.ediciones-eni.com](http://www.ediciones-eni.com)



Podrá descargar algunos elementos de este libro en la página web de Ediciones ENI: <http://www.ediciones-eni.com>.

Escriba la referencia ENI del libro **RIT22PHMY** en la zona de búsqueda y valide. Haga clic en el título y después en el vínculo de descarga.

## Capítulo 1 Introducción

1. Objetivo del libro .....	13
2. Funcionamiento de un sitio Web.....	13

## Capítulo 2 Instalación

1. Introducción .....	17
2. Instalación de EasyPHP 13.1 .....	18

## Capítulo 3 Las bases del lenguaje PHP

1. Las etiquetas .....	23
1.1 Sintaxis básica.....	23
1.2 Inserción de etiquetas PHP en el código XHTML .....	24
1.3 Envío de datos al servidor Web .....	26
1.4 Inserción del código XHTML con la instrucción echo.....	27
1.5 Comentarios .....	28
2. Las variables .....	29
2.1 Asignación.....	29
2.2 Tipos de variables.....	30
2.3 Particularidades del tipo de variable string .....	30
2.4 La concatenación .....	32
2.5 Transtipado.....	33
3. Las constantes.....	35

**Capítulo 4**  
**Funciones y estructuras de control**

1.	Las condiciones .....	37
1.1	If .....	37
1.2	Switch .....	42
2.	Los bucles .....	45
2.1	For .....	45
2.2	While .....	47
2.3	Do while .....	48
2.4	Foreach .....	49
3.	Las tablas .....	49
3.1	Tabla numérica .....	49
3.2	Tabla asociativa .....	51
3.3	Recorrido de una tabla .....	52
3.3.1	El bucle for .....	52
3.3.2	El bucle foreach .....	54
3.3.3	print_r .....	56
3.4	Funciones en las tablas .....	56
3.4.1	Longitud de una tabla .....	56
3.4.2	Existencia de un valor en una tabla .....	57
3.4.3	Existencia de una clave en una tabla .....	58
3.4.4	Ordenar una tabla .....	59
3.4.5	Búsqueda en una tabla .....	63
3.4.6	Recorrer una cadena de caracteres en una tabla .....	64
3.4.7	Reagrupar los valores de una tabla en una cadena .....	65
3.4.8	Trocear una cadena con una longitud fija .....	66
3.4.9	Añadir elementos al final de la tabla .....	67
3.4.10	Eliminar un elemento al final de la tabla .....	68
3.4.11	Selección de un elemento de la tabla de forma aleatoria .....	69
3.5	Tabla de varias dimensiones .....	70
3.5.1	Por medio de una tabla temporal .....	71
3.5.2	Almacenar directamente los valores en la tabla general .....	71

3.6 Ejercicios .....	72
3.6.1 Enunciados.....	72
3.6.2 Soluciones .....	73
4. Procesamiento de las cadenas de caracteres .....	76
4.1 Funciones de manipulación de cadenas.....	76
4.1.1 <code>strlen()</code> .....	76
4.1.2 <code>substr()</code> .....	77
4.1.3 <code>strstr()</code> .....	78
4.1.4 <code>str_replace()</code> .....	79
4.1.5 <code>trim()</code> .....	80
4.1.6 <code>strtolower()</code> .....	81
4.1.7 <code>strtoupper()</code> .....	81
4.1.8 <code>strpos()</code> .....	82
4.1.9 <code>str_word_count()</code> .....	83
4.1.10 <code>str_pad()</code> .....	85
4.2 Las expresiones regulares .....	85
4.2.1 Las mayúsculas y las minúsculas .....	87
4.2.2 Búsqueda de una palabra, y no una cadena .....	88
4.2.3 El símbolo O.....	90
4.2.4 Comienzo de la cadena .....	90
4.2.5 Fin de cadena .....	91
4.2.6 Un carácter en una clase .....	91
4.2.7 Rango de caracteres en una clase .....	92
4.2.8 La no presencia de un rango de caracteres en una clase ..	93
4.2.9 Los cuantificadores .....	94
4.2.10 Intervalos de reconocimiento .....	96
5. Los operadores.....	98
5.1 Operadores de cadena.....	98
5.1.1 La concatenación .....	98
5.1.2 Asignación .....	98
5.2 Operadores aritméticos .....	99
5.2.1 La suma .....	99
5.2.2 La resta .....	100

5.2.3 La multiplicación .....	100
5.2.4 La división .....	101
5.2.5 El módulo .....	101
5.2.6 El incremento .....	102
5.2.7 La resta .....	103
5.3 Operadores de comparación .....	104
5.3.1 La igualdad .....	105
5.3.2 La diferencia .....	106
5.3.3 La comparación .....	107
5.4 El operador ternario .....	108
5.5 Operadores lógicos .....	108
5.5.1 Y .....	109
5.5.2 O .....	111
6. Las funciones .....	111
6.1 Creación .....	114
6.2 Alcance de las variables .....	115
6.3 Las variables globales .....	116
6.4 Las variables estáticas .....	117
6.5 Funciones útiles .....	120
6.6 Paso por referencia .....	121
6.7 Funciones de la función de gestión .....	123
6.8 Recursividad .....	125
6.9 Funciones predefinidas en PHP .....	126
6.9.1 Generar un número aleatorio .....	126
6.9.2 Redondear un número decimal .....	127
6.9.3 Tomar el valor absoluto de un número .....	128
6.9.4 Crear un identificador único .....	128
6.9.5 Mostrar información de PHP .....	130
6.9.6 Enviar un e-mail .....	131
7. Almacenar una función en una variable .....	132
7.1 Ejercicio .....	132
7.1.1 Enunciados .....	133
7.1.2 Soluciones .....	

8. Las fechas .....	137
9. Los archivos.....	146
9.1 Introducción .....	146
9.2 Lectura rápida .....	146
9.3 Escritura rápida.....	148
9.4 Abrir y cerrar un archivo .....	148
9.5 Leer y escribir .....	149
9.6 Concurrencia.....	156
9.7 Manipulación de archivos .....	157
9.8 Manipulación de directorios .....	160
10. Los includes.....	167
11. Ejercicios .....	168
11.1 Enunciados .....	168
11.2 Soluciones .....	169

## Capítulo 5

### Transmitir datos de una página a otra

1. Las variables superglobales.....	173
1.1 <code>\$GLOBALS</code> .....	173
1.2 <code>\$_SERVER</code> .....	174
1.3 <code>\$_ENV</code> .....	177
1.4 <code>\$_SESSION</code> .....	179
1.5 <code>\$_COOKIE</code> .....	180
1.6 <code>\$_FILES</code> .....	183
2. El método GET .....	186
2.1 Utilización del método GET .....	186
2.2 Comprobar la presencia de la variable en la URL.....	189
2.3 Comprobar el valor de la variable en la URL .....	190
2.4 Información complementaria .....	192
2.4.1 Argumentos con el mismo nombre .....	192
2.4.2 Argumentos de tipo tabla .....	193
2.4.3 Argumentos con caracteres especiales .....	194

3.	El método POST .....	196
3.1	Utilización del método POST .....	196
3.2	Los diferentes elementos del formulario .....	198
3.2.1	Campo de tipo texto .....	199
3.2.2	Campo de tipo contraseña .....	201
3.2.3	Área de texto .....	201
3.2.4	Lista desplegable de elección simple.....	202
3.2.5	Lista desplegable de elección múltiple .....	204
3.2.6	Lista de casillas de selección .....	206
3.2.7	Lista de botones de opción .....	208
3.2.8	Los campos ocultos .....	209
3.2.9	El botón submit .....	210
3.2.10	El botón reset .....	213
3.2.11	Formulario completo.....	213
4.	Otros métodos .....	215
4.1	El método \$_REQUEST .....	215
5.	Zonas con el mismo nombre .....	218
6.	Varios formularios en la misma página.....	218
7.	Control de datos y redirección de páginas .....	219
7.1	Introducción .....	219
7.2	Datos obligatorios.....	220
7.3	Eliminación de espacios no deseados.....	220
7.4	Longitud máxima .....	221
7.5	Caracteres permitidos.....	222
7.6	Magic quotes.....	224
7.7	Redirección de página.....	225
8.	Ejercicios .....	225
8.1	Enunciados .....	227
8.2	Soluciones .....	

**Capítulo 6**  
**Manipular una imagen**

1.	La librería GD .....	233
2.	Creación de una imagen.....	233
2.1	Header .....	233
2.2	Creación de una imagen vacía .....	234
2.3	Creación y visualización de una imagen completa.....	234
3.	Texto y color.....	236
3.1	El color.....	236
3.2	El texto .....	237
3.3	La transparencia .....	238
3.4	Cambiar el tamaño de una imagen .....	239
3.5	Superponer las imágenes .....	240
4.	Las formas .....	243
5.	Ejemplos .....	245
5.1	Ejemplo 1.....	245
5.2	Ejemplo 2.....	246

**Capítulo 7**  
**Base de datos MySQL**

1.	Presentación .....	249
1.1	Introducción .....	249
1.2	Estructura .....	250
2.	PHPMyAdmin.....	251
3.	El lenguaje SQL.....	263
3.1	Presentación .....	263
3.2	Leer datos .....	264
3.3	Escribir datos.....	265
3.4	Filtrar datos.....	267
3.5	Los alias .....	270

3.6 Ordenar datos .....	271
3.7 Eliminar datos .....	272
3.8 Modificar datos .....	273
3.9 Las uniones .....	275
3.10 El agrupamiento .....	279
4. SQL avanzado .....	282
4.1 Las funciones e instrucciones SQL .....	282
4.1.1 Limitar datos .....	282
4.1.2 Valores distintos .....	283
4.1.3 Convertir en mayúsculas .....	284
4.1.4 Convertir en minúsculas .....	285
4.1.5 Redondear un número decimal .....	286
4.1.6 Valor absoluto de un número decimal .....	286
4.1.7 Número aleatorio .....	287
4.1.8 Longitud de un campo .....	287
4.1.9 Eliminar los espacios de un campo .....	288
4.1.10 Extraer una subcadena de un campo .....	289
4.1.11 Concatenar varios campos .....	290
4.1.12 Posición de una cadena de caracteres en un campo .....	291
4.1.13 Añadir una secuencia de caracteres .....	292
4.1.14 Sustitución de una cadena de caracteres .....	292
4.1.15 Comprobar el valor de un campo .....	294
4.1.16 Examinar la fecha actual .....	295
4.1.17 Extraer la fecha de un campo date y hora .....	295
4.1.18 Diferencia entre dos fechas .....	295
4.1.19 Añadir un intervalo de tiempo a una fecha .....	297
4.1.20 Añadir un intervalo de tiempo a una hora .....	297
4.1.21 Sustracción de un intervalo de tiempo a una fecha .....	298
4.1.22 Sustracción de un intervalo de tiempo a una hora .....	298
4.1.23 Unir dos consultas .....	299
4.2 Las subconsultas .....	304
4.3 Los procedimientos almacenados y funciones .....	

4.4	Otros objetos de MySQL .....	310
4.4.1	Las tablas .....	310
4.4.2	Los índices .....	311
4.4.3	Las vistas .....	312
4.4.4	Trigger .....	313
5.	Ejercicios SQL .....	315
5.1	Enunciados .....	317
5.2	Soluciones .....	320
6.	Acceso a las bases de datos con PHP .....	321
6.1	Introducción .....	321
6.2	Conexión .....	322
6.3	Desconexión .....	324
6.4	Consultas no preparadas .....	324
6.4.1	Leer datos .....	324
6.4.2	Escribir datos .....	330
6.4.3	Eliminar datos .....	332
6.4.4	Actualizar datos .....	333
6.5	Consultas preparadas .....	335
6.5.1	Introducción .....	335
6.5.2	Leer datos .....	335
6.5.3	Escribir datos .....	338
6.5.4	Modificar datos .....	340
6.5.5	Eliminar datos .....	341
6.5.6	Almacenar un resultado .....	343
6.5.7	Examinar los errores de una consulta preparada .....	345
7.	PDO .....	346
7.1	Introducción .....	346
7.2	Conexión .....	348
7.3	Consultas no preparadas .....	350
7.3.1	Leer datos .....	350
7.3.2	Escribir datos .....	352
7.3.3	Eliminar datos .....	354

7.3.4	Actualizar datos.....	355
7.4	Consultas preparadas.....	356
7.4.1	Leer datos.....	356
7.4.2	Escribir datos.....	358
7.4.3	Eliminar datos.....	360
7.4.4	Modificar datos.....	361
7.4.5	Llamar a un procedimiento almacenado .....	361
8.	Ejercicios .....	364
8.1	Enunciados .....	364
8.2	Soluciones.....	366

## **Capítulo 8**

### **El objeto**

1.	Introducción .....	375
2.	La clase .....	376
2.1	Introducción .....	376
2.2	La encapsulación.....	377
2.3	Visibilidad de los atributos y de los métodos .....	377
2.4	Añadir un método a la clase.....	379
2.5	Utilización de la clase.....	380
2.6	Actualizar y leer los atributos de la instancia.....	380
2.7	Paso como argumento de tipo objeto .....	384
2.8	El constructor .....	387
2.9	El destructor .....	389
2.10	Ejercicio.....	390
2.11	Las constantes de clase.....	391
2.12	Los atributos y métodos estáticos.....	393
2.12.1	Método estático.....	393
2.12.2	Atributo estático .....	396

3.	La herencia .....	399
3.1	Introducción .....	399
3.2	Protected .....	403
3.3	Sustitución .....	404
3.4	Herencia en cascada .....	407
4.	Las clases abstractas .....	408
5.	Las clases finales .....	411
6.	Los métodos mágicos .....	412
7.	Namespaces.....	420
8.	Ejercicios .....	424
8.1	Enunciados .....	424
8.2	Soluciones .....	428

## **Capítulo 9** **Configuración**

1.	PHP.ini.....	449
2.	Archivo de configuración MySQL: My.ini .....	451
3.	Archivo de configuración Apache: Httpd.conf.....	452

## **Capítulo 10** **Seguridad y rendimiento**

1.	Fallos de seguridad XSS .....	455
1.1	XSS no permanente .....	455
1.2	XSS permanente.....	457
1.3	Error de página .....	457
2.	Derechos de la base de datos .....	458
3.	Inyección SQL (addslashes) .....	460
4.	Comprobación de la sesión .....	462

5.	Rendimiento .....	463
5.1	Optimizar el rendimiento en PHP.....	463
5.2	Optimizar el rendimiento en MySQL.....	464

## **Capítulo 11**

### **Casos prácticos y corregidos**

1.	Crear un blog (procedimiento) .....	465
2.	Crear un blog (objeto) .....	466
3.	Crear una newsletter .....	470
4.	Crear un flujo RSS .....	478
5.	Gestión de un parque informático en MVC.....	479
6.	Crear un sitio Web para gestionar becarios.....	480

Índice.....	511
-------------	-----

# Aprender a desarrollar un sitio Web con PHP y MySQL

## Ejercicios y corregidos (2<sup>a</sup> edición)

Este libro se dirige a un **público de programadores principiantes** que ya conocen HTML y CSS y que quieren entender el funcionamiento de una aplicación Web para poder crear sus propios **sitios Web dinámicos con PHP y MySQL**.

En la primera parte del libro, el lector va a instalar su entorno de desarrollo EasyPHP y va a descubrir las **bases del lenguaje PHP**, (con la versión 5.5) sus principales funciones y estructuras de control, así como una explicación sobre la **transmisión de datos** entre las páginas y la **librería gráfica** (efectos especiales sobre una imagen). Estas partes teóricas se acompañan de numerosos ejemplos.

Lo mismo ocurre en la segunda parte del libro, dedicada al **lenguaje SQL**. El lector va a descubrir una **base de datos MySQL** y los distintos métodos para acceder a ella con PHP (PDO, SQL Avanzado), así como la manera de **asegurar la seguridad** de la base de datos. Un capítulo se dedica a los primeros pasos con **Programación Orientada a Objetos** y otro a la **administración de la configuración y su rendimiento**.

Para que el lector pueda poner en práctica los conocimientos aprendidos, el autor ha preparado **numerosos ejercicios** al final de cada capítulo (ejemplos: cómo crear un blog, una newsletter, un sitio de administración...) y expone los ejercicios corregidos.

Los **elementos complementarios** se descargan en esta página.

### Los capítulos del libro:

Introducción – Instalación – Las bases del lenguaje PHP – Funciones y estructuras de control – Transmitir datos de una página a otra – Manipular una imagen – Base de datos MySQL – El objeto – Configuración – Seguridad y rendimiento – Casos prácticos y corregidos

---

### Olivier ROLLET

Diplomado en Ingeniería Eléctrica e Informática Industrial, **Olivier ROLLET** ha trabajado durante doce años como programador en numerosos proyectos de sitios Web. Hoy en día es formador en el área del desarrollo de nuevas tecnologías Web y conoce a la perfección las expectativas de los lectores, proporcionándoles un libro operativo para adentrarse en la creación de sitios Web dinámicos.

## Objetivo del libro

Este libro se dirige a un público con conocimientos en la creación de sitios Web estáticos de HTML (*Hypertext Markup Language*), aunque no es necesario que tenga conocimientos en desarrollo informático o algorítmica.

El objetivo de este libro es explicar cómo crear de manera dinámica e interactiva un sitio Web, con ayuda de ejemplos. Tras la lectura de este libro, será capaz de instalar y crear un sitio Web con PHP (*Hypertext Processor*)/MySQL.

Los requisitos previos son: conocimientos de HTML y algunas nociones de JavaScript, con algunos ejercicios.

# Funcionamiento de un sitio Web

Cuando ejecuta una URL (*Uniform Resource Locator*), por ejemplo <http://www.google.es>, ¿qué ocurre en el navegador (Internet Explorer, Firefox, Chrome...)?

La URL se envía a un servidor Web a través de la red. Este servidor procesa la solicitud y reenvía el flujo HTML al navegador.

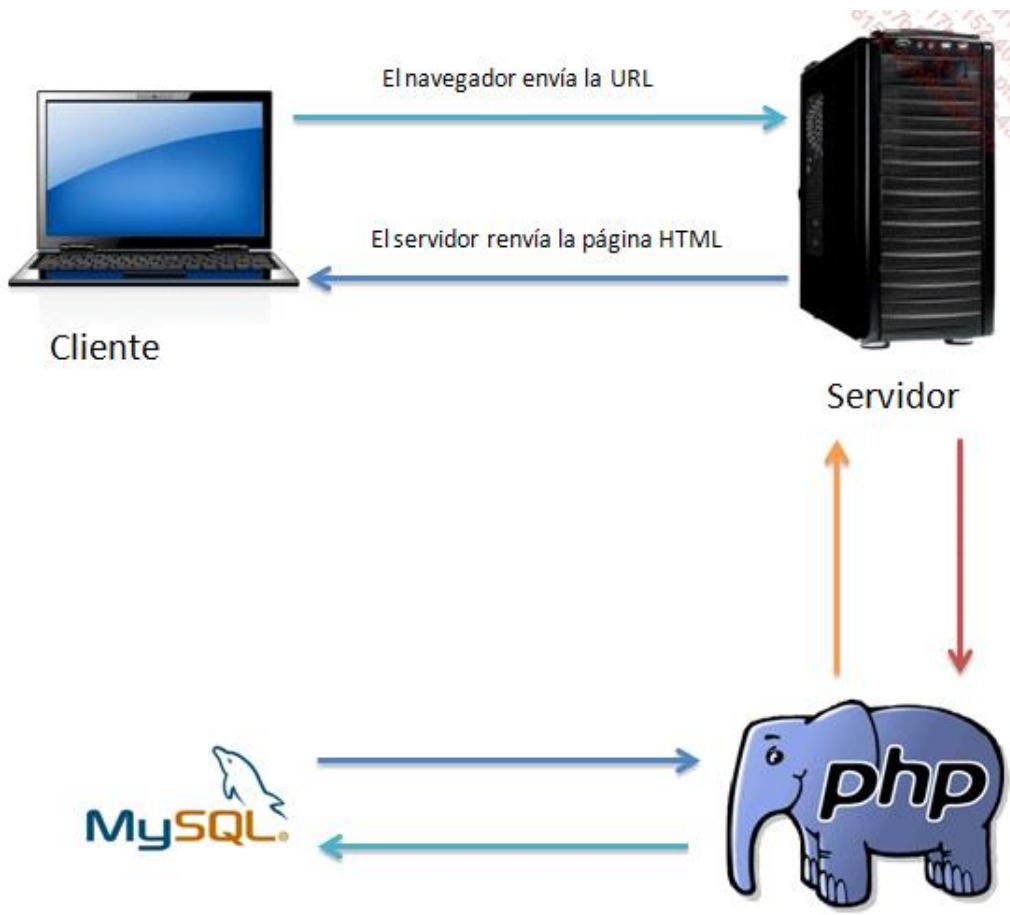
Existen dos tipos de sitios Web: los sitios Web estáticos y los dinámicos.

Los sitios Web estáticos solo contienen código cliente, es decir, un código que puede interpretar su navegador. Por ejemplo: HTML y JavaScript.



En cambio, los sitios Web dinámicos contienen lenguaje servidor porque el servidor Web interpreta este lenguaje y envía al navegador el flujo HTML. Por ejemplo, PHP, Java (no confundir con JavaScript) y Active Server Pages.NET (ASP.NET) son lenguajes del lado servidor.

Cuando almacena información en su sitio Web, por ejemplo un inicio de sesión y una contraseña, un blog, artículos, etc., está obligado a utilizar una base de datos para almacenar la información y utilizar un lenguaje servidor que pueda leer o escribir la información en la base de datos.



Cuando ejecuta una URL desde un sitio Web dinámico (por ejemplo: <http://google.es>), se ejecutan dos procesos diferentes:

- 1) El servidor Web (Apache es el más frecuente) lee su URL y ejecuta el código servidor (en nuestro ejemplo, PHP) que va a crear un código HTML. A continuación el servidor envía a su navegador el flujo HTML que se ha creado.
- 2) Su navegador recibe el flujo HTML y lo interpreta de manera gráfica, mostrando la página HTML.

De este modo, cuando quiera probar en su equipo un sitio Web estático, simplemente haga doble clic en la página HTML para abrirla en su navegador.

La URL será del tipo: C:\HTML\li.html

Pero si quiere probar un sitio Web dinámico con PHP, obligatoriamente debe instalar en su equipo un servidor Web (por ejemplo, Apache) y ejecutar su página PHP con una URL de tipo: <http://127.0.0.1/test.php>

## Introducción

Para crear un sitio Web estático, solo necesita un navegador (Internet Explorer, Firefox, Chrome...) y un editor de texto, por ejemplo Notepad++ o Sublime Text.

Para crear un sitio Web dinámico, necesita un navegador, un servidor Web que ejecute las páginas PHP y un servidor de base de datos.

En este caso utilizaremos el servidor Web Apache y el servidor de base de datos MySQL, que son las herramientas más utilizadas para crear sitios Web de poca envergadura. Son gratuitos y están disponibles como un paquete. Se instalan fácilmente en UNIX, Windows y Mac.

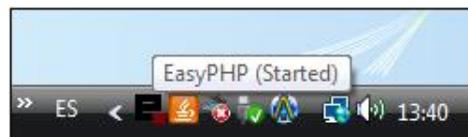
Los paquetes más utilizados son WAMP (XAMP en Linux, MAMP en Mac) y EasyPHP. En este libro utilizaremos EasyPHP 13.1 de Windows.

Para más información puede consultar las páginas Web [www.php.net](http://www.php.net), <http://php.net/manual/es/index.php> y [www.easypht.org](http://www.easypht.org) (desde este último enlace puede descargar la última versión de EasyPHP).

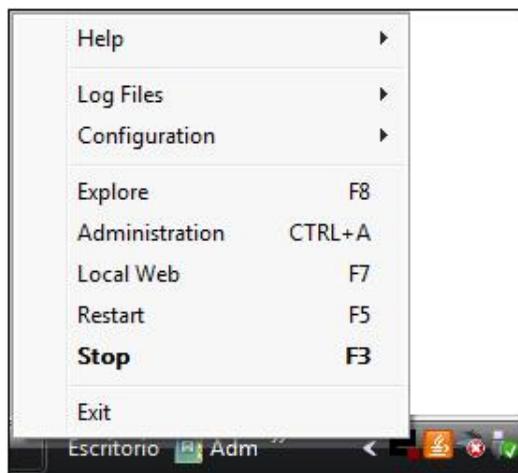
## Instalación de EasyPHP 13.1

De manera predeterminada, EasyPHP se instalará en: C:\ProgramFiles\EasyPHP-DevServer-13.1VC11.

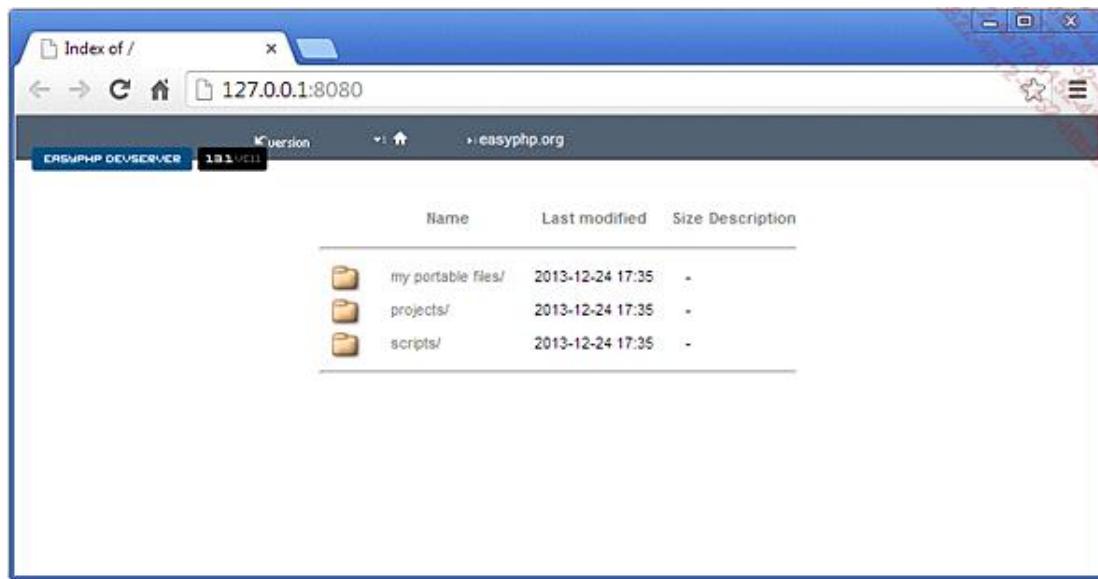
Una vez instalado, haga clic en **Mostrar los iconos ocultos** y verá el ícono  en el área de notificación de la barra de tareas:



Haga clic con el botón derecho del ratón en  y aparecerá un menú:



Haga clic en **Local Web**; se abrirá en su navegador la siguiente página:



Apache le envía esta página Web y la URL es <http://127.0.0.1/>. Esta dirección, también llamada localhost, corresponde a la dirección de su servidor Web en modo local. Solo quien utilice este equipo visualizará esta página y las páginas PHP que va a crear.

Las páginas PHP que va a codificar se almacenarán en el directorio C:\Program Files\EasyPHP-DevServer-13.1VC11

\data\localweb.

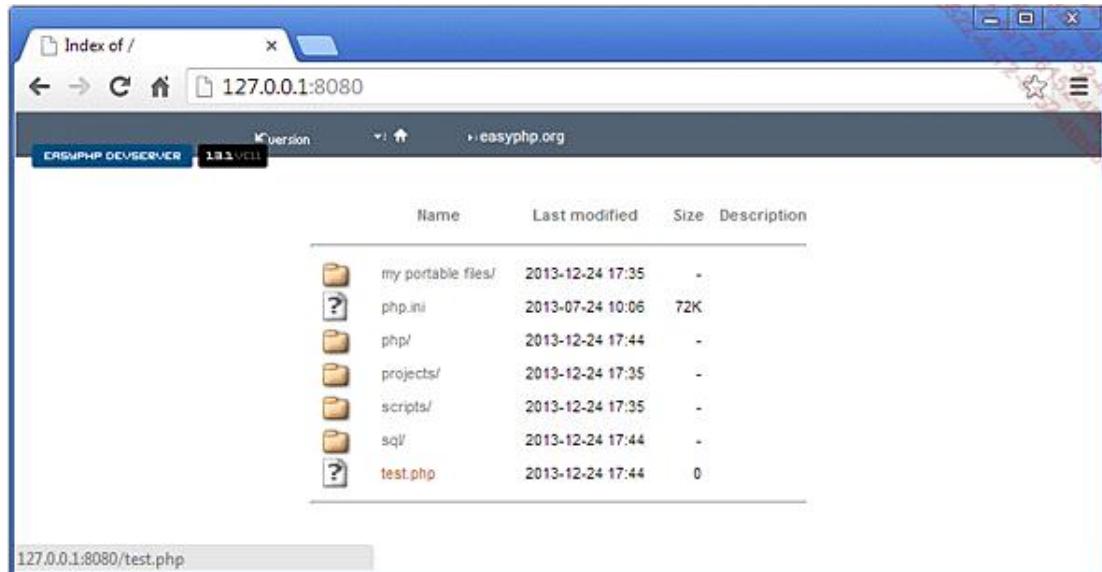
Atención: en versiones anteriores, las páginas PHP se ubicaban en la carpeta www.

Observe que hay tres carpetas por defecto, que están vacías y que se pueden eliminar si lo desea.

Se utilizará Google Chrome como navegador por defecto.

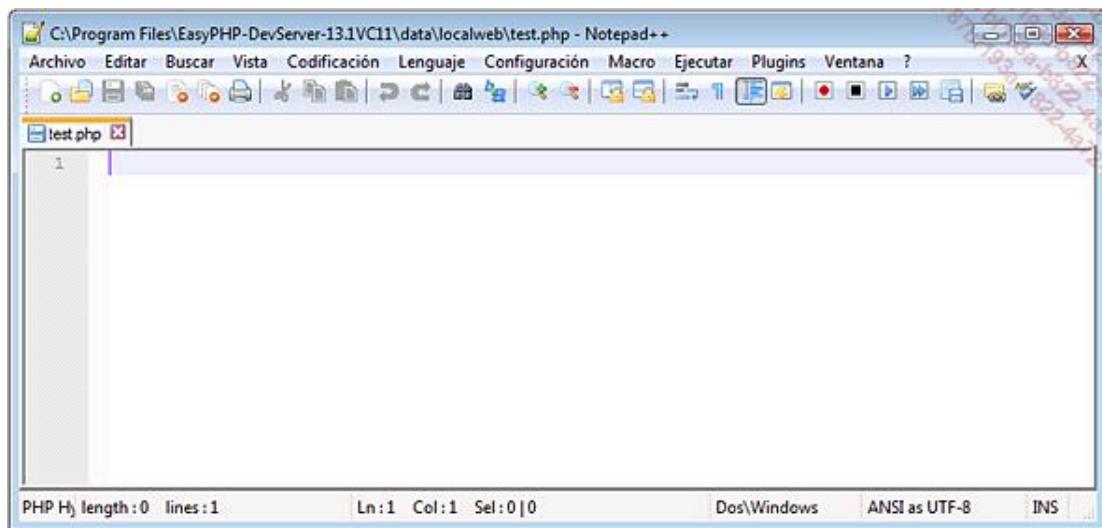
Por ejemplo, cree un archivo vacío y guárdealo como C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\test.php.

En su navegador, vuelva a la dirección http://127.0.0.1, actualice la página con [F5] y aparecerá su página test.php.



Ahora solo le queda elegir un editor de archivos para escribir sus páginas PHP.

Vamos a utilizar Notepad++. Es gratuito, se instala fácilmente y contiene una función muy práctica, que consiste en asignar colores dependiendo de la sintaxis, para visualizar los errores de sintaxis HTML o PHP.



A continuación, escriba su código HTML y PHP y guarde la página en el archivo C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb. Su página siempre debe tener la extensión .php, salvo si se trata de una configuración

especial del servidor Web. En el siguiente ejemplo, la página se llama test.php y contiene el código HTML y PHP entre las etiquetas <?php y ?>.

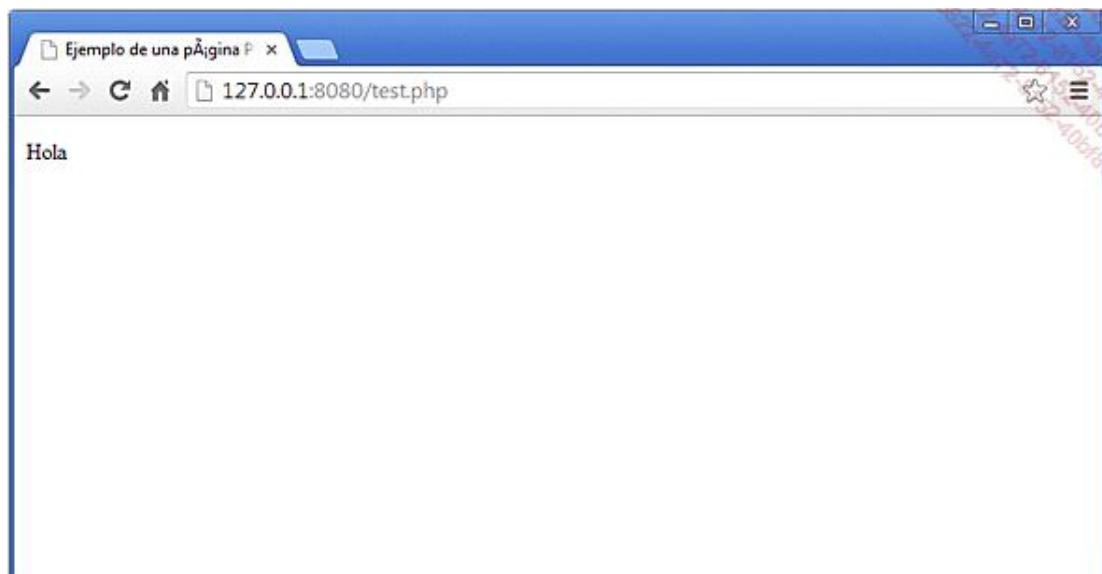
The screenshot shows the Notepad++ interface with the file 'test.php' open. The code contains an XML declaration, an HTML document structure with a title and body, and a PHP echo statement. The status bar at the bottom shows the file length (273), number of lines (10), current line (Ln:10), column (Col:8), selection (Sel:0|0), encoding (Dos\Windows), and character set (ANSI as UTF-8). A red box highlights the PHP echo line.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title> Ejemplo de una página PHP </title>
</head>
<body>
<?php echo '<p>Hola</p>'?>
</body>
</html>
```

En el siguiente ejemplo, la función echo crea <p>Hola</p> en HTML y el navegador recibe todo el flujo HTML de la página:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Ejemplo de página PHP</title>
</head>
<body>
<p>Hola</p>
</body>
</html>
```

Y dará como resultado: Hola



Haga clic con el botón derecho del ratón y elija en su navegador la opción **Ver código fuente de la página**. Solo visualizará el código HTML, nunca el código servidor en PHP. No se preocupe si no lo ha entendido todo, porque vamos a explicar más adelante cómo se escribe una página PHP.

# Las etiquetas

## 1. Sintaxis básica

XHTML (*Extensible Hypertext Markup Language*) es un lenguaje de etiquetas. Es el sucesor de HTML y se basa en la sintaxis de XML. Asegura la compatibilidad tanto en equipos clásicos como en smartphones.

Ya conoce las etiquetas <html>, <body>, <head>...

Escriba PHP entre dos etiquetas. Se definen de la siguiente manera:

<?php: indica el comienzo del código PHP

?>: indica el final del código PHP

Si usa Notepad++, estas etiquetas se mostrarán en color rojo.

Si retomamos el ejemplo del capítulo Instalación, el código PHP se escribirá de la siguiente manera:

```
<?php  
echo '<p>Hola</p>';  
?>
```

También puede escribir este código en una sola línea:

```
<?php echo '<p>Hola</p>'; ?>
```

Existen otras formas de escribir estas etiquetas. En lugar de <?php y ?, puede escribir:

```
<script language="php"> </script>
```

O bien:

```
<% %>
```

Y también:

```
<? ?>
```

**Atención:** en el archivo de configuración php.ini, las dos últimas sintaxis requieren un parámetro especial.

## 2. Inserción de etiquetas PHP en el código XHTML

Puede insertar un código PHP en cualquier ubicación del código XHTML.

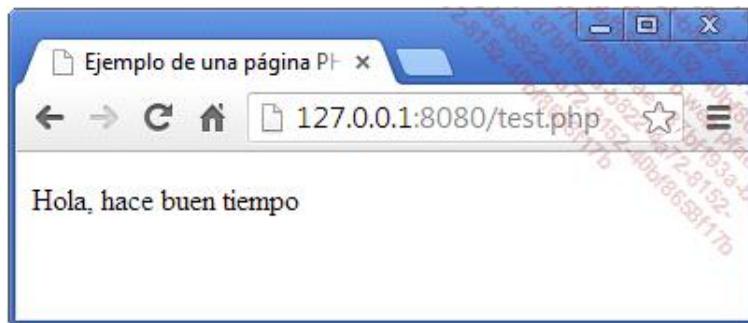
```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo de página PHP</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        Hola, hace <?php echo 'buen tiempo'; ?>
    </body>
</html>

```

Este código genera la siguiente página web en Chrome:



Haga clic con el botón derecho del ratón y escoja **Ver código fuente de la página**; obtendrá el siguiente código:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo de página PHP</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        Hola, hace buen tiempo </body>
</html>

```

Este es el código HTML que el navegador ha recibido e interpretado. Observe que no se puede ver el código servidor entre las etiquetas php.

También puede generar de forma dinámica el título de la página HTML, es decir, el contenido de la etiqueta <title>.

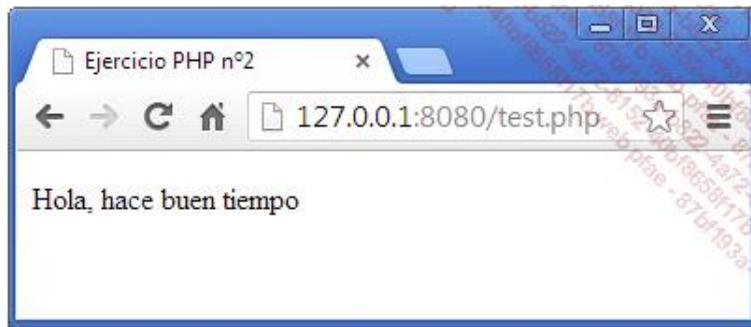
```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo PHP nº2 <?php echo 'PHP nº2'; ?></title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        Hola, hace <?php echo 'buen tiempo'; ?>
    </body>
</html>
```

Este código va a generar en Chrome lo siguiente:



### 3. Envío de datos al servidor Web

Existen varias instrucciones para enviar datos al servidor, es decir, para insertar código HTML en una página Web.

La primera instrucción es **echo** y se escribe de la siguiente manera:

```
<?php echo 'texto'; ?>
```

También puede escribir este código:

```
<?php echo "texto"; ?>
```

O bien:

```
<?php echo('texto'); ?>
```

Atención: si utiliza la primera sintaxis, no se interpretarán las variables (vea el siguiente ejemplo).

La segunda instrucción es **print** y se escribe de la siguiente manera:

```
<?php print('texto'); ?>
```

Por tanto, **print** equivale a **echo**.

Existen otras variantes de **print**:

- **printf()**: muestra una cadena de caracteres formateada.
- **sprintf()**: devuelve una cadena formateada.
- **vprintf()**: muestra una cadena formateada.
- **sscanf()**: analiza una cadena con ayuda de un formato.
- **fscanf()**: analiza un archivo en función del formato.
- **flush()**: vacía los búferes de salida.

Observe que una instrucción siempre termina con un punto y coma.

También puede escribir varias instrucciones en la misma línea, siempre y cuando vayan separadas por punto y coma.

```
<?php echo 'texto'; ?> equivale a <?php echo 'tex'; echo 'to'; ?> y a  
<?php      echo 'tex';  
            echo 'to';  
?>
```

#### 4. Inserción del código XHTML con la instrucción echo

La función **echo** permite insertar cualquier código HTML, por ejemplo:

```
<?php echo '<table><tr><td>texto</td></tr></table>'; ?>
```

Y como resultado inserta una tabla HTML.

También puede insertar una imagen de la siguiente manera:

```
<?php echo ''; ?>
```

Por tanto, puede escribir una página Web completa con la instrucción **echo**.

```
<?php  
$nombre = "Juan Carlos"  
echo '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" ,  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">';  
echo '<html xmlns="http://www.w3.org/1999/xhtml">';  
echo '<head>';  
echo '<title>PHP ENI</title>';  
echo '</head>';  
echo '<body>';  
echo '<p>';  
echo "Hola $nombre !<br />";
```

```
echo 'La fecha es 7 de marzo de 2013.'; //
echo '</p>';
echo '</body>';
echo '</html>';

?>
```

El uso de las variables permite convertir el sitio en dinámico, es decir, la información mostrada se genera en una base de datos cuyo contenido cambia con regularidad. En este ejemplo la variable es \$nombre y la cadena de caracteres está entre comillas.

## 5. Comentarios

Los comentarios son muy útiles en PHP porque permiten ignorar el código que el servidor Web tiene que ejecutar. Solamente el usuario puede ver el texto de los comentarios, que sirve para explicar las líneas del código PHP.

Existen dos tipos de comentarios:

- Monolínea:

```
<?php

// comentario monolínea
// echo 'no veremos nada';

?>
```

- Multilínea:

```
<?php

/* comentario multilínea
   echo 'no veremos';
   echo 'nada';
*/
?>
```

En ninguno de los dos casos se ejecuta la instrucción **echo**.

# Las variables

## 1. Asignación

Una variable es una información que se almacena temporalmente en la memoria, es decir, es una zona de la memoria que almacena información en una página PHP y que se destruye automáticamente cuando la página ya no se ejecuta.

Una variable PHP comienza siempre con \$, seguida de una letra y de una secuencia de letras, cifras o del signo \_.

Por ejemplo, \$edad.

**Atención:** PHP distingue entre mayúsculas y minúsculas, por lo que \$nombre es distinto de \$Nombre.

Una variable siempre tiene un nombre y un valor.

Por ejemplo, \$edad = 25, el valor 25 se asigna a la variable \$edad gracias al signo =.

No es necesario definir y buscar el tipo de variable. Se hace automáticamente.

De este modo, puede escribir:

```
<?php  
  
$edad = 25; //variable de tipo numérico  
//después  
$edad = 'toto'; //variable de tipo texto  
  
?>
```

Esto no origina ningún error. El tipo de variable puede cambiar. En el ejemplo anterior cambia del tipo integer al tipo string (ver el siguiente párrafo).

## 2. Tipos de variables

Hay dos categorías de variables:

- Simples:
  - Los números enteros llamados **integer** son 1, 2, 3... y los números negativos, -1, -2, -3...
  - Los números decimales llamados **float** son los números positivos o negativos con comas (1.35665). **Atención:** el punto se utiliza como separador.
  - La cadena de caracteres **string**: cualquiera con dobles comillas ("hola") o comillas simples ('hola').
  - Los **booleanos**: solo tienen dos tipos de valores: verdadero o falso, clasificados como **true** o **false**.
- Compuesta:
  - Las tablas: permiten almacenar varias variables. Podrá consultarla en el capítulo Funciones y estructuras de control - Las tablas.

- Los objetos: son más complejos y se describen en el capítulo El objeto.

### 3. Particularidades del tipo de variable string

Si desea mostrar la frase «¿Qué pasa con un apóstrofo ' como este?», no podrá escribir:

```
<?php $variable1 = '¿Qué pasa con un apóstrofo ' como este?'; ?>
```

El apóstrofo en la instrucción **echo** genera un error.

Para solucionar este problema, hay dos soluciones:

- Usar comillas dobles en la instrucción **echo**:

```
<?php $variable1 = "¿Qué pasa con un apóstrofo ' como este?"; ?>
```

- Escapar en los apóstrofos con \:

```
<?php $variable1 = '¿Qué pasa con un apóstrofo \' como este?'; ?>
```

Cuando se añade la barra invertida delante del apóstrofo, se evita que este se interprete como el final de la instrucción **echo**.

De esta manera, en el segundo caso la cadena "¿Qué pasa con un apóstrofo ' como este?" estará en la variable **\$variable1**.

Lo mismo ocurre con las comillas:

```
<?php $variable1 = 'Mis "súper" amigos'; ?>
```

o

```
<?php $variable1 = "Mis \"súper\" amigos"; ?>
```

La barra invertida anula el comportamiento de cierre de instrucción del apóstrofo o de las comillas dobles. También puede servir para anular el comportamiento del \$.

Si quiere mostrar **\$variable1 = pepe**, escriba:

```
<?php  
  
$variable1 = 'pepe';  
echo "\$variable1 = $variable1";  
  
?>
```

Para terminar, puede acceder a un carácter de una cadena con la siguiente sintaxis:

`$var[x]` o `$var` es el nombre de la variable que contiene la cadena de caracteres y `x` es la posición del carácter que debe recuperar. Tenga en cuenta que para PHP el primer carácter está en la posición 0.

El siguiente ejemplo, primero muestra h y a continuación a.

```
<?php  
  
$variable1 = 'hola';  
echo $variable1[0];  
echo $variable1[3];  
  
?>
```

## 4. La concatenación

Es un conjunto de cadenas de caracteres. PHP permite la concatenación usando la coma o el punto.

```
<?php  
echo 'hola '.'lee esta ayuda';  
?>
```

Equivale a:

```
<?php  
echo 'hola ','lee esta ayuda';  
?>
```

Da como resultado:

Hola lee esta ayuda

Si quiere concatenar la cadena "hola" y "aquí hay un apóstrofo '", no podrá escribir:

```
<?php  
echo 'hola '.'aquí hay un apóstrofo '';  
?>
```

El apóstrofo cierra la cadena de caracteres que ha empezado en "aquí", por lo que PHP genera un error.

Para evitar este problema:

Inserta el carácter \ de escape en el apóstrofo:

```
<?php  
    echo 'Hola '.'aquí hay un apóstrofo \' ';  
?>
```

O bien abre y cierra con comillas dobles la cadena de caracteres:

```
<?php  
    echo "hola"."aquí hay un apóstrofo '\";  
?>
```

Si introduce en una variable la cadena "aquí hay un apóstrofo ":

```
<?php  
    $variable = "aquí hay un apóstrofo '\";  
    echo "hola".$variable;  
?>
```

Para mostrar una variable en una cadena de caracteres con ayuda de **echo**, escriba lo siguiente:

```
<?php  
    $variable = "aquí hay un apóstrofo '\";  
    echo "hola $variable";  
?>
```

PHP interpreta la variable y muestra: "Hola aquí hay un apóstrofo ", pero no "hola \$variable".

Si quiere mostrar "hola\$variable", no utilice \$:

```
<?php  
    $variable = "aquí hay un apóstrofo '\";  
    echo "hola \${$variable}";  
?>
```

Para terminar, y para una mayor claridad del código, utilizaremos la sintaxis:

```
<?php  
    $variable = "aquí hay un apóstrofe '\";  
    echo "hola \".$variable;  
?>
```

Esto permite mostrar en Notepad++ la \$variable en color azul, y por tanto resaltar las variables del texto.

## 5. Transtipado

Se utiliza para transformar un tipo de variable en otro.

Por ejemplo, si tiene un número decimal en una variable y desea transformarlo en un entero, solo debe convertirlo en entero. Esta conversión se denomina cast.

```
<?php  
$variable = 15.325;  
echo "El valor entero es:".(int)$variable;  
?>
```

Da como resultado:

El valor entero es:15

En este ejemplo la variable \$variable se transforma en tipo **int** y se concatena con la cadena de caracteres "El valor entero es:".

Por tanto, la sintaxis es:

(tipo)\$variable

Y el tipo es igual a:

- (int) o (integer) -> tipo entero
- (bool) o (boolean) -> booleano
- (double) o (float) o (real) -> tipo double
- (string) -> tipo cadena de caracteres
- (array) -> tipo array
- (object) -> tipo objeto

Puede añadir espacios entre los paréntesis, antes y después del tipo.

```
<?php  
$variable = 15.325;  
echo "El valor entero es:". ( integer )$variable;  
?>
```

Da como resultado:

El valor entero es:15

Otro ejemplo consiste en transformar un número entero en cadena de caracteres:

```
<?php  
$variable = 15.325;
```

```
echo "El valor del tipo de cadena de caracteres es:  
".(string)$variable;  
?>
```

Da como resultado:

El valor del tipo de cadena de caracteres es: 15.325

En el siguiente ejemplo, se transforma un número entero en booleano:

```
<?php  
$variable = 15.35;  
$varbool = (bool)$variable;  
echo "El tipo de variable \$varbool es:".gettype($varbool);?>
```

Da como resultado:

El tipo de variable \$varbool es: boolean

## Las constantes

Una constante permite definir un dato, dándole un valor permanente válido para el resto del programa.

Esta constante es muy útil si quiere utilizar regularmente un valor sin tener que escribirlo cada vez. Por ejemplo, si utiliza normalmente PI y no quiere escribir cada vez el mismo valor 3,1415926535 puede crear una constante llamada PI por valor de 3,1415926535 y así podrá utilizar PI en el código del programa.

Podemos definir una constante con la palabra clave **define**:

```
<?php
define('NOMBRE_DE_LA_CONSTANTE', 'valor de la constante');
?>
```

Para definir y mostrar PI:

```
<?php
define('PI', 3,1415926535);
echo PI
?>
```

Da como resultado:

3,1415926535

El nombre y la constante se escriben siempre en mayúsculas. El valor puede ser una cadena de caracteres, un valor numérico o un booleano.

No se puede volver a definir la constante. Conserva su valor permanentemente durante todo el programa, es decir, en la misma página PHP.

Para saber si una constante está definida, puede utilizar la función **defined**. Esta función devuelve true si la constante está definida, y false si no lo está.

```
<?php
define('PI', 3,1415926535);
$var_definida = defined('PI');
echo $var_definida;
?>
```

Da como resultado:

True

# Las condiciones

## 1. If

Las instrucciones del tipo condición permiten ejecutar el código si una condición es verdadera.

Por ejemplo, si el nombre es Roberto, muestre «Bienvenido»:

```
<?php  
$nombre = 'Roberto'; //declaración de la variable $nombre  
  
if ($nombre == 'Roberto') //comprueba la variable $nombre  
{  
    echo 'Bienvenido';  
}  
?>
```

El valor Roberto se asigna a la variable \$nombre y se comprueba con ayuda de la instrucción **if**.

Por tanto, la sintaxis es:

```
if (condition) { instrucción }
```

Tenga en cuenta que:

- «es igual a» se designa **==**
- «es diferente de» se designa **!=**
- «es inferior a» se designa **<**
- «es superior a» se designa **>**
- «es inferior o igual a» se designa **<=**
- «es superior o igual a» se designa **>=**

La instrucción **si no** se designa con **else**.

Por ejemplo, si el nombre es igual a Roberto, se muestra «Bienvenido», o «Hasta pronto» en cualquier otro caso:

```
<?php  
  
$nombre = 'Pepe'; //declaración de la variable $nombre  
  
if ($nombre == 'Roberto') //comprueba la variable $nombre  
{  
    echo 'Bienvenido';  
}  
else  
{  
    echo 'Hasta pronto';  
}
```

Da como resultado:

Hasta pronto

El código comprueba si la variable \$nombre es Roberto, pero como no es el caso, el código entra en **else** (si no), y ejecuta echo "Hasta pronto";.

La última instrucción en las condiciones es **else if**, llamada **si no si**.

Esto permite probar otras condiciones que no han sido comprobadas por **if**.

Por ejemplo, si el nombre es Roberto, se muestra «Bienvenido» ; si no, si es Pepe se muestra «Hola» ; si no, «Hasta pronto»:

```
<?php

$nombre = 'Pepe'; //declaración de la variable $nombre

if ($nombre == 'Roberto') //comprueba la variable $nombre
{
    echo 'Bienvenido';
}
else if ($nombre == 'Pepe') //comprueba la variable $nombre
{
    echo 'Hola';
}
else
{
    echo 'Hasta pronto';
}
?>
```

Da como resultado:

Hola

El código comprueba si \$nombre es igual a Roberto y también si es igual a Pepe.

Como efectivamente es igual a Pepe, el código ejecuta echo "Hola";.

Puede añadir tantas instrucciones **else if** como quiera.

Si la condición (\$nombre == "Roberto") es cierta, las otras condiciones **else if** ni siquiera se comprueban, por tanto si entra en **if** se asegura de no pasar nunca por **else if** ni **else**.

Por el contrario, si escribe:

```

<?php

$nombre = 'Roberto'; //declaración de la variable $nombre

if ($nombre == 'Roberto') //comprueba la variable $nombre
{
    echo 'Bienvenido';
}

if ($nombre == 'Pepe') //comprueba la variable $nombre
{
    echo 'Hola';
}
else
{
    echo 'Hasta pronto';
}
?>

```

El código comprueba si \$nombre es igual a Roberto y si también es igual a Pepe, pero como \$nombre se inicia con el valor Roberto, este código muestra «Bienvenido» y «Hasta pronto».

También puede anidar los **if** unos dentro de otros tantas veces como quiera.

Por ejemplo:

```

<?php

$edad = 30; //declaración de la variable $edad

if ($edad > 20) //comprueba la variable $edad
{
    if ($edad == 30) //comprueba la variable $edad
    {
        echo 'Bienvenido';
    }
    else {
        echo 'Hasta pronto';
    }
}
?>

```

Da como resultado:

Bienvenido

El programa comprueba si la variable \$edad es superior a 20, y, como es el caso, el programa comprueba si \$edad es igual a 30 y ejecuta echo "Bienvenido";.

Para terminar, puede intercalar el código HTML entre las condiciones en PHP.

Por ejemplo:

```
<?php  
  
$nombre = 'Roberto'; //declaración de la variable $nombre  
  
if ($nombre == 'Roberto') //comprueba la variable $nombre  
{  
?  
    Hola  <!--Código HTML-->  
<?php  
}  
else  
{  
?  
    Hasta pronto  <!--Código HTML-->  
<?php  
}  
?>
```

Es igual a:

```
<?php  
  
$nombre = 'Roberto'; //declaración de la variable $nombre  
  
if ($nombre == 'Roberto') //comprueba la variable $nombre  
{  
    echo 'Hola';  
}  
else  
{  
    echo 'Hasta pronto';  
}  
?>
```

## 2. Switch

La instrucción **switch** es igual a **if**, pero el programador la utiliza para comprender mejor el código. Si se utiliza un **break** para salir del **switch**, entonces es igual a **if**. De lo contrario, se ejecutarán todas las instrucciones del **case** en el que ha entrado.

La sintaxis es:

```
switch (condición) {  
    case expresión: instrucción  
    case expresión: instrucción  
    ...  
}
```

```

<?php

$nombre = 'Roberto'; //declaración de la variable $nombre

switch ($nombre) //comprueba la variable $nombre
{
    case 'Roberto':
        echo 'Hola';
        break;
    case 'Juan':
        echo 'Hasta pronto';
        break;
}
?>

```

La instrucción **break** provoca la salida del **switch** y si \$nombre es igual a "Roberto" el código ejecutará echo "Hola" y **break**, y saldrá del **switch** sin comprobar "Juan".

Eso es igual a la instrucción **else if** en lugar de **if**:

```

<?php

$nombre = 'Roberto'; //declaración de la variable $nombre

if ($nombre == 'Roberto') //comprueba la variable $nombre
{
    echo 'Hola';
}
else if ($nombre == 'Juan')
{
    echo 'Hasta pronto';
}
?>

```

En el siguiente ejemplo, si no utiliza la instrucción **break**:

```

<?php

$nombre = 'Roberto'; //declaración de la variable $nombre

switch ($nombre) //comprueba la variable $nombre
{
    case 'Roberto': echo 'Hola<br />';
    case 'Juan':    echo 'Hasta pronto';
}
?>

```

El valor Roberto se asigna a la variable \$nombre y esta variable se comprueba con ayuda de la instrucción **switch**. El ejemplo anterior muestra lo siguiente:

Hola

Hasta pronto

La instrucción **case** permite comparar cada valor con la variable \$nombre y ejecuta las instrucciones después de los **:**.

Para terminar, puede añadir la instrucción **default** para ejecutar el código en caso de no haber encontrado ninguna similitud con las instrucciones **case**.

Por ejemplo:

```
<?php

$nombre = 'Pepe'; //declaración de la variable $nombre

switch ($nombre) //comprueba la variable $nombre
{
    case 'Roberto':
        echo "Hola";
        break;
    case 'Juan':
        echo "Hasta pronto";
        break;
    default:
        echo "Nadie tiene este nombre";
}
?>
```

Da como resultado:

Nadie tiene este nombre

Como \$nombre es igual a "Pepe", el código no entra en el "Roberto" ni en el **case** "Juan", pero sí en **default**.

La instrucción **default** es igual a la instrucción **else**.

Vea el siguiente ejemplo con dígitos:

```
<?php

$edad = 25; //declaración de la variable $edad

switch ($edad) //comprueba la variable $edad
{
    case 20:
        echo "Tiene 25 años.";
        break;
    case 25:
        echo "Tiene 25 años.";
        break;
}
```

```
    default:  
        echo "No tiene 20 ni 25 años.";  
    }  
?>
```

Da como resultado:

Tiene 25 años.

En definitiva, si tiene que comprobar muchas condiciones, utilice **switch**.

# Los bucles

## 1. For

Un bucle permite repetir x veces la ejecución de un código.

Por ejemplo, si quiere mostrar diez veces «Hola», solo tiene que escribir el bucle **for**.

```
<?php  
  
for ($i = 1; $i <= 10; $i++)  
{  
    echo 'Hola <br />';  
}  
?>
```

La variable `$i` representa el contador del bucle. No está obligado a recurrir a la variable `$i`, pero por norma general es el nombre que se emplea.

Por tanto, la sintaxis es:

```
for ($i=número inicial; $i <= número final; aumento)  
{  
    instrucciones  
}
```

`$i++` es igual a `$i=$i+1` y representa el aumento de `$i`. Puede escribir `$i=$i+2` para aumentar o `$i=$i-1` para disminuir.

Por ejemplo, puede escribir los números de 100 a 150 con el siguiente código:

```
<?php  
  
for ($i = 100; $i <= 150; $i++)  
{  
    echo $i.'<br />';  
}  
?>
```

La instrucción `echo $i.'<br />'`; se repite 50 veces y `$i` aumenta en 1 cada vez.

`<br />` permite saltar una línea entre cada número para no tener que mostrarlos todos.

La instrucción **break** permite detener el bucle.

Por ejemplo, si quiere mostrar cinco veces «Hola», solo debe escribir un bucle **for**:

```
<?php
```

```

for ($i = 1; $i <= 10; $i++)
{
    echo 'Hola <br />';
    if ($i == 5) {
        break;
    }
}
?>

```

Da como resultado :

```

Hola
Hola
Hola
Hola
Hola

```

El bucle se detiene cuando **\$i** es igual a 5 (y no a 10).

## 2. While

El bucle **while** significa «mientras que», es decir, el bucle se ejecutará siempre y cuando una condición sea verdadera.

Por ejemplo, para mostrar diez veces «Hola», solo debe escribir un bucle **while**:

```

<?php
$i = 1;
while ($i <= 10)
{
    $i=$i+1;
    echo 'Hola <br />';
}
?>

```

La variable **\$i** representa el contador del bucle. Pero mientras **\$i** sea inferior o igual a 10, se repetirá el bucle.

Por lo tanto, la sintaxis es:

```

$i=número inicial
while ($i <= número final)
{
    aumento
    instrucciones
}

```

No olvide poner el aumento de \$i en las instrucciones de **while**; de lo contrario \$i nunca valdrá 10 y tendrá un bucle infinito.

Tenga en cuenta que el valor de salida de \$i se pone antes del bucle y que este valor debe respetar la condición del bucle (`$i <= número final`) para entrar en el bucle.

Si escribe:

```
<?php  
$i = 11;  
while ($i <= 10)  
{  
    $i=$i+1;  
    echo 'Hola <br />';  
}  
?>
```

Nunca pasará por el bucle porque \$i vale 11 en un principio y no cumple la condición del bucle.

El bucle **while** es igual al bucle **for**; en algunas ocasiones le resultará muy útil si desconoce el número de veces que va a ejecutar un bucle, sobre todo si va a leer el bucle **while** en la base de datos y la condición de salida del bucle depende del valor leído en la base de datos.

### 3. Do while

El bucle **Do while** significa «hacer mientras», es decir, el bucle se ejecutará siempre y cuando una condición sea verdadera. Se diferencia del bucle while en que la expresión se ejecuta al menos una vez.

Por ejemplo, para mostrar diez veces "Hola", debe escribir el bucle **Do while**:

```
<?php  
$i = 1;  
do  
{  
    $i=$i+1;  
    echo 'Hola <br />';  
} while ($i <= 10)  
?>
```

La variable \$i representa el contador del bucle. Pero esta vez debe leer: ejecutar el bucle si \$i es inferior o igual a 10.

Por tanto, la sintaxis es:

```
$i=número inicial  
do  
{  
    aumento  
    instrucciones  
} while ($i <= número final)
```

## 4. Foreach

Este bucle se describe en la sección siguiente, dedicada a las tablas.

# Las tablas

## 1. Tabla numérica

Una tabla es como una variable, pero puede almacenar varios valores.

Una tabla se define con una clave (llamada índice) y un valor:

Clave	Valor
1	Juan
2	Roberto
3	Pablo
4	Pedro
5	Alonso

Para crear esta tabla, escriba lo siguiente:

```
<?php  
$tabla = array('Juan', 'Roberto', 'Pablo', 'Pedro', 'Alonso');  
?>
```

Aquí la tabla se denomina \$tabla, pero la puede llamar de cualquier otra manera.

También puede acceder directamente al valor de una tabla a través de su índice, con la siguiente sintaxis:

```
<?php  
echo $tabla[0];  
?>
```

Da como resultado:

Juan

En efecto, \$tabla[x] es una variable que tiene como valor un elemento de la tabla.

**Atención:** los índices de las tablas comienzan desde 0.

Para sustituir la cadena de caracteres 'Roberto' por 'Nadia' en el índice 1 de la tabla, escriba:

```
<?php  
$tabla[1] = 'Nadia';  
?>
```

También puede crear una tabla vacía y rellenarla de la siguiente manera:

```
<?php
$tabla = array();
$tabla[0] = 'Juan';
$tabla[1] = 'Roberto';
$tabla[2] = 'Pablo';
$tabla[3] = 'Pedro';
$tabla[4] = 'Alonso';
?>
```

O bien de esta manera:

```
<?php
$tabla = array();
$tabla[] = 'Juan';
$tabla[] = 'Roberto';
$tabla[] = 'Pablo';
$tabla[] = 'Pedro';
$tabla[] = 'Alonso';
?>
```

O incluso:

```
< ?php
$tabla = [ 'Juan', 'Roberto', 'Pablo', 'Pedro', 'Alonso' ] ;
?>
```

PHP rellena automáticamente los índices, que es lo mismo que escribir:

```
<?php
$tabla = array('Juan', 'Roberto', 'Pablo', 'Pedro', 'Alonso');
?>
```

## 2. Tabla asociativa

En un tabla asociativa puede decidir la clave que va a introducir.

Por ejemplo:

Clave	Valor
A1	Juan
B4	Roberto
3	Pablo
PEPE	Pedro
H	Alonso

En este ejemplo la clave puede tomar cualquier valor; no tiene que ser necesariamente un número.

Este tipo de tabla se escribe de la siguiente manera:

```
<?php  
$tabla = array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'PEPE'=>'Pedro',  
'H'=>'Alonso');  
?>
```

La asociación se escribe con los símbolos =>. El primero es la clave y el segundo, el valor.

Otra forma de completar esta tabla es:

```
<?php  
$tabla = array();  
$tabla['A1'] = 'Juan';  
$tabla['B4'] = 'Roberto';  
$tabla[3] = 'Pablo';  
$tabla['Pepe'] = 'Pedro';  
$tabla['H'] = 'Alonso';  
?>
```

Puede utilizar esta tabla cuando la clave tiene una información importante.

Por ejemplo, si quiere almacenar las características de una persona en una tabla \$persona, escriba:

```
<?php  
$persona = array();  
$persona['Apellido'] = 'Martín';  
$persona['Nombre'] = 'Mónica';  
$persona['Edad'] = 50;  
?>
```

A continuación, si quiere mostrar la edad, escriba:

```
<?php  
echo $persona['Edad'];  
?>
```

### 3. Recorrido de una tabla

Existen varias soluciones para recorrer una tabla.

#### a. El bucle for

```

<?php
//creación de la tabla
$tabla = array('Juan','Roberto','Pablo','Pedro','Alonso');
//bucle en la tabla
for ($i = 0; $i < 5; $i++) {
    //mostrar los valores concatenados de la tabla
    //con un salto de línea
    echo $tabla[$i].'  


```

Este ejemplo muestra:

Juan  
Roberto  
Pablo  
Pedro  
Alonso

Por tanto, \$i varía de 0 a 4, el bucle muestra \$tabla[0], cuyo valor es 'Juan', \$tabla[1], cuyo valor es 'Roberto', etc.

Algunas veces los elementos de una base de datos se recuperan en una tabla y no se puede saber con antelación el número de elementos que contiene esta tabla. Si desconoce el tamaño de la tabla, puede usar las funciones **count()** o **sizeof()**, que se detallan más adelante.

Por ejemplo:

```

<?php
//creación de la tabla
$tabla = array('Juan','Roberto','Pablo','Pedro','Alonso');
//bucle en los registros de la tabla
for ($i = 0; $i < sizeof($tabla); $i++) {
    //muestra los valores concatenados de la tabla
    //con un salto de línea
    echo $tabla[$i].'  


```

Este ejemplo muestra lo siguiente:

Juan  
Roberto  
Pablo  
Pedro  
Alonso

## b. El bucle foreach

Este bucle es muy práctico porque no tiene que estar pendiente del tamaño de la tabla.

El bucle **foreach** no usa un contador. Almacena de uno en uno los valores de la tabla en una variable temporal, que en el siguiente ejemplo es \$val. Puede dar a esta variable el nombre que quiera y no necesita definirla.

```
<?php
//creación de la tabla
$tabla = array('Juan','Roberto','Pablo','Pedro','Alonso');
//bucle en la tabla
foreach ($tabla as $val) {
    //muestra los valores concatenados de la tabla
    //con un salto de línea
    echo $val.'<br />';
}
?>
```

Da como resultado:

Juan  
Roberto  
Pablo  
Pedro  
Alonso

Este bucle tiene otra ventaja: también permite mostrar la clave de la tabla:

```
<?php
//creación de la tabla
$tabla = array('A1'=>'Juan','B4'=>'Roberto',3=>'Pablo','Pepe'=>'Pedro',
'H'=>'Alonso');
//bucle en la tabla
foreach ($tabla as $clave => $val) {
    //muestra los valores concatenados de la tabla
    //con un salto de línea
    echo 'Clave:'.$clave.', valor:'.$val.'<br />';
}
?>
```

Da como resultado:

Clave:A1, valor:Juan  
Clave:B4, valor:Roberto  
Clave:3, valor:Pablo  
Clave:Pepe, valor:Pedro  
Clave:H, valor:Alonso

El bucle **foreach** se escribe con la variable \$clave (aunque le puede dar cualquier otro nombre), y contiene la clave de la tabla, así como el signo => y la variable \$val, que contiene el valor correspondiente a la clave.

También puede utilizar el bucle **foreach** sin la clave, como muestra el siguiente ejemplo:

```
<?php
//creación de la tabla
$tabla = array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',
'H'=>'Alonso');
//bucle en la tabla
foreach ($tabla as $val) {
    //muestra los valores concatenados de la tabla
    //con un salto de línea
    Echo 'valor:'.$val.'<br />';
}
?>
```

Da como resultado:

```
valor:Juan
valor:Roberto
valor:Pablo
valor:Pedro
valor:Alonso
```

### c. print\_r

Los programadores utilizan esta función para mostrar el contenido de la tabla sin cambiar el formato.

Ejemplo:

```
<?php
//creación de la tabla
$tabla = array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',
'H'=>'Alonso');
print_r($tabla);
?>
```

Da como resultado:

```
Array ( [A1] => Juan [B4] => Roberto [3] => Pablo [Pepe] => Pedro [H] => Alonso)
```

## 4. Funciones en las tablas

### a. Longitud de una tabla

La función **count()** o **sizeof()** permiten conocer el número de elementos de una tabla, por ejemplo:

```

<?php
//creación de la tabla
$tabla = array('A1'=>'Juan','B4'=>'Roberto',3=>'Pablo','Pepe'=>'Pedro',
'H'=>'Alonso');
$thisSize = count($tabla);
echo 'El tamaño de la tabla es:'. $thisSize;
?>

```

Da como resultado:

El tamaño de la tabla es: 5

La sintaxis es:

```
$numero_de_elementos = count($tabla);
```

con \$numero\_de\_elementos de tipo numérico

Esta función devuelve 0 si la tabla está vacía.

## b. Existencia de un valor en una tabla

La función **in\_array()** permite buscar un elemento en una tabla.

Por ejemplo:

```

<?php
//creación de la tabla
$tabla = array('A1'=>'Juan','B4'=>'Roberto',3=>'Pablo','Pepe'=>'Pedro',
'H'=>'Alonso');
if (in_array('Roberto',$tabla)) {
    echo 'Roberto está en la tabla';
}
?>

```

Da como resultado:

Roberto está en la tabla

Por tanto, su sintaxis es:

```
$presencia = in_array($valor_búsqueda,$tabla);
```

con \$presencia de tipo booleano.

Esta función puede tomar un tercer argumento, que consiste en comprobar el tipo de valor que se encuentra con respecto al valor deseado. Este argumento es una variable booleana que tiene un valor falso por defecto, es decir,

no tiene en cuenta el tipo.

Explicación: una tabla tiene el valor '33' pero se está buscando la cifra 33. No son del mismo tipo, ya que '33' es del tipo string (cadena de caracteres) y 33 es de tipo numérico.

Ejemplo:

```
<?php
//creación de la tabla
$tabla = array('10','33','33','78');
if (in_array(33,$tabla, true)) { // añadir true para tener en cuenta
    // el tipo
    echo '33 está en la tabla';
} else {
    echo "33 no está en la tabla";
}
?>
```

Da como resultado:

33 no está en la tabla

La función busca en la tabla el número 33, pero solamente existe la cadena de caracteres '33'.

La sintaxis con el tipo opcional es:

```
$presencia = in_array($valor_buscado,$tabla,$tipo_equivalente);
```

con \$presencia y \$tipo\_equivalente de tipo booleano.

### c. Existencia de una clave en una tabla

La función **array\_key\_exists()** permite conocer la existencia de una clave en una tabla.

Por ejemplo:

```
<?php
//creación de la tabla
$tabla = array('A1'=>'Juan','B4'=>'Roberto',3=>'Pablo',
'Pepe'=>'Pedro','H'=>'Alonso');
if (array_key_exists(3,$tabla)) {
    echo 'La clave 3 está en la tabla';
}
?>
```

Da como resultado:

La clave 3 está en la tabla

La sintaxis es:

```
$presencia = array_key_exists($Clave_buscada,$tabla);
```

con \$presencia de tipo booleano.

Otro ejemplo:

```
<?php  
//creación de la tabla  
$tabla = array('Juan','Roberto','Pablo','Pedro','Alonso');  
if (array_key_exists(4,$tabla)) {  
    echo 'La Clave 4 está en la tabla, su valor es:'. $tabla[4];  
}  
?>
```

Da como resultado:

La Clave 4 está en la tabla. Su valor es: Alonso

#### d. Ordenar una tabla

Existen varias funciones que permiten ordenar una tabla. Algunas ordenan en orden descendente, otras siguiendo la clave y no el valor, etc.

- **sort()**: ordena los valores de menor a mayor.

Por ejemplo:

```
<?php  
//creación de la tabla  
$tabla =  
array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',  
'H'=>'Alonso');  
sort($tabla); //ordena la tabla  
foreach ($tabla as $clave=>$valor) {  
    echo 'Clave:'.$clave.', valor:'. $valor.'<br />';  
}  
?>
```

Da como resultado:

Clave:0, valor:Alonso

Clave:1, valor:Juan

Clave:2, valor:Pedro

Clave:3, valor:Pablo

Clave:4, valor:Roberto

-  La función **sort()** pierde la clave de origen. Ya no tiene la clave 'H' para 'Alonso', sino 0. Las claves de origen se sustituyen por un número creciente, comenzando desde 0.

- **asort()**: ordena los valores de menor a mayor, conservando la pareja clave/valor.

Por ejemplo:

```
<?php
//creación de la tabla
$tabla =
array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',
'H'=>'Alonso');
asort($tabla); //ordena la tabla conservando la clave
foreach ($tabla as $clave=>$valor) {
    echo 'Clave:'.$clave.', valor:'.$valor.'<br />';
}
?>
```

Da como resultado:

```
Clave:H, valor:Alonso
Clave:A1, valor:Juan
Clave:Pepe, valor:Pedro
Clave:3, valor:Pablo
Clave:B4, valor:Roberto
```

Esta vez las claves se han conservado correctamente.

- **rsort()**: ordena los valores de mayor a menor.

Por ejemplo:

```
<?php
//creación de la tabla
$tabla =
array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',
'H'=>'Alonso');
rsort($tabla); //ordena la tabla del valor más grande al más pequeño
foreach ($tabla as $clave=>$valor) {
    echo 'Clave:'.$clave.', valor:'.$valor.'<br />';
}
?>
```

Da como resultado:

```
Clave:0, valor:Roberto
```

```
Clave:1, valor:Pablo  
Clave:2, valor:Pedro  
Clave:3, valor:Juan  
Clave:4, valor:Alonso
```

- **arsort()**: ordena los valores de mayor a menor, conservando la pareja clave/valor.

Por ejemplo:

```
<?php  
//creación de la tabla  
$tabla =  
array('A1'=>'Juan', 'B4'=>'Roberto', '3'=>'Pablo', 'Pepe'=>'Pedro',  
'H'=>'Alonso');  
arsort($tabla); //ordena la tabla conservando la clave  
foreach ($tabla as $clave=>$valor) {  
    echo 'Clave:'.$clave.', valor:'.$valor.'<br />';  
}  
?>
```

Da como resultado:

```
Clave:B4, valor:Roberto  
Clave:3, valor:Pablo  
Clave:Pepe, valor:Pedro  
Clave:A1, valor:Juan  
Clave:H, valor:Alonso
```

- **ksort()**: ordena las claves de la tabla de la más pequeña a la más grande, conservando la pareja clave/valor.

Por ejemplo:

```
<?php  
//creación de la tabla  
$tabla =  
array('A1'=>'Juan', 'B4'=>'Roberto', '3'=>'Pablo', 'Pepe'=>'Pedro',  
'H'=>'Alonso');  
ksort($tabla); //ordenar la tabla siguiendo la clave  
foreach ($tabla as $clave=>$valor) {  
    echo 'Clave:'.$clave.', valor:'.$valor.'<br />';  
}  
?>
```

Da como resultado:

```
Clave:A1, valor:Juan
```

```
Clave:B4, valor:Roberto  
Clave:H, valor:Alonso  
Clave:Pepe, valor:Pedro  
Clave:3, valor:Pablo
```

- **krsort()**: ordena las claves de la tabla de mayor a menor, conservando la pareja clave/valor.

Por ejemplo:

```
<?php  
//creación de la tabla  
$tabla =  
array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',  
'H'=>'Alonso');  
krsort($tabla); //ordenar la tabla de manera descendente siguiendo  
//la clave  
foreach ($tabla as $clave=>$valor) {  
    echo 'Clave:'.$clave.', valor:'.$valor.'<br />';  
}  
?>
```

Da como resultado:

```
Clave:3, valor:Pablo  
Clave:Pepe, valor:Pedro  
Clave:H, valor:Alonso  
Clave:B4, valor:Roberto  
Clave:A1, valor:Juan
```

## e. Búsqueda en una tabla

La función **array\_search()** equivale a **in\_array()**. Permite buscar un elemento en una tabla, pero vuelve a enviar la clave del elemento que se ha buscado.

Por ejemplo:

```
<?php  
//creación de la tabla  
$tabla = array('A1'=>'Juan', 'B4'=>'Roberto', 3=>'Pablo', 'Pepe'=>'Pedro',  
'H'=>'Alonso');  
$clave_elemento = array_search('Roberto',$tabla);  
echo "La clave del elemento buscado es:".$clave_elemento;  
?>
```

Da como resultado:

La clave del elemento buscado es: B4

La sintaxis es:

```
$clave = array_search($valor_buscado,$tabla);
```

Otro ejemplo:

```
<?php  
//creación de la tabla  
$tabla = array('Juan','Roberto','Pablo','Pedro','Alonso');  
$clave_elemento = array_search('Roberto',$tabla);  
echo "La clave del elemento buscado es:".$clave_elemento;  
  
?>
```

Da como resultado:

La clave del elemento buscado es:1

La clave del elemento 'Roberto' es 1, porque la clave de la tabla empieza por 0.

#### f. Recorrer una cadena de caracteres en una tabla

La función **explode( )** permite insertar una cadena en una tabla, utilizando un separador.

Por ejemplo:

```
<?php  
$conjunto = "1;2;3;4;5";  
$tabla = explode(";", $conjunto);  
echo "El primer valor de la tabla es:".$tabla[0]."<br />";  
echo "El segundo valor de la tabla es:".$tabla[1];  
?>
```

Da como resultado:

El primer valor de la tabla es:1

El segundo valor de la tabla es:2

La sintaxis es:

```
$tabla_recortada = explode($separador,$cadena_a_recortar);
```

Otro ejemplo:

```
<?php
```

```
$conjunto = "1-2-3-4-5";
$tabla = explode("-", $conjunto);
echo "Los valores de la tabla son:";
foreach ($tabla as $valor) {
    echo $valor . ";";
}
?>
```

Da como resultado:

Los valores de la tabla son:1;2;3;4;5;

### **g. Reagrupar los valores de una tabla en una cadena**

La función **implode()** permite reagrupar los valores de una tabla en una cadena, utilizando un separador.

Por ejemplo:

```
<?php
$tabla = array("Juan", "Roberto", "Pablo");
$nombres = implode(";", $tabla);
echo "Los nombres son:".$nombres;
?>
```

Da como resultado:

Los nombres son:Juan;Roberto;Pablo

La sintaxis es:

```
$cadena = implode($separador, $tabla);
```

Otro ejemplo:

```
<?php
$tabla = array(5=>"Juan", 3=>"Roberto", 2=>"Pablo");
$nombres = implode("-", $tabla);
echo "Los nombres son:".$nombres;
?>
```

Da como resultado:

Los nombres son:Juan-Roberto-Pablo

Como puede observar, las claves de una tabla no tienen ningún efecto en la función **implode()**.

## **h. Trocear una cadena con una longitud fija**

La función **str\_split()** permite trocear en la tabla una cadena de longitud fija, utilizando un parámetro que indica el tamaño de los trozos de la cadena.

Por ejemplo:

```
<?php  
$conjunto="1;2;3;4;5";  
$tabla=str_split($conjunto,2);  
echo "Los elementos son:";  
foreach ($tabla as $valor) {  
    echo $valor." ";  
}  
?>
```

Da como resultado:

Los elementos son:1;2;3;4;5;

Por tanto, su sintaxis es:

```
$tabla=str_split($cadena,$longitud);
```

Otro ejemplo:

```
<?php  
$conjunto="11=12=13=14=15";  
$tabla=str_split($conjunto,3);  
echo "Los elementos son:";  
foreach ($tabla as $valor) {  
    echo $valor."<br/>";  
}  
?>
```

Da como resultado:

Los elementos son:11=  
12=  
13=  
14=  
15=

## **i. Añadir elementos al final de la tabla**

La función **array\_push()** permite añadir uno varios elementos al final de la tabla.

Por ejemplo:

```
<?php  
$tabla = array("Juan", "Roberto", "Pablo");  
array_push($tabla, "Pedro", "Alonso");  
echo $tabla[4];  
?>
```

Da como resultado:

Alonso

Por tanto, su sintaxis es:

```
array_push($tabla,$valor1,$valor2,...);
```

Otro ejemplo:

```
<?php  
$tabla = array();  
for ($i=0;$i<=10;$i++) {  
    array_push($tabla,$i);  
    echo $tabla[$i].";";  
}  
?>
```

Da como resultado:

0;1;2;3;4;5;6;7;8;9;10;

#### j. Eliminar un elemento al final de la tabla

La función **array\_pop()** permite eliminar un elemento al final de la tabla y devolver su valor.

Por ejemplo:

```
<?php  
$tabla = array("Juan", "Roberto", "Pablo");  
$nombre = array_pop($tabla);  
echo "El nombre eliminado es:".$nombre;  
?>
```

Da como resultado:

El nombre eliminado es:Pablo

Por tanto, su sintaxis es:

```
$valor_eliminado=array_pop($tabla);
```

Otro ejemplo:

```
<?php

$tabla = array("Juan", "Roberto", "Pablo");
echo "Antes de la eliminación, el número de elementos en la tabla
es:".count($tabla)."  
";
$numero_elementos = count($tabla);
for ($i=0;$i<$numero_elementos;$i++) {
    $nombre = array_pop($tabla);
}
echo "El último nombre eliminado es:".$nombre."  
";
echo "Después de la eliminación, el número de elementos de la tabla
es:".count($tabla);

?>
```

Da como resultado:

Antes de la eliminación, el número de elementos en la tabla es:3

El último nombre eliminado es:Juan

Después de la eliminación, el número de elementos de la tabla es:0

## k. Selección de un elemento de la tabla de forma aleatoria

La función **array\_rand()** permite seleccionar uno a varios elementos de una tabla de forma aleatoria y devolver las claves correspondientes.

Por ejemplo:

```
<?php

$tabla = array("Juan", "Roberto", "Pablo");
$clave_aleatoria = array_rand($tabla);
echo "El nombre seleccionado al azar es:".$tabla[$clave_aleatoria];

?>
```

Da como resultado:

El nombre seleccionado al azar es:Roberto (o Juan o Pablo)

Por tanto, su sintaxis es:

```
$clave_seleccionada=array_rand($tabla, $numero_seleccionado);
```

El parámetro \$numero\_seleccionado es opcional, por defecto es igual a 1. Corresponde al número de valores que se toman al azar. El siguiente ejemplo utiliza:

```
<?php  
  
$tabla = array("Juan", "Roberto", "Pablo");  
$tabla_clave_aleatoria = array_rand($tabla,2);  
echo "El primer número seleccionado al azar es:". "  
$tabla[$tabla_clave_aleatoria[0]]."<br />";  
echo "El segundo número seleccionado al azar es:". "  
$tabla[$tabla_clave_aleatoria[1]]";  
  
?>
```

La \$tabla\_clave\_aleatoria es una tabla que contiene las claves de los elementos seleccionados al azar. Si \$tabla\_clave\_aleatoria[0] contiene la clave del primer elemento seleccionado al azar, su valor se obtiene por \$tabla[\$tabla\_clave\_aleatoria[0]].

El número seleccionado no debe ser igual o superior al número de elementos de la tabla.

## 5. Tabla de varias dimensiones

Una tabla de varias dimensiones es una tabla dentro otra. Puede ser muy útil si quiere almacenar información relacionada entre las tablas.

Por ejemplo, si quiere crear esta tabla:

Clave	Valor	
Pablo	Clave	Valor
	Nombre	PABLO
	Profesión	ministro
	edad	50
Roberto	Clave	Valor
	NOMBRE	ROBERTO
	Profesión	agricultor
	edad	45

Puede hacerlo de dos maneras.

### a. Por medio de una tabla temporal

```
<?php  
  
$tab_caracteristica_PABLO = array("nombre" => "PABLO", "profesión" =>  
"ministro", "edad" => 50);  
$tab_caracteristica_ROBERTO= array("nombre" => "ROBERTO", "profesión" =>  
"agricultor", "edad" => 45);  
$tab_persona['PABLO'] = $tab_caracteristica_PABLO;  
$tab_persona['ROBERTO'] = $tab_caracteristica_ROBERTO;  
  
echo $tab_persona['ROBERTO']['profesión'];
```

?>

Da como resultado:

agricultor

La sintaxis para acceder al valor de un elemento de la tabla es:

```
$elemento = $tabla['Clave1']['Clave2']
```

La clave1 representa la clave de la tabla general y la clave2 representa la clave de la subtabla.

### **b. Almacenar directamente los valores en la tabla general**

```
<?php  
  
$tab_persona['ESTEFANIA']['nombre'] = "PABLO";  
$tab_persona['ESTEFANIA']['profesión'] = "ministro";  
$tab_persona['ESTEFANIA']['edad'] = 50;  
  
$tab_persona['LUNA']['nombre'] = "ROBERTO";  
$tab_persona['LUNA']['profesión'] = "agricultor";  
$tab_persona['LUNA']['edad'] = 45;  
  
echo $tab_persona['ESTEFANIA']['nombre'];  
  
?>
```

Da como resultado:

PABLO

La manera de crear una tabla no influye en su uso. Por tanto, la sintaxis para acceder al valor de un elemento es la misma.

## **6. Ejercicios**

### **a. Enunciados**

#### **Ejercicio 1 (fácil)**

Cree una tabla que contenga los números del 1 al 10 y otra tabla con los números del 11 al 20. A continuación, cree otra tabla con la suma de las dos primeras tablas y muestre sus valores. Para crear estas tablas, utilice los bucles.

#### **Ejercicio 2 (fácil)**

Cree una tabla con 10 valores al azar entre 1 y 100. La función rand(\$min,\$max); permite obtener un número al azar entre \$min y \$max. Ordene esta tabla del valor mayor al menor, introduzca todos los valores en una cadena de caracteres separados por ; y muestre la cadena.

### **Ejercicio 3 (difícil)**

En la sección Tabla con varias dimensiones, hemos visto el código que se utiliza para crear esta tabla multidimensional:

Clave	Valor	
Pablo	Clave	Valor
	Nombre	PABLO
	Profesión	ministro
	edad	50
Roberto	Clave	Valor
	Nombre	ROBERTO
	Profesión	agricultor
	edad	45

Cree el código PHP que permita generar esta tabla en HTML con ayuda de los bucles.

### **Ejercicio 4 (fácil)**

Observe estas dos tablas:

La tabla1 se compone de los siguientes elementos: 6,25,35 y 61

La tabla2 se compone de los siguientes elementos: 12,24 y 46

Escriba el código con el valor S que permita calcular un valor representativo de estas dos tablas. El valor S se calcula multiplicando cada valor de la tabla1 por el valor de la tabla2 y sumando el total.

En este ejemplo, el valor S será igual a:

12\*6+12\*25+12\*35+12\*61+24\*6+24\*25+24\*35+24\*61+46\*6+46\*25+ 46\*35+46\*61

Sin duda alguna, para hacer este ejercicio debe utilizar los bucles.

### **Ejercicio 5 (dificultad media)**

Cuando se acceda a su página PHP, muestre los banners de forma aleatoria entre tres opciones. Los elementos de esos banners son una imagen, un enlace y una descripción de esta imagen. Estos elementos se almacenan en una tabla.

#### **b. Soluciones**

##### **Solución del ejercicio 1**

```
<?php  
  
$tabla1 = array();
```

```

$tabla2 = array();
$tablaSuma = array();

for ($i=1;$i<=10;$i++) { //tabla de 1 a 10
    $tabla1[$i]=$i;
}
for ($i=1;$i<=10;$i++) { //tabla de 11 a 20
    $tabla2[$i]=$i+10;
}
for ($i=1;$i<=10;$i++) { //tabla con la misma suma que las otras 2 tablas
    $tablaSuma[$i]=$tabla1[$i]+$tabla2[$i];
}

//Muestra los valores de la suma de las otras 2 tablas
for ($i=1;$i<=10;$i++) {
    echo $tablaSuma[$i]."<br />";
}

?>

```

### **Solución del ejercicio 2**

```

<?php

$tabla1 = array();

for ($i=1;$i<=10;$i++) { //tabla de 1 a 10
    $tabla1[$i] = rand(1,100);
}
sort($tabla1);
$valores = implode(";", $tabla1);
echo "Los valores son:".$valores;

?>

```

### **Solución del ejercicio 3**

```

<?php

$tab_caracteristica_ESTEFANIA = array( "nombre" => "PABLO", "profesión" =>
"ministro", "edad" => 50);
$tab_caracteristica_LUNA= array( "nombre" => "ROBERTO", "profesión" =>
"agricultor", "edad" => 45);
$tab_persona['ESTEFANIA'] = $tab_caracteristica_ESTEFANIA;
$tab_persona['LUNA'] = $tab_caracteristica_LUNA;
echo '<table border="1">';
echo '<tr><td>';
echo 'Clave';
echo '</td><td colspan="2">';
echo 'Valor';
echo '</td></tr>';

```

```

foreach ($tab_persona as $clave => $val) {
    // $clave corresponde a ESTEFANIA o LUNA
    echo '<tr><td rowspan="4">';
    echo $clave;
    echo '</td>';
    echo '<td>Clave</td><td>Valor</td></tr>';
// bucle en la tabla $val corresponde a $tab_caracteristica_ESTEFANIA o
// $tab_caracteristica_LUNA
    foreach ($val as $clave2 => $val2) {
        echo '<tr>';
        echo '<td>' . $clave2 . '</td>';
        echo '<td>' . $val2 . '</td>';
        echo '</tr>';
    }
}
echo '</tabla>';
?>

```

#### **Solución del ejercicio 4**

```

<?php

$tabla1=array(6,25,35,61);
$tabla2=array(12,24,46);
$suma = 0; // inicialización de la variable $suma

$nb_tabla1=count($tabla1); // tamaño de la tabla1
$nb_tabla2=count($tabla2); // tamaño de la tabla2

for ($i=0;$i<=$nb_tabla1-1;$i++) {
    for ($j=0;$j<=$nb_tabla2-1;$j++) {
        $suma = $suma + $tabla1[$i] * $tabla2[$j];
    }
}
echo "El valor S es: ".$suma;

?>

```

#### **Solución del ejercicio 5**

```

<?php

// creación de tablas que contienen la imagen de los banners
// y su descripción
$tabBanners = array(
1 => array('http://www.su_sitioweb.com' , 'http://www.su_sitioweb.com/banner.gif' , 'Descripción 1'),
2 => array('http://www.su_sitioweb.com2' , 'http://www.su_sitioweb2.com/banner.gif' , 'Descripción 2'),
3 => array('http://www.su_sitioweb3.com' , 'http://www.su_sitioweb3.com/banner.gif' , 'Descripción 3')
)

```

```
su_sitioweb3.com/banner.gif','Descripción 3'));  
  
//elección aleatoria del banner  
$opcion = array_rand($tabBanners, 1);  
  
//muestra los banners  
echo '<a href="', $tabBanners[$opcion][0] ,'" title="',  
$tabBanners[$opcion][2] ,'">';  
echo '';  
echo '</a>';  
  
?>
```

# Procesamiento de las cadenas de caracteres

## 1. Funciones de manipulación de cadenas

En esta parte del capítulo se tratan todas las funciones PHP que permiten manipular las cadenas de caracteres.

### a. **strlen()**

La función **strlen()** devuelve la longitud de una cadena de caracteres.

Por ejemplo:

```
<?php  
  
$nombre = "Roberto";  
$longitud = strlen($nombre);  
echo "La longitud de la cadena es:".$longitud;  
  
?>
```

Da como resultado:

La longitud de la cadena es:7

Por tanto, su sintaxis es la siguiente:

```
$longitud= strlen($cadena);
```

Otro ejemplo:

```
<?php  
  
$nombre = " Hola, Roberto ";  
$longitud = strlen($nombre);  
echo "La longitud de la cadena es:".$longitud;  
  
?>
```

Da como resultado:

La longitud de la cadena es:15

También se cuentan los espacios.

### b. **substr()**

La función **substr()** devuelve un trozo de la cadena partiendo de una posición y con una longitud dada.

Por ejemplo:

```
<?php  
  
$nombre = "Roberto";  
$trozo = substr($nombre,2,3);  
echo "El trozo de la cadena es:". $trozo;  
  
?>
```

Da como resultado:

El trozo de la cadena es:ber

 La posición del inicio comienza desde 0.

No necesita indicar la longitud. La función devuelve los caracteres hasta el final de la cadena de caracteres.

Por tanto, su sintaxis es:

```
$trozo_cadena =  
substr($cadena,$posicion_inicio,longitud_cadena);
```

Otro ejemplo:

```
<?php  
  
$nombre = "Hola, me llamo Roberto";  
$trozo = substr($nombre,15);  
echo "El trozo de la cadena es:". $trozo;  
  
?>
```

Da como resultado:

El trozo de la cadena es: Roberto

Al no especificar el último parámetro, la función toma la cadena de caracteres hasta el final.

### c. **strstr()**

La función **strstr()** devuelve un trozo de la cadena desde un carácter hasta el final de la cadena.

Por ejemplo:

```
<?php
```

```
<?php  
$email = "Roberto.lopez@españa.es";  
$trozo = strstr($email,'@');  
echo "El trozo de la cadena es:". $trozo;  
?>
```

Da como resultado:

El trozo de la cadena es:@españa.es

La función devuelve la cadena de caracteres que va del carácter @ hasta el final de la cadena.

Por tanto, su sintaxis es:

```
$trozo_cadena = strstr($cadena,$caracter_busqueda);
```

La función devuelve false si no se encuentra ninguna cadena de caracteres.

Otro ejemplo:

```
<?php  
  
$email = "Roberto.lopez@españa.es";  
$trozo = strstr($email,'.');//busca punto  
echo "El trozo de la cadena es:". $trozo;  
  
?>
```

Da como resultado:

El trozo de la cadena es:.lopez@españa.es

La función devuelve un trozo de cadena desde el primer carácter encontrado.

#### d. str\_replace()

La función **str\_replace()** permite sustituir, dentro de la cadena de caracteres principal, un trozo de una cadena por otra.

Por ejemplo:

```
<?php  
  
$email = "Roberto.lopez@españa.es";  
$nuevo_nombre = str_replace('españa','mexico',$email);  
echo "El nuevo nombre de la cadena es:".$nuevo_nombre;  
  
?>
```

Da como resultado:

El nuevo nombre de la cadena es:Roberto.lopez@mexico.es

La sintaxis es:

```
$nueva_cadena = str_replace($cadena_buscada,  
$cadena_que_sustituye ,$cadena_principal);
```

En el siguiente ejemplo, puede poner una tabla en lugar de la **\$cadena\_buscada**:

```
<?php  
  
$tabla_cadena_buscada =  
array ("a","e","i","o","u","Y","A","E","I","O","U","Y");  
$email = "Roberto.lopez@españa.es";  
$nuevo_nombre = str_replace($tabla_cadena_buscada,'',$email);  
echo "El nuevo nombre de la cadena quitando todas las vocales  
es:".$nuevo_nombre;  
  
?>
```

Da como resultado:

El nuevo nombre de la cadena quitando todas las vocales es: Rbrt.lpz@spñ.s

En este ejemplo, la función **str\_replace** sustituye todas las vocales de la tabla por una cadena vacía.

### e. trim()

La función **trim()** permite eliminar los espacios al principio y al final de la cadena.

Por ejemplo:

```
<?php  
  
$email = " Roberto.lopez@españa.es " ;  
$longitud_nombre = strlen($email);//longitud de la cadena $nombre:25  
$nuevo_nombre = trim($email);//suprimir los espacios  
$longitud_nuevo_nombre = strlen($nuevo_nombre);  
//longitud de la cadena $nombre:23  
echo "El nuevo nombre de la cadena es:".$nuevo_nombre." con  
".$longitud_nuevo_nombre." caracteres";  
  
?>
```

Da como resultado:

El nuevo nombre de la cadena es:Roberto.lopez@españa.es con 23 caracteres

Por tanto, su sintaxis es:

```
$nueva_cadena = trim($cadena);
```

Esta función elimina los espacios, las tabulaciones y los saltos de línea.

#### f. **strtolower()**

La función **strtolower()** permite convertir una cadena en minúsculas.

Por ejemplo:

```
<?php  
  
$nombre = "ROBERTO";  
$nombre = strtolower($nombre);  
echo "El nombre de la cadena en minúsculas es:". $nombre;  
  
?>
```

Da como resultado:

El nombre de la cadena en minúsculas es:roberto

Por tanto, su sintaxis es:

```
$cadena_minusculas = strtolower($cadena);
```

#### g. **strtoupper()**

La función **strtoupper()** permite convertir una cadena en mayúsculas.

Por ejemplo:

```
<?php  
  
$nombre = "Juan";  
$nombre = strtoupper($nombre);  
echo "El nombre de la cadena en mayúsculas es:". $nombre;  
  
?>
```

Da como resultado:

El nombre de la cadena en mayúsculas es:JUAN

Por tanto, su sintaxis es:

```
$cadena_mayusculas = strtoupper($cadena);
```

- La función **ucfirst()** pone el primer carácter en mayúsculas. La función **ucwords()** pone la primera letra de cada palabra en mayúsculas.

## h. strpos()

La función **strpos()** devuelve la posición de la primera aparición en una cadena de caracteres.

Por ejemplo:

```
<?php  
  
$email = "Juan.lopez@españa.es";  
$posicion = strpos($email,'@');  
echo "La posición de @ es:". $posicion;  
  
?>
```

Da como resultado:

La posición de @ es:10

Por tanto, su sintaxis es:

```
$posicion = strpos($cadena,$ocurrencia_buscada);
```

- La posición empieza desde 0. Así, la posición de j en la cadena \$email es 0.

También hay otras funciones parecidas, que son:

- **strrpos()**: devuelve la posición de la última aparición en una cadena de caracteres.
- **stripos()**: devuelve la posición de la primera aparición en una cadena de caracteres sin tener en cuenta las mayúsculas y minúsculas.

Otro ejemplo:

```
<?php  
  
$email = "Juan.lopez@españa.es";  
$posicion = strrpos($email,'ñ');  
echo "La última posición de la letra ñ es:". $posicion;  
  
?>
```

Da como resultado:

La última posición de la letra ñ es:15

### i. str\_word\_count()

La función **str\_word\_count()** devuelve el número de palabras que están dentro de la cadena de caracteres.

Por ejemplo:

```
<?php  
  
$frase = "Hola, hace buen tiempo";  
$numero = str_word_count($frase);  
echo "El número de palabras en la cadena es:". $numero;  
  
?>
```

Da como resultado:

El número de palabras en la cadena es:4

Por tanto, su sintaxis es:

```
$posicion = str_word_count($cadena);
```

Esta función puede utilizar un argumento opcional, que es el formato. Si vale 0, la función devuelve el número de palabras como antes. Si vale 1, la función devuelve una tabla que contiene las palabras de la cadena de caracteres.

Por ejemplo:

```
<?php  
  
$frase = "Hola, hace buen tiempo";  
$tabla = str_word_count($frase,1);  
print_r($tabla);  
  
?>
```

Da como resultado:

Array ( [0] => Hola [1] => hace [2] => buen [3] => tiempo)

Si este parámetro vale 2, la función devuelve una tabla que contiene las palabras de la cadena de caracteres y la posición de la primera letra de la palabra clave.

Por ejemplo:

```
<?php  
  
$frase = "Hola, hace buen tiempo";  
$tabla = str_word_count($frase,2);  
print_r($tabla);  
  
?>
```

Da como resultado:

```
Array ( [0] => Hola [6] => hace [11] => buen [16] => tiempo )
```

En esta función, la noción de la palabra depende de la localización actual. Así, la coma no se considera como una palabra.

### j. str\_pad()

La función **str\_pad()** permite completar una cadena hasta un tamaño dado.

Por ejemplo:

```
<?php  
$cadena_inicio = "Hola";  
echo str_pad($cadena_inicio, 10, '!');  
?>
```

Da como resultado:

```
Hola!!!!
```

La función toma la cadena de origen "Hola" y la completa con la cadena "!" hasta un total de 10 caracteres.

Por tanto, su sintaxis es:

```
str_pad($cadena_origen, $numero_carácter_total,  
$cadena_para_completar);
```

El código que muestra 10 espacios en HTML, es:

```
<?php  
echo str_pad(' ', 60, '&nbsp;');  
?>
```

## 2. Las expresiones regulares

Las expresiones regulares permiten realizar búsquedas o sustituciones muy complejas en una cadena de caracteres.

Por ejemplo, si quiere saber si un correo electrónico contiene el carácter @ y el carácter . o si quiere cambiar el formato de una fecha del inglés al español, el uso de expresiones regulares permiten hacerlo en un solo paso.

En este soporte, utilizaremos PCRE (*Perl-Compatible Regular Expression*), que usa las funciones de expresiones regulares más rápidas. También disponemos de POSIX (*Portable Operating System Interface*), donde las funciones comienzan por ereg, pero hoy en día están obsoletas. La codificación debe hacerse en ANSI en Notepad++, para que los siguientes ejemplos funcionen correctamente.

- La función **preg\_match()** devuelve verdad si el valor que se busca está en la cadena de caracteres.

Por ejemplo:

```
<?php

if (preg_match("/web/","El webdesigner crea un sitio web."))
{
    echo "Se ha encontrado la cadena web.";
}
else {
    echo "La cadena Web no se ha encontrado.";
}

?>
```

Da como resultado:

Se ha encontrado la cadena web.

Por tanto, su sintaxis es:

```
$existe = preg_match ($pattern,$cadena);
```

con \$existe de tipo booleano.

\$pattern es una cadena que indica a la función **preg\_match()** cómo debe realizar la búsqueda. Esta cadena empieza y termina con un delimitador que suele ser el símbolo /. Aunque también podrá ver el símbolo #.

En el siguiente ejemplo, la función **preg\_match()** comprueba si la cadena "El webdesigner crea un sitio Web." contiene la cadena "Web".

Esta función tiene en cuenta las mayúsculas y las minúsculas.

Por ejemplo:

```
<?php

if (preg_match("/WEB/","El webdesigner crea un sitio Web."))
{
    echo "La cadena Web se ha encontrado.";
}
```

```

    else {
        echo "La cadena Web no se ha encontrado.";
    }
?>

```

Da como resultado:

La cadena Web no se ha encontrado.

Porque la palabra WEB es distinta de Web.

- La función **`preg_replace()`** sustituye el contenido por otro que pasa en argumento. Esta función devuelve la cadena de caracteres transformada.

Por tanto su sintaxis es:

```

$cadena_transformada =
preg_replace($pattern,$cadena_para_sustituir,$cadena original);

```

### a. Las mayúsculas y las minúsculas

Si no quiere tener en cuenta las mayúsculas y las minúsculas, solo tiene que agregar **i** después del último /.

Por ejemplo:

```

<?php

if (preg_match("/WEB/i","El webdesigner crea un sitio Web."))
{
    echo "La cadena Web se ha encontrado.";
}
else {
    echo "La cadena Web no se ha encontrado.";
}

?>

```

Da como resultado:

La cadena Web se ha encontrado.

Como puede observar, la búsqueda se realiza gracias al patrón de búsqueda (pattern). Puede agregarle muchas otras opciones. Vamos a ver las más utilizadas.

### b. Búsqueda de una palabra, y no una cadena

Hasta ahora, la búsqueda de la cadena de caracteres se realiza en una frase. Por ejemplo, si busca la cadena "designer":

```

<?php

if (preg_match("/designer/","El webdesigner crea un sitio Web."))
{
    echo "La cadena designer se ha encontrado.";
}
else {
    echo "La cadena designer no se ha encontrado.";
}

?>

```

Da como resultado:

La cadena designer se ha encontrado.

Pero si busca solamente la palabra aquellas palabras que comienzan con "designer", debe agregar \b delante de la palabra del patrón de búsqueda.

Por ejemplo:

```

<?php

if (preg_match("/\bdesigner/","El webdesigner crea un sitio Web."))
{
    echo "La palabra designer se ha encontrado.";
}
else {
    echo "La palabra designer no se ha encontrado.";
}

?>

```

Da como resultado:

La palabra designer no se ha encontrado.

En efecto: designer forma parte de la palabra webdesigner, pero no es una palabra propiamente dicha.

Por el contrario, si escribe:

```

<?php

if (preg_match("/\bwеб/","El webdesigner crea un sitio Web."))
{
    echo "la palabra Web se ha encontrado.";
}
else {
    echo "La palabra Web no se ha encontrado.";
}

```

```
}
```

```
?>
```

Da como resultado:

La palabra Web se ha encontrado.

Si desea encontrar una palabra aislada, es suficiente con incluirla dentro de \b.

Por ejemplo:

```
<?php

if (preg_match("/\bsitio\b/","El diseñador web diseña un sitio web."))
{
    echo "Se ha encontrado la palabra sitio.";
}
else
{
    echo "No se ha encontrado la palabra sitio.";
}

?>
```

Muestra:

Se ha encontrado la palabra sitio.

### c. El símbolo O

Este símbolo es | , y permite buscar una cadena u otra.

Por ejemplo:

```
<?php

if (preg_match("/webdesigner|grafista/","El webdesigner crea
un sitio Web."))
{
    echo "La cadena webdesigner o la cadena grafista
se ha encontrado.";
}
else {
    echo "La cadena webdesigner o la cadena grafista
no se ha encontrado.";
}

?>
```

Da como resultado:

La cadena webdesigner o la cadena grafista se ha encontrado.

La cadena "grafista" no existe, pero la cadena "webdesigner" sí.

#### d. Comienzo de la cadena

Este símbolo es ^ y permite buscar una cadena empezando por una palabra.

Por ejemplo:

```
<?php

if (preg_match("/^El/","El webdesigner crea un sitio Web.")) {
    echo "La cadena comienza con la palabra 'El'.";
}
else {
    echo "La cadena no comienza con la palabra 'El'.";
}

?>
```

Da como resultado:

La cadena comienza con la palabra 'El'.

#### e. Fin de cadena

Este símbolo es \$, y permite buscar una cadena que termina con una palabra.

Por ejemplo:

```
<?php

if (preg_match("/Web.$/","El webdesigner crea un sitio Web.")) {
    echo "La cadena termina con la palabra 'Web.'.";
}
else {
    echo "La cadena no termina con la palabra 'Web.'.";
}

?>
```

Da como resultado:

La cadena termina con la palabra 'Web.'

## f. Un carácter en una clase

Una clase permite definir un conjunto de caracteres que están en una cadena. Su sintaxis es [...], con los caracteres dentro de los corchetes.

Por ejemplo:

```
<?php

if (preg_match("/b[oi]n/","de otra manera este sitio Web es bonito."))
{
    echo "La cadena contiene la cadena bon o bin.";
}
else {
    echo "La cadena no contiene la cadena bon o bin.";
}

?>
```

Da como resultado:

La cadena contiene la cadena bon o bin.

La expresión regular es verdadera si contiene la cadena "bon" o la cadena "bin".

Si escribe: preg\_match("/b[aieu]n/,(cadena de caracteres)") entonces la expresión regular sería verdadera si contiene la palabra "ban" o "bin" o "ben" o "bun".

Por ejemplo:

```
<?php

if (preg_match("/b[aou]n$/","de otra manera este sitio Web es bonito."))
{
    echo "La cadena termina con la palabra ban o bon o bun.";
}
else {
    echo "La cadena no termina con la palabra ban o bon o bun.";
}

?>
```

Da como resultado:

La cadena no termina con la palabra ban o bon o bun.

De este modo, se ha añadido un carácter \$ al final de la expresión regular y por tanto la cadena debe terminar con las palabras ban, bon o bun.

## g. Rango de caracteres en una clase

Resulta muy tedioso escribir todas las letras del alfabeto y los dígitos del 0 al 9 en una clase. Afortunadamente, el símbolo - (guión), permite definir un rango de caracteres o números.

Por ejemplo:

```
<?php

if (preg_match("/p[a-z]e/","En el puerto de Amsterdam."))
{
    echo "La cadena contiene la cadena pue.";
}
else {
    echo "La cadena no contiene la cadena pue.";
}

?>
```

Da como resultado:

La cadena contiene la cadena pue.

La expresión regular es verdadera si la cadena contiene otra cadena que comienza con p, seguida de cualquier otra letra del alfabeto y después una e. Por tanto, la palabra "puerto" encaja bien en este supuesto.

Otro ejemplo:

```
<?php

if (preg_match("/ [0-9] /","Esta fruta cuesta 10 euros."))
{
    echo "La cadena contiene un espacio y una cifra entre 0 y 9.";
}
else {
    echo "La cadena no contiene un espacio y una cifra entre 0 y 9.";
}

?>
```

Da como resultado:

La cadena contiene un espacio y una cifra entre 0 y 9.

La expresión regular es verdadera si la cadena contiene un espacio seguido de una cifra entre 0 y 9. Como la cadena contiene el número 10 precedido de un espacio, cumple la expresión regular.

#### **h. La no presencia de un rango de caracteres en una clase**

Si no quiere los caracteres de una clase, tiene que agregar el símbolo ^ al principio de la clase. Este símbolo es el mismo que el que indica la palabra al comienzo de la cadena.

Por ejemplo:

```

<?php

if (preg_match("/n[^a-z]/", "Ponemos una admiración!")) {
    echo "La cadena contiene la letra 'n' seguida de un carácter no
alfabético.";
}
else {
    echo "La cadena no contiene la letra 'n' seguida de un carácter no
alfabético.";
}

?>

```

Da como resultado:

La cadena contiene la letra 'n' seguida de un carácter no alfabético.

En efecto, la palabra admiración contiene una letra n seguida de un carácter que no es una letra, por lo que la expresión regular devuelve verdadero.

### i. Los cuantificadores

Los cuantificadores sirven para definir el número de veces que se repite un carácter o una clase. Los tres principales cuantificadores son:

El símbolo ?: indica que no aparece o aparece una vez el carácter o la clase anterior.

Por ejemplo:

```

<?php

if (preg_match("/da?m/","En el puerto de Amsterdam.")) {
    echo "La cadena contiene dam o dm.";
}
else {
    echo "La cadena no contiene dam o dm.";
}

?>

```

Da como resultado:

La cadena contiene dam o dm.

La expresión regular busca en la cadena "En el puerto de Amsterdam" la letra d seguida de la letra a y de la letra m.

El símbolo + indica una o varias apariciones del carácter o de la clase anterior.

Por ejemplo:

```
<?php

if (preg_match("/da+m/","En el puerto de Amsterdam.")) {
    echo "La cadena contiene dam o daam o daaam...";
}
else {
    echo "La cadena no contiene dam o daam o daaam...";
}

?>
```

Da como resultado:

La cadena contiene dam o daam o daaam...

La expresión regular busca en la cadena "En el puerto de Amsterdam" la letra d seguida de una o varias veces la letra a y de la letra m.

El símbolo \* indica cero, una o varias apariciones del carácter o de la clase anterior.

Por ejemplo:

```
<?php

if (preg_match("/da*m/","En el puerto de Amsterdam.")) {
    echo "La cadena contiene dm o dam o daam o daaam...";
}
else {
    echo "La cadena no contiene dm o dam o daam o daaam...";
}

?>
```

Da como resultado:

La cadena contiene dm o dam o daam o daaam...

La expresión regular busca en la cadena "En el puerto de Amsterdam" la letra d seguida de una o varias veces la letra a y seguida de la letra m.

## j. Intervalos de reconocimiento

Sirven para definir con precisión cuántas veces se puede repetir un carácter o un grupo de caracteres. Este intervalo se realiza con las llaves {}.

- Si quiere que la letra "a" se repita exactamente dos veces, la expresión regular es: a{2}
- Si quiere que la letra "a" se repita al menos dos veces, la expresión regular es: a{2,}

- Si quiere que la letra "a" se repita entre dos y cinco veces, la expresión regular es: a{2,5}

Por ejemplo:

```
<?php

if (preg_match("/1{1,}/", "Nº de teléfono:0034912569875.")) {
    echo "Está por lo menos una vez el número 1 en su Nº de
teléfono.";
}
else {
    echo "No está el número 1 en su Nº de teléfono.";
}

?>
```

Da como resultado:

Está por lo menos una vez el número 1 en su Nº de teléfono.

 Si en su expresión regular pone una cadena con un ?, por ejemplo, si quiere buscar la cadena "¿Quién?", no tiene que interpretar el ? como un cuantificador que indica 0 o 1 del carácter anterior. Para evitar esto, tiene que usar el símbolo \ (barra invertida) que permite evitar el símbolo siguiente \, es decir, no interpretarlo como un símbolo, sino como un carácter.

Por ejemplo:

```
<?php

if (preg_match("/\?/", "¿Quién esta ahí?")) {
    echo "El carácter ? está en su frase.";
}
else {
    echo "El carácter ? no está en su frase.";
}

?>
```

Da como resultado:

El carácter ? está en su frase.

Atención: Los símbolos ?, +, ^, \*, \$ se interpretan como caracteres, y no como símbolos de expresiones regulares cuando están dentro de una clase [...].

Observe que resulta muy difícil escribir una expresión regular. Por fortuna, puede encontrar fácilmente en Internet las expresiones regulares más utilizadas.

A continuación encontrará una expresión regular que comprueba si una dirección de correo electrónico es válida:

```
<?php

$email = "Juan.lopez@espana.es";
if (preg_match("/^[_a-z0-9.-]+@[a-z0-9.-]{2,}\\.[_a-z]{2,4}$/",
$email)) {
    echo "La dirección de correo electrónico es válida.";
}
else {
    echo "La dirección de correo electrónico no es válida.";
}

?>
```

Da como resultado:

La dirección de correo electrónico es válida.

# Los operadores

## 1. Operadores de cadena

### a. La concatenación

Ha tenido la oportunidad de ver, en el capítulo Las bases del lenguaje PHP, la concatenación, que se designa con . (punto) o , (coma), y que permite unir dos cadenas de caracteres.

Por ejemplo:

```
<?php  
  
$nombre = "Juan";  
$apellido = "ESTEFANIA";  
echo $nombre." ".$apellido; //concatenación del nombre +  
//un espacio + el apellido  
  
?>
```

Da como resultado:

Juan ESTEFANIA

### b. Asignación

Se utiliza desde un principio en todos los ejemplos.

La asignación se designa con el signo =. Permite asignar un valor a una variable.

También puede utilizar la combinación .=, que permite concatenar una cadena a una variable y asignarla a esta variable.

Por ejemplo:

```
<?php  
  
$nombre = "Juan";  
$apellido = "López";  
$nombre .= " "; //concatenación del nombre + un espacio  
$nombre .= $apellido; //concatenación del nombre + el apellido  
echo $nombre;  
  
?>
```

Es igual a:

```
<?php
```

```
$nombre = "Juan";
$apellido = "López";
$nombre = $nombre." "; //concatenación del nombre + un espacio
$nombre = $nombre.$apellido //concatenación del nombre + el apellido
echo $nombre;

?>
```

Da como resultado:

Juan López

## 2. Operadores aritméticos

### a. La suma

El operador se designa por +.

Por ejemplo:

```
<?php

$numero = 11;
$resultado = $numero + 5;
echo $resultado;

?>
```

Da como resultado:

16

### b. La resta

El operador se designa por -.

Por ejemplo:

```
<?php

$numero = 11;
$resultado = $numero - 5;
echo $resultado;

?>
```

Da como resultado:

6

### c. La multiplicación

El operador se designa por \*.

Por ejemplo:

```
<?php  
  
$numero = 11;  
$resultado = $numero * 3;  
echo $resultado;  
  
?>
```

Da como resultado:

33

### d. La división

El operador se designa por /.

Por ejemplo:

```
<?php  
  
$numero = 10;  
$resultado = $numero / 5;  
echo $resultado;  
  
?>
```

Da como resultado:

2

### e. El módulo

El módulo es el resto del resultado de dividir el dividendo por el divisor. El operador se designa por %.

Por ejemplo:

```
<?php
```

```
$numero = 11;  
$resultado = $numero % 5;  
echo $resultado;  
  
?>
```

Da como resultado:

1

En efecto, 11 es igual a  $10 + 1$ . 10 es divisible por 5 y resta 1.

Este operador es muy útil para saber si un número es divisible por otro.

Por ejemplo:

```
<?php  
  
$numero = 11;  
$resto = $numero % 5;  
if ($resto == 0) {  
    echo "El número ".$numero." es divisible por 5";  
}  
else {  
    echo "El número ".$numero." No es divisible por 5";  
}  
  
?>
```

Da como resultado:

El número 11 no es divisible por 5

## f. El incremento

El operador se designa por **++**. El orden en que se ubica el operador tiene mucha importancia. **++\$numero** incrementa \$numero y devuelve \$numero, mientras que **\$numero++** devuelve \$numero y lo incrementa.

Por ejemplo:

```
<?php  
  
$numero = 10;  
echo ++$numero; //incrementa y muestra  
echo ";";  
echo $numero;  
  
?>
```

Da como resultado:

11;11

Otro ejemplo:

```
<?php  
  
$numero = 10;  
echo $numero++; //muestra e incrementa  
echo ";";  
echo $numero;  
  
?>
```

Da como resultado:

10;11

### **g. La resta**

El operador se designa por --.

Por ejemplo:

```
<?php  
  
$numero = 10;  
echo --$numero; //resta y muestra  
echo ";";  
echo $numero;  
  
?>
```

Da como resultado:

9;9

Otro ejemplo:

```
<?php  
  
$numero = 10;  
echo $numero--; //muestra y resta  
echo ";";  
echo $numero;
```

?>

Da como resultado:

10;9

### 3. Operadores de comparación

#### a. La igualdad

El operador se designa por **==**.

Por ejemplo:

```
<?php

$numero_1 = 11;
$numero_2 = 11.0;
if ($numero_1 == $numero_2) {
    echo "Los dos números son idénticos";
}
else {
    echo "Los dos números no son idénticos";
}

?>
```

Da como resultado:

Los dos números son idénticos

En efecto, los números 11 y 11.0 no son del mismo tipo, pero los valores son iguales.

Por el contrario, el operador **====** prueba el valor y el tipo. Por tanto, 11, de tipo int, es distinto de 11.0, que es de tipo float.

Ejemplo:

```
<?php

$numero_1 = 11;
$numero_2 = 11.0;
if ($numero_1 === $numero_2) {
    echo "Los dos números son idénticos";
}
else {
    echo "Los dos números no son idénticos";
}
```

Da como resultado:

Los dos números no son idénticos

### b. La diferencia

El operador se designa por !=.

Por ejemplo:

```
<?php  
  
$numero = 11;  
$cadena = "11";  
if ($numero != $cadena) {  
    echo "El número y la cadena son diferentes";  
}  
else {  
    echo "El número y la cadena no son diferentes";  
}  
  
?>
```

Da como resultado:

El número y la cadena no son diferentes

Por el contrario, !== prueba el valor y el tipo de las dos variables, por tanto:

```
<?php  
  
$numero = 11;  
$cadena = "11";  
if ($numero !== $cadena) {  
    echo "El número y la cadena son diferentes";  
}  
else {  
    echo "El número y la cadena nos son diferentes";  
}  
  
?>
```

Da como resultado:

El número y la cadena son diferentes

### c. La comparación

El operador «inferior a» se designa <.

Por ejemplo:

```
<?php

$numero_1 = 11;
$numero_2 = 12;
if ($numero_1 < $numero_2) {
    echo "El número ".$numero_1." es inferior al número
".$numero_2;
}
else {
    echo "El número ".$numero_1." no es inferior al número "
.$numero_2;
}

?>
```

Da como resultado:

El número 11 es estrictamente inferior al número 12

El operador «inferior o igual a» se designa <=.

Del mismo modo, el operador «superior a» se designa >.

Por ejemplo:

```
<?php

$numero_1 = 11;
$numero_2 = 12;
if ($numero_1 > $numero_2) {
    echo "El número ".$numero_1." es superior al número
".$numero_2";
}
else {
    echo "El número ".$numero_1." no es superior al número
".$numero_2";
}

?>
```

Da como resultado:

El número 11 no es estrictamente superior al número 12

El operador superior o igual se designa **>=**.

## 4. El operador ternario

El operador se designa por **?**. Es igual a if else, pero en una sola línea. Su sintaxis es:

```
condición?expresión1:expración2
```

Si la condición es verdadera, se utiliza la expresión1, y se utiliza la expresión2 si no lo es.

Por ejemplo:

```
<?php  
  
$numero_1 = 11;  
$numero_2 = 12;  
echo ($numero_1 == $numero_2)?"Los dos números son idénticos":  
"Los dos números no son idénticos";  
  
?>
```

Da como resultado:

Los dos números no son idénticos

Los valores de \$numero\_1 y \$numero\_2 son distintos, entonces la condición (\$numero\_1 == \$numero\_2) es falsa y por tanto se muestra la expresión2, es decir, "Los dos números no son idénticos".

## 5. Operadores lógicos

### a. Y

El operador se designa por **&&** o **and**.

Por ejemplo:

```
<?php  
  
$edad = 10;  
$nombre = "Juan";  
if ($edad == 10 && $nombre == "Juan") {  
    echo "ok";  
}  
else {  
    echo "no correcto";  
}  
  
?>
```

---

Da como resultado:

ok

Para que la condición `if` sea cierta, es necesario que `$edad` sea igual a 10 y que `$nombre` sea igual a "Juan". Como se da el caso, muestra ok.

### b. O

El operador se designa por `||` u **or**.

Por ejemplo:

```
<?php  
  
$edad = 10;  
$nombre = "Roberto";  
if ($edad == 10 || $nombre == "Juan") {  
    echo "ok";  
}  
else {  
    echo "no correcto";  
}  
  
?>
```

Da como resultado:

ok

Para que la condición `if` sea cierta, `$edad` debe ser igual a 10 o bien `$nombre` igual a "Juan". Como es el caso, muestra ok.

Otro ejemplo:

```
<?php  
  
$edad = 11;  
$nombre = "Roberto";  
if ($edad == 10 && $nombre == "Juan") {  
    echo "ok";  
}  
else {  
    echo "no correcto";  
}  
  
?>
```

Da como resultado:

no correcto

No existe ni \$nombre igual a "Juan", ni \$edad igual a 10, por tanto la condición if es falsa y por tanto se ejecuta el código correspondiente al bloque del else (si no).

 Si los operadores || y && están seguidos, el operador && tiene prioridad sobre el operador ||.

Otro ejemplo:

```
<?php

$edad = 11;
$nombre = "Roberto";
if ($nombre == "Roberto" || $edad == 11 && $edad == 10) {
    echo "ok";
}
else {
    echo "no correcto";
}

?>
```

Y es igual a:

```
<?php

$edad = 11;
$nombre = "Roberto";
if ($nombre == "Roberto" || ($edad == 11 && $edad == 10)) {
    echo "ok";
}
else {
    echo "no correcto";
}

?>
```

Da como resultado:

ok

En este supuesto, si pone los paréntesis, ocurre lo siguiente:

```
<?php

$edad = 11;
```

```
$nombre = "Roberto";
if (($nombre == "Roberto" || $edad == 11) && $edad == 10) {
    echo "ok";
}
else {
    echo "no correcto";
}

?>
```

Y da como resultado:

no correcto

Por tanto, según donde ponga los paréntesis, el significado de la condición puede ser completamente diferente.

# Las funciones

## 1. Creación

Las funciones permiten reutilizar varias veces el código PHP. Por ejemplo, si tiene una página Web con precio sin IVA de diferentes productos, puede crear una función para calcular el precio con IVA de cada producto. Esto evita escribir en cada línea el cálculo en PHP. En su desarrollo, intente agrupar el código. Así el mantenimiento es más fácil. De hecho, cuanto más código escriba, es más probable que cometa errores.

La sintaxis para crear una función es: **función nombre\_de\_la función (\$argumento) { }**

Nunca ponga espacios ni caracteres especiales en el nombre de una función. El parámetro también se llama argumento.

Intente nombrar la función con palabras separadas por el símbolo \_ o con mayúsculas que expliquen lo que hace la función, por ejemplo: nombre\_de\_la\_función o NombreDeLaFunción.

Por ejemplo:

```
<?php

function calculo_iva($precio_bruto) {
    return $precio_bruto * 1.21;
}

?>
```

Esta función calcula el precio con IVA a partir del precio sin IVA, que pasa como parámetro, y devuelve el resultado a través de la palabra clave **return**. Una función no está obligada a devolver un resultado; puede servir solo para mostrar un mensaje, por ejemplo.

Para recurrir a esta función, escriba lo siguiente:

Por ejemplo:

```
<?php
calculo_iva(12);
?>
```

O al final:

```
<?php

function calculo_iva($precio_bruto) {
    return $precio_bruto * 1.21;
}
$precio_iva = calculo_iva(12);
echo "12 euros sin IVA corresponden a ".$precio_iva." euros con IVA";
```

Da como resultado:

12 euros sin IVA corresponden a 14.52 euros con IVA

El número 12 pasa como parámetro de la función; por tanto \$precio\_bruto toma el valor 12, y la función devuelve  $12 * 1.21$  (14.52) en la variable \$precio\_iva.

Puede crear funciones que tomen varios parámetros o ningún parámetro.

En el siguiente ejemplo, la función toma varios parámetros y devuelve un valor:

```
<?php

function junta_palabra($palabra1,$palabra2,$palabra3) {
    $devuelve = $palabra1." ".$palabra2." ".$palabra3;
    Return $devuelve;
}
echo junta_palabra("Hola", "Juan", "ESTEFANIA");

?>
```

Da como resultado:

Hola Juan ESTEFANIA

En el siguiente ejemplo, la función no toma ni devuelve ningún valor:

```
<?php

function muestra_Hola() {
    echo "Hola";
}
muestra_Hola();

?>
```

Da como resultado:

Hola

## 2. Alcance de las variables

Este concepto es muy importante porque es el origen de muchos errores, sobre todo para aquellas personas que no han aprendido las nociones generales del desarrollo.

Las variables declaradas o que pasan como argumentos a una función solo son válidas en la función.

Por ejemplo:

```
<?php

function muestra_palabra($nombre) {
    echo $nombre;
}
muestra_palabra("Hola "); //muestra Hola
echo $nombre; //error porque $nombre no se define

?>
```

Da como resultado:

Hola

**Notice:** Undefined variable: nombre

La variable \$nombre solo es válida en la función muestra\_palabra.

Para evitar este error, defina la variable \$nombre fuera de la función:

```
<?php

$nombre = "Juan";
function muestra_palabra($nombre) {
    echo $nombre;
}
muestra_palabra("Hola "); //muestra Hola
echo $nombre; //muestra Juan

?>
```

Da como resultado:

Hola Juan

**Atención:** la variable \$nombre definida antes que la función no es la misma que \$nombre pasada con el parámetro de la función.

Lo mismo ocurre con una variable declarada fuera de la función, que tampoco es válida en la función.

Por ejemplo:

```
<?php

$nombre = "Juan";
function muestra_palabra() {
```

```

        echo $nombre; //muestra un error
    }
muestra_palabra();

?>

```

Da como resultado:

**Notice:** Undefined variable: nombre

### 3. Las variables globales

Puede declarar una variable con la palabra clave **global**. Esto tiene como efecto definir la variable en todo el código PHP de su página y en las funciones.

Si retomamos el ejemplo anterior agregando \$nombre como variable global en la función:

```

<?php

$nombre = "Juan";
function muestra_palabra() {
    global $nombre;
    echo $nombre;
}
muestra_palabra();

?>

```

Da como resultado:

Juan

La otra solución consiste en utilizar la tabla asociativa **\$GLOBALS**, que contiene todas las variables con su valor. Esta tabla tiene un alcance global a toda la página php (capítulo Transmitir datos de una página a otra - Las variables superglobales).

### 4. Las variables estáticas

Una variable declarada con la palabra clave **static** en una función permite conservar su valor cuando se llama varias veces a la función. Si no se declara como static, se elimina el valor de la variable cada vez que se llama a la función:

```

<?php

function muestra_numero() {
    $numero = 0;
    $numero=$numero + 1;
    echo $numero." ";
}

```

```
}
```

```
muestra_numero(); //muestra 1
```

```
muestra_numero(); //muestra 1
```

```
?>
```

Da como resultado:

```
1; 1;
```

En el siguiente ejemplo, \$numero se declara como static:

```
<?php
```

```
function muestra_numero() {
```

```
    static $numero = 0;
```

```
    $numero=$numero + 1;
```

```
    echo $numero."; "
```

```
}
```

```
muestra_numero(); //muestra 1
```

```
muestra_numero(); //muestra 2
```

```
?>
```

Da como resultado:

```
1; 2;
```

De esta manera, \$numero conserva su valor de una llamada a otra de la función.

## 5. Funciones útiles

- La función **isset()** permite probar que existe una variable. Si existe, devuelve `true`, y si no existe, `false`.

Por ejemplo:

```
<?php
```

```
$frase = "Hola, hace buen tiempo";
```

```
if (isset($frase)){
```

```
    echo "La variable existe.";
```

```
}
```

```
else {
```

```
    echo "La variable no existe.";
```

```
}
```

```
?>
```

Da como resultado:

La variable existe.

Por tanto, su sintaxis es:

```
$existe = iset ($variable);
```

- La función **var\_dump()** permite mostrar el tipo de contenido y el contenido de una variable.

Por ejemplo:

```
<?php  
  
$frase = "Hola, hace buen tiempo";  
var_dump($frase);  
  
?>
```

Da como resultado:

```
string(21) "Hola, hace buen tiempo"
```

Por tanto, su sintaxis es:

```
var_dump($variable);
```

Esta función también acepta las tablas como parámetro.

Por ejemplo:

```
<?php  
  
$tabla = array("Fresa", "Plátano", array(1, 2, 3));  
var_dump($tabla);  
  
?>
```

Da como resultado:

```
array(3) { [0]=> string(6) "Fresa" [1]=> string(6) "Plátano" [2]=> array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) } }
```

- La función **empty()** permite comprobar si una variable es o no nula. Devuelve **true** si es nula, y **false** si no lo es.

Por ejemplo:

```
<?php

$frase = "Hola, hace buen tiempo";
if (empty($frase)){
    echo "La variable es nula.";
}
else {
    echo "La variable no es nula.";
}

?>
```

Da como resultado:

La variable no es nula.

Desde PHP5.5 puede pasar como argumento de una función.

Por ejemplo:

```
<?php

function always_false() {
    return false;
}

if (empty(always_false())) {
    echo 'Se mostrará esto';
}

?>
```

Da como resultado:

Se mostrará esto

Por tanto, su sintaxis es:

```
$nula = empty ($variable o function());
```

## 6. Paso por referencia

Cuando una variable se pasa como argumento de una función, se pasa por el valor, es decir, es una copia de la variable que se ha pasado como argumento.

Ejemplo de paso por el valor:

```

<?php

function añadir_señor($argumento) {
    $argumento = $argumento . " Señor";
}
$texto = "Hola";
añadir_señor($texto);
echo $texto;

?>

```

Da como resultado:

Hola

Se pasa la variable \$texto como argumento a la función; no se cambia su valor en la función, solo se envía su valor a la función.

El paso del parámetro por referencia se realiza agregando el símbolo & delante de la variable. Tiene como efecto pasar la dirección de memoria de la variable y así puede modificar su valor.

Ejemplo de paso por referencia:

```

<?php

function añadir_señor(&$argumento) { //paso por referencia
    $argumento = $argumento . " Señor";
}
$texto = "Hola";
añadir_señor($texto);
echo $texto;

?>

```

Da como resultado:

Hola Señor

Se pasa la variable \$texto como argumento a la función con ayuda de &\$argumento y se modifica al concatenar \$argumento con la cadena de caracteres " Señor". Al salir de la función, \$texto tiene como valor \$argumento, es decir, "Hola Señor".

## 7. Funciones de la función de gestión

- La función **func\_get\_arg(int \$numero)** devuelve un elemento de la lista de argumentos de entrada, es decir, los argumentos que se han pasado con \$numero, como el número del argumento.

Por ejemplo:

```

<?php

function muestra_argumentos($argumento1,$argumento2) {
    echo "El valor del primer argumento es:".func_get_arg(0);
    echo "El valor del segundo argumento es:".func_get_arg(1);
}
muestra_argumentos("Hola","Roberto");

?>

```

Da como resultado:

```

El valor del primer argumento es:Hola
El valor del segundo argumento es:Roberto

```

- La función **func\_num\_args()** devuelve el número de argumentos de entrada, es decir, los argumentos que se han pasado.

Por ejemplo:

```

<?php

function muestra_numero_argumentos($argumento1,$argumento2) {
    echo "El número de argumentos es:".func_num_args();
}
muestra_numero_argumentos("Hola","Roberto");

?>

```

Da como resultado:

```

El número de argumentos es:2

```

- La función **func\_get\_args()** devuelve los argumentos que pasan como argumentos en forma tabla.

Por ejemplo:

```

<?php

function muestra_argumentos($argumento1,$argumento2) {
    $num_args = func_num_args();
    $arg_lista = func_get_args();
    for ($i = 0; $i < $num_args; $i++) {
        echo "El argumento $i es: " . $arg_lista[$i] . "<br />\n";
    }
}
muestra_argumentos("Hola","Roberto");

```

```
?>
```

Da como resultado:

```
El argumento 0 es:Hola  
El argumento 1 es:Roberto
```

- La función **function\_exists()** devuelve un booleano e indica si una función existe o no.

Por ejemplo:

```
<?php  
  
function muestra_numero_argumentos($argumento1,$argumento2) {  
    echo "El número de argumentos es:".func_num_args();  
}  
if (function_exists('muestra_numero_argumentos')) {  
    echo "La función existe.";  
} else {  
    echo "La función no existe.";  
}  
  
?>
```

Da como resultado:

```
La función existe.
```

 Puede crear su función de manera dinámica con ayuda de la función **create\_function()**. Para más información, visite la página Web: [www.php.net/manual/es/function.create-function.php](http://www.php.net/manual/es/function.create-function.php)

## 8. Recursividad

La recursividad significa que una función se llama a sí misma. Es muy útil para navegar en una arborescencia, sobre todo si se desconoce de antemano el número de ramas.

Por ejemplo:

```
<?php  
  
function funcion_recursiva($n)  
{  
    $n++; //incrementa n de 1 en 1  
    echo "$n,";  
    if($n < 10){ // si n es inferior a 10, vuelve a llamar la función  
        funcion_recursiva($n);  
    }  
}
```

```

    }
funcion_recursiva(0); // muestra los números de 1 a 10

?>

```

Da como resultado:

1,2,3,4,5,6,7,8,9,10,

En la condición `if ($n < 10)`, la función se llama a sí misma, y se puede llamar hasta diez veces en total. De hecho, si no hubiera una condición (`$n < 10`), el bucle sería infinito.

A continuación veremos el código que muestra los valores de las tablas tridimensionales:

```

<?php

function mostrar_tabla($tabla,$titulo="", $nivel=0) {
    // Parámetros
    // - $tabla = tabla que debe mostrar el contenido
    // - $titulo = título que hay que mostrar encima del contenido
    // - $nivel = nivel de visualización
    if ($titulo != "") { // Si hay un título, mostrarlo.
        echo "<br /><b>".$titulo."</b><br />";
    }
    // Comprobar si hay datos.
    if (isset($tabla)) { // Hay datos
        // Navegar por la tabla que se pasa como argumento.
        foreach ($tabla as $clave => $valor) {
            // mostrar la clave (con indentation función
            // del nivel).
            // htmlentities() es la función que convierte los
            // caracteres especiales HTML
            echo str_pad('',12*$nivel, ' ');
            htmlentities($clave).' = ';
            // mostrar el valor
            if (is_array($valor)) { // comprueba si es una tabla
                echo '<br />';
                // Llama recursivamente a mostrar_tabla para
                // mostrar la tabla en cuestión
                mostrar_tabla($valor,'',$nivel+1);
            } else { // es un valor escalonado
                // mostrar el valor.
                echo htmlentities($valor).'  
';
            }
        }
    } else { // no hay datos
        echo '<br />';
    }
}
// Declaración de las tablas.
$tab_caracteristica_ESTEFANIA = array("nombre" => "PABLO",
"profesión" => "ministro", "edad" => 50);

```

```

$tab_caracteristica_LUNA= array("nombre" => "ROBERTO",
"profesión" => "agricultor","edad" => 45);
$tab_persona['ESTEFANIA'] = $tab_caracteristica_ESTEFANIA;
$tab_persona['LUNA'] = $tab_caracteristica_LUNA;

// Mostrar una tabla con dos dimensiones (Apellido/Características).
mostrar_tabla($tab_persona,'Apellido/Características');

?>

```

Da como resultado:

#### **Apellido/Características**

```

ESTEFANIA =
    nombre = PABLO
    profesión = ministro
    edad = 50

LUNA =
    nombre = ROBERTO
    profesión = agricultor
    edad = 45

```

La principal ventaja de esta función recursiva es que, si tiene una tabla de tres, cuatro o más dimensiones, no es preciso cambiar el código, porque no depende del número de dimensiones.

## **9. Funciones predefinidas en PHP**

Hay alrededor de 4500 funciones predefinidas en PHP. Las puede consultar en la siguiente página Web, donde están agrupadas por temas: <http://www.php.net/manual/es/funcref.php>

Ya ha visto algunas de ellas en secciones anteriores, como **substr()** o **implode()**. Ahora vamos a ver algunas funciones complementarias que son también muy útiles.

### **a. Generar un número aleatorio**

La función **rand()** permite generar un valor aleatorio comprendido entre 0 y 32768. Puede pasar como argumento de entrada unos límites mínimo y máximo.

Por ejemplo:

```

<?php

echo rand()."<br />";
echo rand(10,20)."<br />";

?>

```

Da como resultado:

**b. Redondear un número decimal**

La función **round(\$numero\_decimal)** permite redondear un número decimal.

Por ejemplo:

```
<?php  
echo round(3.141592);  
?>
```

Da como resultado:

3

Esta función acepta la precisión como argumento complementario, es decir, el número de dígitos que quiera poner después de la coma.

Por ejemplo:

```
<?php  
echo round(3.141592,2); //2 dígitos después de la coma  
?>
```

Da como resultado:

3.14

Otro ejemplo:

```
<?php  
echo round(3.779592,2); //2 dígitos después de la coma  
?>
```

Da como resultado:

3.78

Es decir se ha redondeado el número.

### c. Tomar el valor absoluto de un número

La función **abs(\$nombre)** permite recuperar el valor absoluto de un número.

Por ejemplo:

```
<?php  
echo abs(-5.2);  
?>
```

Da como resultado:

5.2

Esta función acepta como parámetro una cadena de caracteres.

Por ejemplo:

```
<?php  
echo abs("68");  
?>
```

Da como resultado:

68

### d. Crear un identificador único

La función **uniqid()** permite generar de forma aleatoria un valor de trece caracteres, de tal manera que este valor sea único. A veces es muy útil para generar identificadores únicos que se van a insertar en la base de datos.

Por ejemplo:

```
<?php  
echo "Id único:".uniqid()." ";  
echo "Id único:".uniqid()." ";  
echo "Id único:".uniqid();  
?>
```

Da como resultado:

Id único:4df0d26502f82, Id único:4df0d26502f86, Id único:4df0d26502f87

#### e. Mostrar información de PHP

La función **phpversion()** permite mostrar la versión actual de PHP.

Por ejemplo:

```
<?php  
echo phpversion  
?>
```

Da como resultado:

5.3.3

La función **phpinfo()** permite mostrar información de la configuración de PHP que ha instalado en su servidor, como las variables de entorno o la configuración de Apache. Esta información se almacena en el archivo `php.ini`, cuya utilización se explicará en el siguiente capítulo.

Por ejemplo:

```
<?php  
phpinfo();  
?>
```



<b>System</b>	Windows NT ANGEL_MARIA1 6.0 build 6002 (Windows Vista Home Premium Edition Service Pack 2) i586
<b>Build Date</b>	Jul 21 2010 20:00:47
<b>Compiler</b>	MSVC6 (Visual C++ 6.0)
<b>Architecture</b>	x86
<b>Configure Command</b>	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--disable-isapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\ sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\ sdk,shared" "--with-oci8-11g=D:\php-sdk\oracle\instantclient11\ sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static"
<b>Server API</b>	Apache 2.0 Handler
<b>Virtual Directory Support</b>	enabled
<b>Configuration File (php.ini) Path</b>	C:\Windows
<b>Loaded Configuration File</b>	C:\Program Files\EasyPHP-5.3.3.1\apache\php.ini
<b>Scan this dir for additional .ini files</b>	(none)
<b>Additional .ini files parsed</b>	(none)
<b>PHP API</b>	20090626
<b>PHP Extension</b>	20090626
<b>Zend Extension</b>	220090626
<b>Zend Extension Build</b>	API220090626,TS,VC6
<b>PHP Extension Build</b>	API20090626,TS,VC6

La función **ini\_get\_all()** devuelve toda la información del archivo PHP.ini, pero en forma de tabla.

Para terminar, la función **get\_loaded\_extensions()** devuelve una tabla que contiene todas las extensiones que se han descargado. Lo veremos en el capítulo Configuración.

#### f. Enviar un e-mail

La función **mail()** permite enviar un e-mail. Es una función básica que no debe utilizar si quiere enviar un gran volumen de mensajes de correo electrónico, porque con cada envío cierra y vuelve a abrir una conexión al servidor. Hay otras funciones más prácticas y eficaces, como PEAR o PHPMailer. Sin embargo, el estudio de esta función permite ver las bases de envío de un e-mail, que son comunes a todas las funciones.

Los parámetros de la función son:

- \$to: los destinatarios del e-mail
- \$subject: asunto del e-mail
- \$message: contenido del e-mail
- \$headers: parámetro opcional que contiene el encabezado del e-mail

El encabezado permite definir el remitente del e-mail (From), el tipo MIME, la codificación y otros parámetros. Puede encontrar más información en el siguiente enlace:

<http://www.php.net/manual/es/function.mail.php>

El remitente (From), la dirección SMTP y el número de puerto SMTP se definen en el archivo PHP.ini, al que puede acceder desde el menú **Configuración - PHP**.

Por ejemplo:

```
<?php

$to = 'persona@ejemplo.com';
$subject = 'Asunto';
$message = '¡Hola a todos!';
$headers = 'From: webmaster@misitio.com'."\r\n".
'Reply-To: webmaster@misitio.com'."\r\n".
'MIME-Version: 1.0'."\r\n";

mail($to, $subject, $message, $headers);
?>
```

La función mail( ) no se puede autenticar y por tanto no funciona en modo local si, por ejemplo, utiliza el servidor SMTP de Gmail. Deberá utilizar la librería Mail-1.2.0 de PEAR que está disponible en la siguiente dirección: <http://pear.php.net/package/Mail/download/1.2.0/>

El e-mail que ha enviado está en formato texto. Para enviarlo en formato HTML, debe declarar este formato en el encabezado (header):

```
$headers = 'From: webmaster@misitio.com'."\r\n"
'Reply-To: webmaster@misitio.com'."\r\n".
'MIME-Version: 1.0'."\r\n".
'Content-type: text/html; charset=iso-8859-1'."\r\n";
```

## Almacenar una función en una variable

PHP permite almacenar una función en una variable. Solo tiene que pasar como argumento el nombre de una función que ya existe.

Por ejemplo:

```
<?php

function añadir($argumento1,$argumento2) {
    return $argumento1 + $argumento2;
}
function sustraer($argumento1,$argumento2) {
    return $argumento1 - $argumento2;
}
function operacion($tipo_operacion,$valor1,$valor2) {
    return $tipo_operacion($valor1,$valor2);
}
echo "6 + 4 =" .operacion("añadir",6,4).", ";
echo "6 - 4 =" .operacion("sustraer",6,4);

?>
```

Da como resultado:

6 + 4 =10, 6 - 4 =2

Observe que el tipo de operación se pasa como parámetro a la función **operacion**. Esta función recupera el tipo de operación (añadir, por ejemplo) y va a llamar a la función **añadir** con sus parámetros. Pero para que esto funcione, ya debe existir la función **añadir**.

Este concepto puede ser de gran utilidad si crea el objeto en PHP, como veremos más adelante.

### 1. Ejercicio

#### a. Enunciados

##### Ejercicio 1 (dificultad media)

Cree una función que muestre una frase que contenga de forma aleatoria las tres palabras «Hola», «Señor» y «Roberto». Cada palabra solo puede aparecer una vez. Esta función recibe como parámetros las tres palabras «Hola», «Señor» y «Roberto».

##### Ejercicio 2 (difícil)

Cree una tabla que contenga diez dígitos aleatorios comprendidos entre 1 y 100, y ordénelos, utilizando métodos como la ordenación de tabla o sort(). Debe crear una función que intercambie dos valores en una tabla. Muestre estos valores separados por una coma.

### **Ejercicio 3 (medio)**

Escriba una función que permita calcular el factorial de un número de manera recursiva.

Por ejemplo, el factorial de 7 es: 1\*2\*3\*4\*5\*6\*7

Muestre entonces el factorial de 20 (2.4329020081766E+18).

### **Ejercicio 4 (difícil)**

Retome el ejercicio 1, pero ahora debe generalizar la función para que tome como parámetro una tabla que contenga cualquier número de palabras.

### **Ejercicio 5 (difícil): creación de una tabla HTML con valores al cubo**

La tabla A tiene los elementos 3, 8, 15 y 16. Cree una tabla B con ayuda de un bucle que contenga todos los elementos comprendidos entre 1 y 20, salvo los elementos de la tabla A. Cree una función que calcule al cubo este dígito. Muestre en una tabla HTML los elementos de la tabla B en la primera columna y en una segunda columna los elementos de B al cubo.

## **b. Soluciones**

### **Solución del ejercicio 1**

```
<?php

function muestra($palabra1,$palabra2,$palabra3) {
    $frase = "";
    $valor_al_azar = rand(1,6); //Hay 6 posibles combinaciones
    if ($valor_al_azar == 1) {
        $frase = $palabra1." ".$palabra2." ".$palabra3;
    }
    else if ($valor_al_azar == 2) {
        $frase = $palabra1." ".$palabra3." ".$palabra2;
    }
    else if ($valor_al_azar == 3) {
        $frase = $palabra2." ".$palabra1." ".$palabra3;
    }
    else if ($valor_al_azar == 4) {
        $frase = $palabra2." ".$palabra3." ".$palabra1;
    }
    else if ($valor_al_azar == 5) {
        $frase = $palabra3." ".$palabra1." ".$palabra2;
    }
    else if ($valor_al_azar == 6) {
        $frase = $palabra3." ".$palabra2." ".$palabra1;
    }
    echo $frase;
}
$palabra1 = "Hola";
$palabra2 = "Señor";
```

```

$palabra3 = "Roberto";
muestra($palabra1,$palabra2,$palabra3);

?>

```

## **Solución del ejercicio 2**

```

<?php

$tabla1 = array();

//Función para intercambiar el valor en una tabla
//La tabla pasa por referencia
function intercambiar(&$tabla, $i, $j)
{
    $temporal = $tabla[$i];
    $tabla[$i] = $tabla[$j];
    $tabla[$j] = $temporal;
}

//rellenar una tabla de 10 valores aleatorios comprendidos
//entre 1 y 100
for ($i=1;$i<=10;$i++) {
    $tabla1[$i]=rand(1,100);
}

$longitud=10;
while($longitud>0)
{
    // búsqueda del mayor valor de la tabla
    $maximo = 1;
    for($i=1; $i<=$longitud; $i++) {
        if($tabla1[$i]>$tabla1[$maximo]) {
            $maximo = $i;
        }
    }
    //intercambia el máximo con el último elemento
    intercambiar($tabla1, $maximo, $longitud);

    // Procesar el resto de la tabla
    $longitud--;
}

//mostrar la tabla
for ($i=1;$i<=10;$i++) { //tabla de 1 a 10
    echo $tabla1[$i].",";
}

?>

```

### **Solución del ejercicio 3**

```
<?php

function factorial($numero)
{
    if($numero === 0) // condición de parada
        return 1;
    else
        return $numero * factorial($numero-1);
}
echo "El factorial de 20 es:".factorial(20);

?>
```

### **Solución del ejercicio 4**

```
<?php

function muestra($tab) {
    $tabla_fin = array(); //tabla que contiene las palabras para
                          //mostrar aleatoriamente
    $j=0; //contador de esta tabla
    $numero_elementos = sizeof($tab);
    while ($j < $numero_elementos) {
        $clave_aleatoria = array_rand($tab);
        //si el valor obtenido aleatoriamente ya está presente en
        //la tabla de fin, disminuye $i para volver a empezar
        //a ejecutar un bucle una vez más.
        if (! in_array($tab[$clave_aleatoria],$tabla_fin)) {
            $tabla_fin[$j] = $tab[$clave_aleatoria];
            $j++;
        }
    }
    foreach ($tabla_fin as $val) {
        echo $val." ";
    }
}
$tabla_palabra[0] = "Hola";
$tabla_palabra[1] = "Señor";
$tabla_palabra[2] = "Roberto";
$tabla_palabra[3] = "Pepe";
muestra($tabla_palabra);

?>
```

### **Solución del ejercicio 5**

```
<?php
```

```

$A = array(3,8,15,16); // creación de la tabla con cuatro elementos
$B = array();
$contador = 0; //iniciar el contador que representa el índice
                //de la tabla $B
for ($i=1;$i<=20;$i++) { //tabla de 1 a 20
    if (! in_array($i,$A)) {
        $contador++; //aumento del contador
        $B[$contador]=$i;
    }
}
//función que devuelve el cubo de un valor
function cubo($valor) {
    $devolver = $valor*$valor*$valor;
    return $devolver;
}

//mostrar la tabla HTML con el valor de la tabla $B y su resultado
//al cubo
echo "<table border='1'>";
echo "<tr><td>valor</td><td>valor al cubo </td></tr>"; //mostrar
                                                                //encabezado
foreach ($B as $val) {
    echo "<tr><td>".$val."</td><td>".cubo($val)."</td></tr>";
}
echo "</table>";

?>

```

## Las fechas

En esta parte tratamos todas las funciones PHP que permiten manipular las fechas. Normalmente las fechas se recuperan en un formato con un idioma concreto, y esto le obliga a convertirlas a su idioma.

- La función **time()** devuelve la hora actual, que se mide en segundos desde el inicio de UNIX (1 de Enero de 1970 00:00:00 GMT). Esta hora también se llama timestamp UNIX.

Por ejemplo:

```
<?php  
  
echo time();  
  
?>
```

Da como resultado:

1381329777

Por tanto, su sintaxis es:

```
time()
```

Esta función se utiliza sobre todo para realizar cálculos con fechas, por ejemplo para encontrar la duración de procesamiento en la base de datos.

- La función **date()** devuelve la fecha en el formato que se ha pasado como argumento.

Por ejemplo:

```
<?php  
  
echo date('d.m.y');  
  
?>
```

Da como resultado:

09.10.13

Por tanto, su sintaxis es:

```
$fecha_del_dia = date($formato)
```

siendo \$formato una cadena que contiene las letras que permiten definir el formato.

A continuación mostramos una lista con los principales formatos que se utilizan en la función date. Esta lista no es exhaustiva y puede encontrar más información en el siguiente enlace: <http://www.php.net/manual/es/function.date.php>

## Día

- J Día del mes con dos dígitos sin ningún cero inicial: de 1 a 31.
- d Día del mes con dos dígitos con un cero inicial en la función del día: de 01 a 31.
- I (L minúscula) Día de la semana en inglés: de Sunday a Saturday.
- w Día de la semana con un formato numérico de 0 (domingo) a 6 (sábado).
- z Día del año: de 0 a 366.

## Semana

- W Número de la semana en el año (las semanas empiezan el lunes). Ejemplo: 42 (la 42.<sup>a</sup> semana del año).

## Mes

- F Mes, textual, versión amplia en inglés, como por ejemplo January o December.
- m Mes en formato numérico, con ceros iniciales: de 01 a 12.
- n Mes sin ceros iniciales: de 1 a 12.
- t Número de días en el mes: de 28 a 31.

## Año

- L Año bisiesto: 1 si es bisiesto, 0 si no lo es.
- Y Año con cuatro dígitos (por ejemplo, 1999 y 2003).
- y Año con dos dígitos (por ejemplo, 99 y 03).

## Hora

- a Ante meridiem y Post meridiem (minúsculas): am o pm.
- A Ante meridiem y Post meridiem (mayúsculas): AM o PM.
- g Hora (formato 12 h) sin ceros iniciales: de 1 a 12.
- G Hora (formato 24 h) sin los ceros iniciales: de 0 a 23.
- h Hora (formato 12 h) con ceros iniciales: de 01 a 12.
- H Hora (formato 24 h) con ceros iniciales: de 00 a 23.
- s Segundos con ceros iniciales: de 00 a 59.
- i Minutos con ceros iniciales: de 00 a 59.

Esta función **date()** puede tomar también como parámetro opcional el timestamp UNIX para definir otra fecha distinta a la fecha del día que quiere mostrar.

Por ejemplo:

```
<?php
```

```

$ProximaSemana = time() + (7 * 24 * 60 * 60); // Añadir una semana a
// la hora actual. O 7 días = 7x24 horas = 7x24x60 minutos =
// 7x24x60x60 segundos
echo "Hoy: ".date('d-m-Y').", ";
echo "Próxima semana: ".date('d-m-Y', $ProximaSemana)."\n";

?>

```

Da como resultado:

Hoy: 01-02-2013, Próxima semana: 08-02-2013

- La función **mkttime()** devuelve el timestamp UNIX desde una fecha que se pasa como parámetro.

Por ejemplo:

```

<?php

echo mktime(0,0,0,2,1,2012); //muestra el timestamp de 01/02/2012

?>

```

Da como resultado:

1328050800

Su sintaxis es:

```
$timestamp = mktime($hora,$minuto,$segundo,$mes,$día,$año)
```

 Esta función corrige los datos no válidos automáticamente. Por ejemplo, 32 de Enero se corregirá como 1 de Febrero.

- La función **microtime()** devuelve el timestamp UNIX en microsegundos. Como parámetro opcional utiliza un booleano, que permite devolver un número real si es verdadero, o una cadena de caracteres si no lo es.

Por ejemplo:

```

<?php

// Muestra en forma de cadena.
echo microtime(). '<br />';
// Muestra en forma de real.
echo microtime(TRUE). '<br />';
// Para mostrar los microsegundos únicamente, transformación (cast)
// de la cadena en modo real.
echo (float) microtime() . '<br />';

```

```
?>
```

Da como resultado:

```
0.31211200 1328104356  
1328104356.3121  
0.3121
```

Por tanto, su sintaxis es:

```
$microsegundo = microtime($bool)
```

- La función **getdate()** devuelve una tabla asociativa de la fecha y la hora actuales.

Por ejemplo:

```
<?php  
  
print_r(getdate());  
  
?>
```

Da como resultado:

```
Array ( [seconds] => 37 [minutes] => 48 [hours] => 10 [mday] => 15 [wday] => 3 [mon] => 6 [year] => 2011  
[yday] => 165 [weekday] => Wednesday [month] => June [0] => 1308127717 )
```

Por tanto, su sintaxis es:

```
$tabla = getdate()
```

- La función **checkdate()** indica si una fecha es válida o no. Esta función tiene en cuenta los años bisiestos. Toma como parámetros el mes, el día y el año, y devuelve verdadero o falso.

Por ejemplo:

```
<?php  
  
$valido = checkdate(13, 10, 2011);  
  
if($valido == true )  
{  
    echo 'La fecha es válida';  
}  
else  
{  
    echo 'La fecha no es válida';  
}
```

```
}
```

```
?>
```

Da como resultado:

La fecha no es válida

De hecho, no existe el decimotercer mes.

Por tanto, su sintaxis es:

```
$valido = checkdate($mes,$día,$año)
```

- La función **strtotime()** permite convertir un texto en inglés en timestamp, por tanto en fecha.

Por ejemplo:

```
<?php

//strtotime reenvía un timestamp y la función date() permite mostrar
//la fecha en un formato adecuado desde el timestamp
echo strtotime("now").", ".date('d-m-Y',strtotime("now")). "<br />";

echo strtotime("10 September 2011").", ".date('d-m-Y',strtotime("10
September 2011")). "<br />";

echo strtotime("next Thursday").", ".date('d-m-Y',strtotime("next
Thursday")). "<br />;

echo strtotime("last Tuesday").", ".date('d-m-Y',strtotime("last
Tuesday")). "<br />;

echo strtotime("+1 day").", ".date('d-m-Y',strtotime("+1 day")). "<br />";
echo strtotime("+1 week").", ".date('d-m-Y',strtotime("+1 week")). "
"<br />;

echo strtotime("+2 week 2 days 2 hours 2 seconds").", ".date('d-m-y',
strtotime("+2 week 2 days 2 hours 2 seconds")). "<br />;

?>
```

Da como resultado:

1381822566, 15-10-2013

1315605600, 10-09-2011

1381960800, 17-10-2013

1381183200, 08-10-2013

1381908966, 16-10-2013

Por tanto, su sintaxis es:

```
$timestamp = strtotime($cadena)
```

- La función **strftime()** permite convertir una fecha que tiene la forma de timestamp en una cadena formateada correctamente. Toma como parámetro el formato de tipo cadena de caracteres y timestamp como opción, si no quiere la fecha actual. Esta función se utiliza con **setlocale()**, que permite definir el país en el que nos encontramos.

Por ejemplo:

```
<?php

setlocale(LC_ALL, "es_ES", "esp");
echo strftime("En España el día es: %A <br />");

echo strftime("Hoy es %d %m %Y <br />");

setlocale(LC_TIME, "fr_FR", "fra");
echo strftime("En Francia el mes es %B.\n");

?>
```

Da como resultado:

En España el día es: miércoles

Hoy es 09 10 2013

En Francia el mes es Octubre.

Por tanto, su sintaxis es:

```
$cadena = strftime($formato)
```

Opciones de formato:

%d: día del mes, con dos dígitos: de 01 a 31.

%m: número del mes, con dos dígitos: de 01 a 12.

%y: año con dos dígitos: por ejemplo, 01.

%Y: año con cuatro dígitos: por ejemplo, 2001.

%H: hora, con formato 24 h.

%M: minutos con dos dígitos: de 00 a 59.

%S: segundos con dos dígitos: de 00 a 59.

%a: nombre abreviado del día de la semana.

%A: nombre completo del día de la semana.

%j: número del día del año con tres dígitos: de 001 a 365.

%w: número del día de la semana: de 0 = domingo a 6 = sábado.

%u: número del día de la semana: de 1 = lunes a 7 = domingo.

%b o %h: nombre abreviado del mes.

%B: nombre completo del mes.

%U: número de la semana en el año: siendo el primer día de la primera semana el primer domingo del año.

%W: número de la semana en el año: siendo el primer día de la primera semana el primer lunes del año.

%V: número de la semana en el año según la normativa ISO 9601.

%c: formato por defecto para la fecha y la hora.

%x: formato por defecto solo para la fecha.

%X: formato por defecto solo para la hora.

%R: idéntico a %H: %M.

%T: idéntico a %H: %M: %S.

%Z: franja horaria, nombre o abreviatura.

%t: tabulación.

%n: vuelta al registro.

%%: un carácter '%' literal.

- La función **date\_default\_timezone\_set()** establece la diferencia horaria de todas las funciones de fecha y hora. Esta función toma como parámetro el identificador de diferencia horaria, que es una cadena de caracteres con la zona y el país. Si quiere formatear la fecha, debe definir la zona de diferencia horaria; de lo contrario, tendrá un mensaje de advertencia de tipo E\_NOTICE.

Por ejemplo, en España:

```
<?php  
date_default_timezone_set('Europe/Madrid');  
?>
```

- La función **date\_create\_from\_format()** permite devolver un objeto date que se formatea desde una cadena de caracteres que contiene una fecha. Esta función también se llama **DateTime::createFromFormat()**. Toma como parámetro el formato de tipo cadena de caracteres, la fecha y la hora en forma de cadena de caracteres y, como

opción, el objeto **DateTimeZone**, que define la zona de diferencia horaria. El formato es el mismo que el que se utiliza con la función **date()**.

Por ejemplo:

```
<?php  
date_default_timezone_set('Europe/Madrid');  
$format = 'Y-m-d H:i:s';  
$date = date_create_from_format($format, '2011-11-15 12:14:19');  
echo "Muestra con el formato día-mes-año hora:minuto:segundo -> "  
.date_format($date, 'd-m-Y H:i:s');  
?>
```

Da como resultado:

Muestra con el formato día-mes-año hora:minuto:segundos -> 15-11-2011 12:14:19

# Los archivos

## 1. Introducción

A veces resulta muy útil almacenar información en un archivo del servidor, en lugar de en la base de datos. Puede ser más rápido y accesible; sin embargo, es menos seguro y pueden surgir errores en la escritura si hay conexiones simultáneas.

A lo largo de esta parte vamos a ver las funciones más utilizadas.

Para empezar, debe crear un archivo llamado «archivo.txt» en el directorio localweb, es decir, donde están sus páginas PHP. Este archivo debe tener derechos de escritura para poder escribir en ellos. Esto se produce automáticamente cuando trabaja en modo local, pero probablemente tendrá que cambiar los permisos del archivo si lo transfiere por FTP a otra ubicación.

## 2. Lectura rápida

- La función **file\_get\_contents()** permite leer el contenido de un archivo y lo devuelve en una cadena de caracteres.

En el siguiente ejemplo, el archivo texto contiene la frase "¡Hola!"

```
<?php  
  
$contenido = file_get_contents('archivo.txt');  
echo $contenido;  
  
?>
```

Da como resultado:

"¡Hola!"

- La función **readfile()** también permite leer el contenido de un archivo, pero devuelve el número de caracteres del archivo y muestra automáticamente el contenido.

En el siguiente ejemplo, el archivo de texto contiene la frase "¡Hola!"

```
<?php  
  
$archivo = 'archivo.txt';  
$numero=readfile($archivo);  
echo "<br /> El número de caracteres del archivo es:".$numero;  
  
?>
```

Da como resultado:

"¡Hola!"

El número de caracteres del archivo es:8

- La función **file()** permite leer el contenido de un archivo, pero devuelve el contenido en una tabla, línea por línea.

En este ejemplo, el archivo de texto contiene la frase "¡Hola!", un salto de línea y la frase "Señora ESTEFANIA."

```
<?php  
  
$tabla = file('archivo.txt');  
foreach ($tabla as $línea) {  
    echo $línea."<br />";  
}  
  
?>
```

Da como resultado:

"¡Hola!"

"Señora ESTEFANIA."

Cada línea del archivo se vuelve a encontrar en cada elemento de la tabla.

### 3. Escritura rápida

La función **file\_put\_contents()** permite escribir en un archivo el contenido de una cadena de caracteres. Toma como parámetro el nombre del archivo y la variable que contiene el texto que va a insertar en el archivo. Si ya existe el archivo, su contenido se elimina.

Por ejemplo:

```
<?php  
  
$contenido = "Hola Sra. ESTEFANIA.";  
file_put_contents("archivo.txt", $contenido);  
  
?>
```

Escrito en el archivo llamado archivo.txt:

Hola Sra. ESTEFANIA.

### 4. Abrir y cerrar un archivo

La función **fopen()** permite desencadenar la apertura del archivo. Esta función tiene dos parámetros: el nombre

del archivo y el modo de apertura del archivo.

Por ejemplo:

```
<?php  
  
$recurso = fopen('archivo.txt', 'r');  
  
?>
```

La variable \$recurso contiene un objeto que permite manipular el archivo. Veremos más adelante cómo se utiliza.

Observe que el carácter 'r' es el segundo parámetro.

La 'r' abre el archivo solo en modo de lectura.

La 'r+' abre el archivo en modo de lectura y escritura.

La 'w' abre el archivo solo en modo de escritura, vacía el archivo y lo crea si aún no existe.

La 'w+' abre el archivo en modo de lectura y escritura, vacía el archivo y lo crea si aún no existe.

La 'a' abre el archivo solo en modo de escritura y además crea el archivo si aún no existe. Lo que escriba después no modificará el texto que ya existe.

La 'a+' abre el archivo en modo de lectura y escritura y además crea el archivo que aún no existe.

- La función **fclose()** permite cerrar el archivo. Esta función devuelve true o false en caso de error.

Por ejemplo:

```
<?php  
  
$recurso = fopen('archivo.txt', 'r');  
fclose($recurso);  
  
?>
```

## 5. Leer y escribir

La función **fgetc()** permite leer el archivo carácter a carácter. Es preciso que incluya un bucle para navegar por todos los caracteres del archivo.

Por ejemplo:

```
<?php  
  
$recurso = fopen('archivo.txt', 'r');
```

```

if (!$recurso) { //comprueba si el archivo está abierto correctamente
    echo "Imposible abrir el archivo archivo.txt";
}
//bucle mientras haya un carácter
while (false !== ($char = fgetc($recurso))) {
    echo $char;
}

fclose($recurso);

?>

```

Da como resultado:

"¡Hola!" "Señora ESTEFANIA."

Observe que no lee el salto de línea; por tanto esta función no es muy práctica.

- La función **fgets()** permite leer el archivo línea a línea. Por tanto, es preciso incluir esta función en un bucle para navegar por todas las líneas del archivo. Toma como parámetro el recurso del archivo y el tamaño de cada línea como opción. La función recupera este número de caracteres o los caracteres hasta que se encuentra un final de línea.

Por ejemplo:

```

<?php

$recurso = fopen('archivo.txt', 'r');
if ($recurso) { //si el archivo se ha abierto correctamente
    while (!feof($recurso)) { //mientras el final del archivo
        //no se encuentre
        $buffer = fgets($recurso, 4096); //4096-1 es el número
        //máximo de caracteres por cada línea, es decir el tamaño
        //del buffer-1
        echo $buffer."<br />";
    }
}
fclose($recurso);

?>

```

Da como resultado:

"¡Hola!"  
"Señora ESTEFANIA."

 La función **feof()** devuelve true si se encuentra al final del archivo.

Si quiere mostrar las líneas de cinco en cinco caracteres:

```

<?php

$recurso = fopen('archivo.txt', 'r');
if ($recurso) {
    while (!feof($recurso)) { //mientras el final del archivo
        //no se encuentre
        $buffer = fgets($recurso, 6); //6-1 es el número máximo
        //de caracteres por cada línea
        echo $buffer."<br />";
    }
}
fclose($recurso);

?>

```

Da como resultado:

```

"¡Hol
a!"""
Señor
a EST
EFANI
A."

```

- La función **fread()** permite leer un archivo completo y devolver su contenido en una cadena de caracteres. Toma como parámetro el recurso, y como opción, la longitud máxima en bytes.

Por ejemplo:

```

<?php

$recurso = fopen('archivo.txt', 'r');
if ($recurso) {
    $contenido = fread($recurso, 10000); //contenido limitado a
                                         //10000-1 caracteres
    echo $contenido;
}
fclose($recurso);

?>

```

Da como resultado:

```

"¡Hola!" "Señora ESTEFANIA."

```

Observe que las cadenas "¡Hola!" y "Señora ESTEFANIA." están en la misma línea, ya que el salto de línea del

archivo de texto no corresponde al salto de línea HTML (<br />).

- La función **fwrite()** permite escribir en un archivo. Toma como parámetros el recurso y una cadena de caracteres que se va a insertar en el archivo.

Por ejemplo:

```
<?php

$recurso = fopen('archivo.txt', 'w');
if ($recurso) {
    fwrite($recurso, 'Hola Señora LUNA.');
}
fclose($recurso);

?>
```

Da como resultado en el archivo archivo.txt:

Hola Señora LUNA.

Si escribe:

```
<?php

$recurso = fopen('archivo.txt', 'w');
if ($recurso) {
    fwrite($recurso, 'Hola ');
    fwrite($recurso, 'Señora LUNA.');
}
fclose($recurso);

?>
```

No cambia nada. El archivo va a contener siempre Hola Señora LUNA.

Si quiere escribir Hola en una línea y Señora LUNA en otra, debe insertar un salto de línea después de Hola. El salto de línea en formato texto es: \r\n. Se escribe con la constante PHP\_EOL en PHP.

Por tanto:

```
<?php

$recurso = fopen('archivo.txt', 'w');
if ($recurso) {
    fwrite($recurso, 'Hola'.PHP_EOL);
    fwrite($recurso, 'Señora LUNA.');
}
fclose($recurso);
```

```
?>
```

Da como resultado en el archivo archivo.txt:

Hola

Señora LUNA.

Si ahora quiere añadir texto, abra el archivo en modo añadir.

Por ejemplo:

```
<?php  
  
$recurso = fopen('archivo.txt', 'a');  
if ($recurso) {  
    fwrite($recurso, ' Adiós.');//añadido  
}  
fclose($recurso);  
  
?>
```

Da como resultado en el archivo archivo.txt:

Hola

Señora LUNA. Adiós.

- La función **rewind()** permite colocar el cursor al principio del archivo. Esto tiene el efecto de volver a escribir sobre el texto existente.

Por ejemplo:

```
<?php  
  
$recurso = fopen('archivo.txt', 'r+');  
if ($recurso) {  
    fwrite($recurso, 'Adiós Señora LUNA.');//añadido  
    rewind($recurso); // coloca el cursor al principio  
    fwrite($recurso, 'Hasta pronto ');//añadido  
}  
fclose($recurso);  
  
?>
```

Da como resultado en el archivo archivo.txt:

Hasta pronto Señora LUNA.

La cadena de caracteres "Adiós" se sustituye por la cadena de caracteres "Hasta pronto", ya que la función **rewind()** va a colocar el cursor de escritura al principio del archivo antes de escribir "Hasta pronto".

Tenga en cuenta que los caracteres que se han escrito sustituyen a aquellos que ya están presentes.

- La función **fputs()** permite escribir una línea en un archivo. Toma como parámetros el recurso y la cadena de caracteres que va a escribir.

Por ejemplo:

```
<?php

$recurso = fopen('archivo.txt', 'r+');
if ($recurso) {
    fputs($recurso, 'Adiós');
    fputs($recurso, ' Señora LUNA.');
}
fclose($recurso);

?>
```

Da como resultado en el archivo archivo.txt:

Adiós Señora LUNA.

- Si después de las instrucciones **fgets** o **fputs** quiere volver al principio del archivo, utilice la función **fseek()**. Esta función toma como parámetros el recurso y la posición en el archivo.

Por ejemplo:

```
<?php

$recurso = fopen('archivo.txt', 'r+');
if ($recurso) {
    fputs($recurso, 'Adiós');
    fseek($recurso, 0); //vuelve a colocar el cursor al principio del archivo
    fputs($recurso, 'Señora LUNA.');
}
fclose($recurso);

?>
```

Da como resultado en el archivo archivo.txt:

Señora LUNA.

- Para terminar, la función **ftell()** permite conocer la posición actual del cursor. Esta función toma como parámetro el recurso.

Por ejemplo:

```
<?php

$recurso = fopen('archivo.txt', 'r+');
if ($recurso) {
    fputs($recurso, 'Adiós');
    fputs($recurso, 'Señora LUNA.');
    echo ftell($recurso);
}
fclose($recurso);

?>
```

Da como resultado:

16

Las cadenas de caracteres 'Adiós' y 'Señora LUNA' contienen 16 caracteres.

## 6. Concurrencia

Cuando tiene un archivo de texto en un servidor, puede que varias personas escriban en él simultáneamente, lo que podría causar un error.

Para evitar este problema, existe la función **flock()**, que toma como parámetro el recurso y una constante que establece el tipo de bloqueo solicitado.

Un bloqueo puede significar que está utilizando el archivo. Hay cuatro tipos de bloqueo:

1. LOCK\_SH para adquirir un bloqueo compartido (modo de lectura).
2. LOCK\_EX para adquirir un bloqueo exclusivo (modo de escritura).
3. LOCK\_UN para liberar un bloqueo (compartido o exclusivo).
4. LOCK\_NB si quiere que flock() no se bloquee durante el bloqueo (no es compatible en Windows). Esta opción se utiliza junto con (O binario |) LOCK\_EX o LOCK\_SH y la función flock() devuelve un error si el archivo ya está bloqueado.

Esta función reenvía **true** en caso de éxito y **false** en caso contrario.

Por ejemplo:

```
<?php

$recurso = fopen('archivo.txt', 'r+');
if (flock($recurso, LOCK_EX)) { // pone un bloqueo exclusivo
    fwrite($recurso, "El bloqueo está puesto.");
    flock($recurso, LOCK_UN); // libera el bloqueo
```

```

} else {
    echo "¡Imposible bloquear el archivo!";
}
fclose($recurso);

?>

```

Da como resultado en el archivo de texto:

El bloqueo está puesto.

## 7. Manipulación de archivos

- La función **copy()** permite volver a copiar un archivo. Esta función toma como parámetro una cadena de caracteres que contiene el archivo de origen y otra cadena de caracteres que contiene el archivo de destino.

Por ejemplo:

```

<?php

copy("archivo.txt", "prueba.txt");

?>

```

Vuelve a copiar el archivo "archivo.txt" en el archivo "prueba.txt" del mismo directorio.

- La función **file\_exists()** permite probar si hay un archivo o una carpeta. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del archivo o de la carpeta.

Por ejemplo:

```

<?php

$archivo = 'archivo.txt';
if(file_exists($archivo)){
    echo "Este archivo existe.";
}
else {
    echo "Este archivo no existe.";
}

?>

```

Da como resultado en el archivo de texto:

Este archivo existe.

- La función **is\_file()** también permite probar si hay un archivo, pero no funciona con directorios. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del archivo.

Por ejemplo:

```
<?php

$archivo = 'archivo.txt';
if(is_file($archivo)){
    echo "Este archivo existe.";
}
else {
    echo "Este archivo no existe.";
}

?>
```

Da como resultado en el archivo de texto:

Este archivo existe.

- La función **is\_executable()** permite probar si se puede ejecutar el archivo. Toma como parámetro una cadena de caracteres que contiene el nombre del archivo.

Por ejemplo:

```
<?php

$archivo = 'archivo.txt';
if(is_executable($archivo)){
    echo "Este archivo es ejecutable.";
}
else {
    echo "Este archivo no es ejecutable.";
}

?>
```

Da como resultado en el archivo de texto:

Este archivo no es ejecutable.

- La función **touch()** permite crear un archivo. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del archivo. Si ya existe el archivo, solo cambia la fecha de su última modificación.

Por ejemplo:

```
<?php  
    touch("archivo.txt");  
  
?>
```

crea el archivo "archivo.txt" si no existe.

- La función **unlink()** permite eliminar un archivo. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del archivo. Es importante que compruebe si el archivo existe antes de eliminarlo; de lo contrario, PHP generará un error.

Por ejemplo:

```
<?php  
  
if(file_exists("archivo.txt")){  
    unlink("archivo.txt");  
}  
  
?>
```

elimina el archivo "archivo.txt" si existe.

- La función **rename()** permite volver a nombrar un archivo. Esta función toma como parámetro una cadena de caracteres que contiene el antiguo nombre del archivo y otra cadena de caracteres que contiene su nuevo nombre. Es importante que compruebe si el archivo existe antes de volver a nombrarlo; de lo contrario, PHP generará un error.

Por ejemplo:

```
<?php  
  
if(file_exists("archivo.txt")){  
    rename("archivo.txt","nuevo_archivo.txt");  
}  
  
?>
```

vuelve a nombrar el archivo "archivo.txt" en "nuevo\_archivo.txt" si existe.

- La función **filesize()** devuelve el tamaño del archivo. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del archivo.

Por ejemplo:

```
<?php  
  
echo filesize("archivo.txt");
```

Da como resultado:

23

## 8. Manipulación de directorios

Esta parte utiliza conceptos de POO (Programación Orientada Objetos). En el capítulo El objeto se explican los requisitos para entender la sintaxis.

- La función **dir()** devuelve una instancia de la clase Directory. Es decir, esta función va a poner el cursor en un directorio lo que permitirá que a continuación pueda leerse ese directorio. Esta función toma como parámetro una cadena de caracteres que contiene la ruta del directorio.

Por ejemplo, si quiere leer los archivos contenidos en el directorio C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb:

```
<?php

$directory = dir("/Program Files/EasyPHP-DevServer-13.1VC11\data\localweb");
echo "Cursor: " . $directory->handle . "<br />";
echo "Ruta: " . $directory->path . "<br />";
while ($entry = $directory->read()) {
    echo $entry . "<br />";
}
$directory->close();

?>
```

Da como resultado:

```
Cursor: Recurso id #3
Ruta: /Program Files/EasyPHP-DevServer-13.1VC11\data\localweb
.
..
archivo.txt
prueba.php
```

La función **dir()** crea un objeto \$directory que contiene las propiedades handle y path, que tienen como valor la ruta de la carpeta. En el próximo capítulo veremos las nociones de objeto y propiedad. Por ahora, recuerde que existe el método **read()**, que permite leer los archivos de un directorio.

- La función **is\_dir()** devuelve **true** si el directorio existe y **false** si no. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del directorio.

Por ejemplo:

```

<?php

if (is_dir("/Program Files/EasyPHP-DevServer-13.1VC11\data\localweb")) {
    echo "La carpeta existe.";
}
else {
    echo "La carpeta no existe.";
}

?>

```

Da como resultado:

La carpeta existe.

- La función **opendir()** permite abrir un directorio. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del directorio. Devuelve un cursor a su directorio. Esta función se asocia con **readdir()**, que permite leer todos los archivos y los subdirectorios de un directorio.

Por ejemplo:

```

<?php

if ($cursor = opendir('.')) { //apertura del directorio actual
//C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb
    while ($archivo = readdir($cursor)) { //mientras exista
        //un archivo en el directorio
        if ($archivo != "." && $archivo != "..") {
            //No mostrar los directorios . y .. de Windows
            echo "$archivo <br />";
        }
    }
    closedir($cursor);
}

?>

```

Da como resultado:

archivo.txt  
prueba.php

Es decir el contenido del directorio C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb) sin las carpetas . y .. propias de Windows.

- La función **filetype()** devuelve el tipo de contenido de un directorio, es decir, un archivo o un subdirectorio. Esta función toma como parámetro una cadena de caracteres que contiene el nombre del directorio o del archivo.

Por ejemplo, si en nuestro directorio localweb tenemos los dos archivos prueba.php y archivo.txt y un subdirectorio llamado imagen:

```
<?php

if ($cursor = opendir('.')) { //apertura del directorio actual
//C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb
    while ($archivo = readdir($cursor)) { //mientras exista
        //un archivo en el directorio
        if ($archivo != ".." && $archivo != ".") {
            //No mostrar los directorios . y .. de Windows
            echo $archivo." de tipo ".$archivo.".filetype($archivo)."  
";
        }
    }
    closedir($cursor);
}

?>
```

Da como resultado:

```
archivo.txt de tipo file
imagen de tipo dir
prueba.php de tipo file
```

- La función **glob()** devuelve una tabla que contiene todos los archivos de un directorio respetando la máscara. Esta función toma como parámetro una cadena de caracteres que contiene la máscara de los archivos que hay que devolver.

Por ejemplo, si en nuestro directorio localweb tenemos los dos archivos prueba.php y archivo.txt y un subdirectorio llamado imagen:

```
<?php

$archivo = glob('/*.txt');
print_r($archivo);

?>
```

Da como resultado:

```
Array ( [0] => ./archivo.txt)
```

La máscara `/*.txt` son todos los archivos de texto del directorio actual.

Si quiere buscar todos los archivos `.jpg` del directorio `imagen` que se encuentra en el directorio actual, es decir `localweb`, escriba el siguiente código:

```
<?php  
  
$archivo = glob('./imagen/*.jpg');  
print_r($archivo);  
  
?>
```

- A diferencia de Windows, la función `glob` distingue entre mayúsculas y minúsculas; por tanto, si tiene archivos con una extensión JPG, el código anterior no los encontrará.

- La función `getcwd()` devuelve el directorio actual.
- La función `chdir()` permite desplazarse en un directorio.

Por ejemplo, si en el directorio localweb hay dos archivos prueba.php y archivo.txt y un subdirectorio llamado imagen:

```
<?php  
  
echo getcwd()."<br />";  
chdir('imagen');  
echo getcwd();  
  
?>
```

Da como resultado:

```
C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb  
C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\imagen
```

- La función `mkdir()` permite crear un directorio.

Por ejemplo, si quiere crear un directorio css en nuestro directorio localweb:

```
<?php  
  
echo getcwd()."<br />";  
mkdir('css'); //creación del directorio css  
chdir('css'); //desplazamiento por el directorio css  
echo getcwd();  
  
?>
```

Da como resultado:

```
C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb  
C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\css
```

- La función **rmdir()** permite eliminar un directorio.

Por ejemplo, si quiere eliminar el directorio css en el directorio localweb:

```
<?php
echo getcwd()."<br />";
rmdir('css'); //Eliminación del directorio css
echo "Eliminación realizada.";
?>
```

Da como resultado:

```
C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\
Eliminación realizada.
```

- La función **dirname()** devuelve la ruta matriz del nombre del directorio, una cadena de caracteres que se ha pasado como parámetro.

Por ejemplo:

```
<?php
$ruta = "/EasyPHP-DevServer-13.1VC11\data\localweb\imagen";
$directorio = dirname($ruta);
echo $directorio;
?>
```

Da como resultado:

```
/EasyPHP-DevServer-13.1VC11\data\localweb
```

- La función **pathinfo()** devuelve la ruta del nombre del directorio, una cadena de caracteres que se ha pasado como parámetro.

Por ejemplo:

```
<?php
$tabla_info = pathinfo("/EasyPHP-DevServer-13.1VC11\data\localweb");
echo $tabla_info["dirname"]; //ruta
echo $tabla_info["basename"]; //directorio
echo $tabla_info["filename"]; //archivo
```

?>

Da como resultado:

/EasyPHP-DevServer-13.1VC11/data

localweb

localweb

## Los includes

La función **include()** es muy útil porque permite llamar a otra página PHP en una página PHP. Si marca **include('funciones.php')**, equivale a pegar el código contenido en la página funciones.php en el lugar donde llama a la función **include()**. Por tanto, esta función recibe como parámetro el nombre de la página PHP que va a incluir.

Por ejemplo, supongamos que tiene una página PHP llamada variable.php que contiene el siguiente código:

```
<?php  
  
$apellido = "López";  
$nombre = "Roberto";  
  
?>
```

En su página actual PHP, se llama a include de la siguiente manera:

```
<?php  
  
include("variable.php");  
echo $nombre." ".$apellido;  
  
?>
```

Da como resultado:

Roberto López

Si su archivo variable.php se encuentra en el directorio inc, el código para llamar a esta página es:

```
<?php  
  
include("inc/variable.php");  
echo $nombre." ".$apellido;  
  
?>
```

La página PHP se puede especificar con una ruta relativa o absoluta. Una ruta absoluta se marca desde la raíz del disco (por ejemplo, C:\Program Files\easy PHP), mientras que una ruta relativa se marca desde el lugar donde se encuentra su archivo PHP. En el próximo capítulo verá que el archivo **php.ini** incluye la directiva **include\_path** que contiene la ruta de búsqueda de los archivos de inclusión.

El archivo que va a incluir puede tener la extensión inc y así recurrir a la variable.inc en lugar de la variable.php. También puede contener solo HTML. Si tiene que insertar un menú en todas las páginas de su sitio Web, cree un archivo menu.php que contenga su menú en HTML. Haga un include ("menu.php") en todas las páginas de su sitio Web. Esto es muy importante para mantener el sitio; si realiza una modificación en el menú, bastará con que la defina una sola vez para que dicha modificación se visualice en todas partes.

- La función **include\_once()** evita que se repita varias veces la inclusión. Puede ocurrir que su archivo include se llame involuntariamente en otro archivo include. Su sintaxis es la misma que la de la función **include()**.
- La función **require()** es similar a la función **include()**, con una diferencia. La función **require()** puede causar un grave error si no consigue ejecutar el código del archivo que pasa como parámetro, mientras que la función **include()** solo causa un aviso (warning). Su sintaxis es la misma que en include, es decir: **require ("página\_includephp")**.
- La función **require\_once()** evita que se repita varias veces la inclusión de tipo require.

## Ejercicios

### 1. Enunciados

#### **Ejercicio 1 (fácil): creación de un archivo contador de páginas**

Cree un archivo de texto que almacene el número de veces que se visita una página.

#### **Ejercicio 2 (medio): creación de un archivo de información de imágenes**

Mueva tres imágenes a un directorio de imágenes y cree una página PHP que cree un archivo de texto que contenga el nombre y el tamaño de estas imágenes. A continuación, copie estas imágenes en un directorio de archivo que esté al mismo nivel que el directorio de imágenes.

#### **Ejercicio 3 (difícil): creación de registro de seguimiento**

Cree una página que escriba en un archivo log.txt la fecha y la hora actual y que muestre en microsegundos el tiempo que se tarda en mover tres imágenes del directorio de imágenes al directorio de archivo.

### 2. Soluciones

#### Solución del ejercicio 1

Cree un archivo contador.txt y ejecute el siguiente código:

```
<?php

$recurso = fopen('contador.txt', 'r+');

$nb_vistas = fgets($recurso); // Lectura de la primera línea que
                             // contiene el número de páginas visitadas
if ($nb_vistas == "") { //comprueba si el archivo no contiene aún
    //número
    $nb_vistas = 0;
}
$nb_vistas++; // Aumento en 1 del número de páginas vistas
fseek($recurso, 0); // el cursor se pone al principio del archivo
fputs($recurso, $nb_vistas); // escritura del nuevo número
de páginas visitadas

fclose($recurso); // Cierre del archivo
echo 'Se ha visitado esta página ' . $nb_vistas. ' veces.';
?>
```

#### Solución del ejercicio 2

```
<?php

$recurso = fopen('archivo_imagen.txt', 'w+'); //creación del archivo
                                               //de texto si no existe.
if ($recurso) {
    fputs($recurso, 'nombre');
    fputs($recurso, 'tamaño.' . PHP_EOL);
    if ($cursor = opendir('./imagenes')) { //apertura del directorio

//((C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\imagen)
        while ($archivo = readdir($cursor)) { //mientras exista un
                                         //archivo en el directorio
            if ($archivo != ".." && $archivo != "...") {
                fputs($recurso, $archivo. ' ');
                //inserción del
                //nombre del archivo
                fputs($recurso, filesize('./imagenes/' . $archivo). ' ');
                bytes' . PHP_EOL); //inserción del tamaño del archivo
            }
        }
        closedir($cursor);
    }
}
fclose($recurso);

if (! is_dir("./archivo")) { //comprueba si el directorio
//ya no existe
    mkdir('archivo'); //creación del directorio archivo
}
if ($cursor = opendir('./imagenes')) { //apertura del directorio
                                         //imagen
//((C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\imagen)
    while ($archivo = readdir($cursor)) { //mientras exista
//un archivo en el directorio
        if ($archivo != ".." && $archivo != "...") {
            copy("./imagenes/" . $archivo, "./archivo/" . $archivo);
        }
    }
    closedir($cursor);
}

?>
```

#### Solución del ejercicio 3

```
<?php

$recurso = fopen('log.txt', 'w+'); //creación del archivo de texto si
                               //no existe.
if ($recurso) {
    fputs($recurso, 'fecha y hora actual:');
    fputs($recurso, date('d.m.Y G:i:s') . PHP_EOL); //escritura de la fecha
    //y de la hora actual
?>
```

```

}

fclose($recurso);

if (! is_dir("./archivo")) { //prueba si el directorio
//no existe todavía
mkdir('archivo'); //creación del directorio archivo
}
$inicio_tiempos = (float)microtime();
if ($cursor = opendir('./imagenes')) { //apertura del directorio imagen
//C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\imagen
while ($archivo = readdir($cursor)) { //mientras exista
//un archivo en el directorio
if ($archivo != "." && $archivo != "..") {
    copy("./imagenes/".$archivo,'./archivo/'.$archivo);
//copia la imagen en la carpeta archivo
    unlink("./imagenes/".$archivo); //eliminar la imagen en el
//directorío imágenes
}
}
closedir($cursor);
}
$fin_tiempos = (float)microtime();
$duracion_tiempos = $fin_tiempos - $inicio_tiempos; //cálculo de la duración
//del desplazamiento de imágenes en microsegundos
echo "La duración para desplazar estas imágenes es de:
".$duracion_tiempos." microsegundos.';

?>

```

# Las variables superglobales

## 1. \$GLOBALS

En el capítulo Funciones y estructuras de control - Las funciones, hemos visto que hay variables locales y globales. Estas variables tienen un alcance determinado dependiendo de dónde se declaran.

En PHP existe la tabla **\$GLOBALS** o "superglobal" que contiene valores válidos, sea cual sea su alcance, es decir, en cualquier ubicación de la página PHP. Esta tabla tiene como índice el nombre de las variables, y como valor, el valor de las variables.

Por ejemplo:

```
<?php

$nombre = "Juan";
$apellido = "Gómez";

function concatena() {
    $GLOBALS['nombre'] = $GLOBALS['nombre'] . " " . $GLOBALS['apellido'];
}

concatena(); //concatena el nombre y el apellido
echo $nombre;

?>
```

Da como resultado:

Juan Gómez

Si ha escrito en la función **\$nombre = \$nombre . " " . \$apellido;**, PHP generará un error debido a que estas variables no tienen el alcance necesario para que la función las pueda conocer.

 Puede pensar que resulta más sencillo declarar solo las variables superglobales, pero no es así. Esto evita muchas confusiones a la hora de declarar las variables en modo local y sobre todo ahorra mucha memoria si muchos usuarios visitan su sitio Web de manera simultánea; cada variable **\$GLOBALS** ocupará memoria en su servidor y correrá el riesgo de saturarse. Si utiliza variables locales, PHP liberará la memoria que ya no se utiliza.

Más adelante veremos que existen otras variables superglobales. Se escriben siempre con mayúsculas, en forma de tabla y son válidas en todas las páginas PHP.

## 2. \$\_SERVER

**\$\_SERVER** es una tabla que se crea automáticamente y que contiene información, como rutas del script, encabezamientos, etc.

Veamos parte de la información más útil:

- **`$_SERVER['REMOTE_ADDR']`**: permite conocer la dirección IP del usuario que ha solicitado la página.

Por ejemplo:

```
<?php  
  
echo $_SERVER['REMOTE_ADDR'];  
  
?>
```

Da como resultado:

127.0.0.1

Si está trabajando con su equipo y su servidor Web crea la dirección `http://127.0.0.1` y está ejecutando esta página PHP, **`$_SERVER['REMOTE_ADDR']`** reenvía esta misma dirección.

- **`$_SERVER['PHP_SELF']`**: permite conocer el nombre del script que se utiliza actualmente.

Por ejemplo:

```
<?php  
  
echo $_SERVER['PHP_SELF'];  
  
?>
```

Da como resultado:

/prueba.php

La página PHP que contiene este código se llama prueba.php.

- **`$_SERVER['REQUEST_METHOD']`**: permite conocer el método de consulta que se utiliza.

Estos métodos son: GET, POST, PUT y HEAD, y se explicarán más adelante.

Por ejemplo:

```
<?php  
  
echo $_SERVER['REQUEST_METHOD'];  
  
?>
```

Da como resultado:

GET

- **`$_SERVER['QUERY_STRING']`**: permite conocer la cadena de consulta que se utiliza, es decir, las variables y los valores que pasan por la URL.

Por ejemplo, si llama a la página PHP pasando como argumento una variable `var` igual a `1`, es decir, `http://127.0.0.1/prueba.php?var=1`:

```
<?php  
echo $_SERVER['QUERY_STRING'];  
?>
```

Da como resultado:

`var=1`

- **`$_SERVER['HTTP_USER_AGENT']`**: permite conocer la versión del navegador y sus librerías que se utilizan para mostrar la página actual.

Por ejemplo:

```
<?php  
echo $_SERVER['HTTP_USER_AGENT'];  
?>
```

Da como resultado:

`Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36(KHTML, Like Gecko)/ Chrome/29.0.1547.76 Safari/537.36`

- **`$_SERVER['HTTP_ACCEPT_LANGUAGE']`**: permite conocer el idioma que utiliza su navegador.

Por ejemplo:

```
<?php  
echo $_SERVER['HTTP_ACCEPT_LANGUAGE'];  
?>
```

Da como resultado:

- **`$_SERVER['HTTP_HOST']`**: permite conocer el nombre del servidor y su dirección IP.

Por ejemplo:

```
<?php
echo $_SERVER['HTTP_HOST'];
?>
```

Da como resultado:

127.0.0.1

Se obtiene el mismo resultado con `$_SERVER[ 'REMOTE_ADDR' ]`. Cuando trabaja en modo local, lo hace como cliente y servidor al mismo tiempo.

### 3. `$_ENV`

Esta tabla contiene las variables de entorno del sistema operativo, como el nombre de usuario y del equipo. Tenga cuidado al comprobar si `php.ini` tiene `variables_order = "EGPCS"`. En el siguiente capítulo explicaremos el contenido de este archivo, al que se accede a través del menú **Configuration - PHP**. A continuación, debe reiniciar el servidor Web una vez que guarde el archivo.

Veamos parte de la información más útil:

- **`$_ENV['USERNAME']`**: permite conocer el nombre del usuario que se conecta a Windows.

Por ejemplo:

```
<?php
echo $_ENV['USERNAME'];
?>
```

Da como resultado:

Juan Gómez

Si se conecta en Windows con este identificador.

- **`$_ENV['COMPUTERNAME']`**: permite conocer el nombre del equipo de la persona que ejecuta esta página PHP.

Por ejemplo:

```
<?php  
echo $_ENV['COMPUTERNAME'];  
?>
```

Da como resultado:

EQUIPO 1

- **\$\_ENV['OS']**: permite conocer el sistema operativo de la persona que ejecuta esta página PHP.

Por ejemplo:

```
<?php  
echo $_ENV['OS'];  
?>
```

Da como resultado:

Windows\_NT

Si introduce el siguiente código, podrá obtener todos los valores de la tabla **\$\_ENV**:

```
<?php  
echo '<pre>';  
print_r($_ENV);  
echo '</pre>';  
?>
```

Da como resultado:

```
Array  
(  
    [ALLUSERSPROFILE] => C:\ProgramData  
    [AMDAPPSDKROOT]=>C:\Program Files (x86)\AMD APP\  
    [APPDATA]=>C:\Users\Olivier\AppData\Roaming  
    etc.  
)
```

## 4. \$\_SESSION

Una sesión es un archivo que se almacena en el servidor de cada una de las personas que se conectan en su sitio Web. Se destruye automáticamente cuando la persona abandona el sitio Web.

**\$\_SESSION** es una tabla asociativa que permite almacenar cualquier valor de cada usuario. Esta tabla es válida en todas las páginas PHP del sitio Web y permite pasar variables de una página a otra. Tenga cuidado con no abusar de ella, porque si se conecta mucha gente en su sitio Web puede llenar la memoria del servidor Web.

Antes de utilizar **\$\_SESSION**, es imprescindible reiniciar la sesión o llamar a una sesión existente en cada página donde quiera utilizar **\$\_SESSION**. Esto se denomina **session\_start()**.

La función **session\_destroy()** sirve para destruir la sesión actual. Tenga cuidado porque esta función no destruye las variables de la sesión asociadas a la sesión actual. Debe utilizar también la función **unset(\$\_SESSION)**. La sesión se destruirá automáticamente cuando el usuario cierre la ventana de su navegador.

Por ejemplo, el siguiente código en la página prueba.php:

```
<?php  
  
session_start();  
$_SESSION['nombre'] = 'Juan';  
  
echo "El nombre en sesión es:". $_SESSION['nombre'];  
  
?>
```

Da como resultado:

El nombre en sesión es:Juan

La página prueba.php contiene un botón que envía a la página mostrar\_sesion.php. En esta página aparece el siguiente código:

```
<?php  
  
session_start();  
  
echo "El nombre en sesión siempre es:". $_SESSION['nombre'];  
  
?>
```

Da como resultado:

El nombre en sesión siempre es:Juan

De hecho, **\$\_SESSION[ 'nombre' ]** queda en la memoria siempre y cuando no cierre su navegador.

Puede utilizar la función **session\_status()**, que devuelve 0 si se desactivan las sesiones, 1 si se activan, pero sin guardar ninguna aquí, y 2 si se activan las sesiones, siempre y cuando se guarde al menos una de ellas.

## 5. \$\_COOKIE

Una cookie es un archivo que contiene información que se almacena en el equipo del visitante. Cada navegador utiliza sus propias cookies. Se pueden almacenar durante varios meses y, por ejemplo, permite mostrar automáticamente su nombre de usuario cuando vuelve a un sitio Web. No almacene información crítica ya que el usuario puede acceder fácilmente a una parte de estos archivos. Por otro lado, si una persona prohíbe las cookies en su navegador, su sitio Web no le funcionará.

**\$\_COOKIE** también es una tabla global; por lo tanto, es válido en todas las páginas de su sitio Web.

Por ejemplo, el siguiente código en la página prueba.php:

```
<?php  
  
$_COOKIE['nombre'] = 'Juan';  
  
echo "El nombre en cookie es:".$_COOKIE['nombre'];  
  
?>
```

Da como resultado:

El nombre en cookie es:Juan

La página prueba.php contiene un botón que envía a la página mostrar\_cookie.php. En esta página el código es:

```
<?php  
  
echo "El nombre en cookie siempre es:".$_cookie['nombre'];  
  
?>
```

Da como resultado:

El nombre en cookie siempre es:Juan

También puede utilizar la siguiente sintaxis para crear una cookie:

```
<?php  
  
setcookie( "nombre" , "Juan" );  
  
// ¡Recargar la página!  
if (isset($_COOKIE['nombre'])) { // comprueba si la cookie está  
presente  
    echo "El nombre en cookie es:".$_COOKIE['nombre'];  
}
```

```
?>
```

Da como resultado:

El nombre en cookie es:Juan

Tenga cuidado al recargar la página para que aparezca la cookie.

La instrucción **setcookie()** permite pasar como argumento el tiempo de expiración de la cookie, es decir, el tiempo necesario antes de que desaparezca de su equipo.

Este tiempo se expresa en segundos con la función **time()**, que reenvía el tiempo actual en segundos y el tiempo en segundos tras el cual su cookie se eliminará. Si no marca un tiempo de expiración, la cookie se destruirá cuando se cierre el navegador.

Por ejemplo:

```
<?php

$tiempo_expiracion = 365 * 24 * 3600; //tiempo en segundos relativo
a 1 año
setcookie("nombre", "Juan", time() + $tiempo_expiracion);

//¡Recargar la página!
if (isset($_COOKIE['nombre'])) { //comprueba si la cookie está
presente
    echo "El nombre en cookie es:". $_COOKIE['nombre'];
}

?>
```

Da como resultado:

El nombre en cookie es:Juan

Esta cookie es válida durante un año en su navegador, salvo que decida eliminar manualmente todas las cookies de su navegador.

De momento hemos visto cómo se almacena una cadena de caracteres o un número. También puede almacenar una cookie en una tabla. Solo tiene que serializar, es decir, transformar la tabla en una cadena de caracteres. Para mostrar los valores nuevamente, deberá deserializar.

Por ejemplo:

```
<?php

$tabla = array('Juan', 'Roberto', 'Ana');
```

```

$tiempo_expiracion = 365 * 24 * 3600; //tiempo en segundos relativo
a 1 año
$cadena_serializada = serialize($tabla); //serialización de la tabla

setcookie("serializa",$cadena_serializada,time()+$tiempo_expiracion);

//;Recargar la página!
if (isset($_COOKIE["serializa"])) { //comprueba si la cookie está
presente
    $nueva_tabla = unserialize($_COOKIE["serializa"]); //deserializa
    print_r($nueva_tabla); //muestra los valores de la tabla
}

?>

```

Da como resultado:

```
Array ( [0] => Juan [1] => Roberto [2] => Ana )
```

## 6. \$\_FILES

**\$\_FILES** es una tabla asociativa que contiene información de archivos que se han transmitido al servidor Web.

Esta tabla contiene el nombre, el tipo, el nombre temporal, el error y el tamaño del archivo que se ha enviado.

Por ejemplo:

Veamos una página elección\_imagen.php que permite elegir una imagen:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio sobre archivos</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
    </head>
<body>
    <form action="upload.php" method="POST"
enctype="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="2097152">
        <p>Seleccione una foto con un tamaño inferior a 2 MB.</p>
        <input type="file" name="photo">
        <br />
        <input type="submit" name="ok" value="Enviar">
    </form>
</body>
</html>

```

Da como resultado:



No olvide el código `enctype="multipart/form-data"` en el formulario; de lo contrario, no se transmitirá el archivo.

Observe también el campo oculto cuyo nombre es `MAX_FILE_SIZE`. Este campo sirve para definir el tamaño máximo del archivo que va a transmitir. Pero tenga cuidado, ya que no todos los navegadores tienen en cuenta este campo; por lo tanto, por seguridad se recomienda configurar este tamaño máximo en el archivo PHP.ini introduciendo **upload\_max\_file-size = 2M**. Si supera este límite, el servidor Web devuelve un error. El archivo PHP.ini también permite configurar otras opciones relativas a archivos que se han transmitido. La propiedad **file\_uploads** permite autorizar o no el envío de archivos. La propiedad **upload\_tmp\_dir** permite definir el directorio temporal que almacena el archivo que se ha transmitido. La propiedad **post\_max\_size** permite definir el tamaño máximo de los datos que ha enviado el formulario (imagen + texto). La propiedad **post\_max\_size** siempre tiene que ser mayor que **upload\_max\_filesize**.

Observe el código de la página upload.php a la que se llama en la acción del formulario y que permite recibir toda la información sobre el archivo que se transmite.

```
<pre><?php print_r($_FILES); ?></pre>
```

En el siguiente ejemplo, supongamos que elige la imagen Hydrangeas.jpg:

```
Array
(
    [photo] => Array
        (
            [name] => Hydrangeas.jpg
            [type] => imagen/pjpeg
            [tmp_name] => C:\Program Files\EasyPHP-DevServer-13.1VC11\
binarias\tmp\php2AO.tmp
            [error] => 0
            [size] => 595284
        )
)
```

Si se produce un error y desea mostrar el tipo de error y a continuación mover el archivo a un directorio específico:

```
<?php
if ($_FILES['photo']['error']) {
```

```

switch ($_FILES['photo']['error']){
    case 1: // UPLOAD_ERR_INI_SIZE
        echo "El tamaño del archivo supera el límite permitido
por el servidor (argumento upload_max_filesize del archivo
php.ini).";
        break;
    case 2: // UPLOAD_ERR_FORM_SIZE
        echo " El tamaño del archivo supera el límite permitido
por el formulario (argumento post_max_size del archivo php.ini).";
        break;
    case 3: // UPLOAD_ERR_PARTIAL
        echo "El envío del archivo se ha interrumpido durante
la transferencia.";
        break;
    case 4: // UPLOAD_ERR_NO_FILE
        echo "El tamaño del archivo que ha enviado es nulo.";
        break;
}
}

else {
//si no hay error entonces $_FILES['nombre del_archivo']['error'] es 0
echo "No hay error en la carga del archivo.<br />";
if ((isset($_FILES['photo']['name'])&&($_FILES['photo']['error'] ==
UPLOAD_ERR_OK)) {
    $ruta_destino = 'archivos/';
    //desplazamiento del archivo del directorio temporal (almacenado
//por defecto) al directorio de destino con la función
//move_uploaded_file($archivo_uploaded,
$carpeta_nombre_archivo_destino)
    move_uploaded_file($_FILES['photo']['tmp_name'],
$ruta_destino.$_FILES['photo']['name']);
    echo "El archivo ".$_FILES['photo']['name']."' se ha copiado
en el directorio archivos";
}
else {
    echo "El archivo no se ha podido copiar en el directorio
archivos.";
}
}

?>

```

Da como resultado:

No hay error en la carga del archivo

El archivo Hydrangeas.jpg se ha copiado en el directorio archivos

# El método GET

## 1. Utilización del método GET

Por ahora ha llamado a sus páginas PHP con la siguiente URL: <http://127.0.0.1/prueba.php>

Cuando instala su página PHP en su servidor Web después de comprar un nombre de dominio, como [www.sitio.es](http://www.sitio.es), obtendrá lo siguiente: <http://www.sitio.es/prueba.php>

Si pasa de una página PHP a otra, se pierde toda la información de la primera página. Para guardar esta información, puede transmitirla a la URL que llama a la segunda página PHP.

Los valores se pasan a la URL de la siguiente manera: <http://127.0.0.1/prueba.php?apellido=Gómez&nombre=Juan>

Añada el signo de interrogación después del nombre de la página PHP, el nombre de la variable, el símbolo = y su valor. Para añadir otras variables con sus valores, añada el símbolo &, el nombre de la variable, el símbolo =, su valor, etc.

Por ejemplo: <http://127.0.0.1/prueba.php?var1=fresa&var2=framboesa&var3= plátano&var4=arándano>

No se recomienda tener más de 256 caracteres en la URL; por lo tanto, este método tiene sus limitaciones.

En el siguiente ejemplo, se crean dos páginas PHP, `get_envia.php` y `get_recibe.php` para transmitir los datos de una página a otra.

El código de la página `get_envia.php` es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejercicio con GET</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <p><a href="get_recibe.php?apellido=Gómez&nombre=Juan">llama la
página get_recibe.php?apellido=Gómez&nombre=Juan</a></p>
  </body>
</html>
```

Esta página contiene un enlace que llama la página `get_recibe.php` con los valores apellido y nombre, así como sus respectivos valores.

Para recuperar el valor de estas variables, se utiliza la tabla asociativa `$_GET`.

Esta tabla tiene el nombre de las variables pasadas en la URL, y como valor el valor de estas variables que se pasan en la URL.

El código de la página `get_recibe.php` es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con GET</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php

echo "El apellido recibido es:". $_GET["apellido"]. "<br />";
echo "El nombre recibido es:". $_GET["nombre"]. "<br />";

?>
</body>
</html>

```

Al hacer clic en el enlace, obtendrá en la URL lo siguiente: [http://127.0.0.1/get\\_recibe.php?apellido=Gómez&nombre=Juan](http://127.0.0.1/get_recibe.php?apellido=Gómez&nombre=Juan)

Da como resultado:

El apellido recibido es:Gómez

El nombre recibido es:Juan

Ahora cambie en la URL de su navegador el valor del nombre y escriba Pedro en lugar de Juan:  
[http://127.0.0.1/get\\_recibe.php?apellido=Gómez&nombre=Pedro](http://127.0.0.1/get_recibe.php?apellido=Gómez&nombre=Pedro)

Da como resultado:

El apellido recibido es:Gómez

El nombre recibido es:Pedro

Cualquier persona que vaya a su sitio Web puede cambiar el valor de estas variables pasadas en la URL, y esto puede ser muy peligroso. Por lo tanto, evite este método de transmisión de datos.

Puede, eso sí, utilizar este método para pasar como argumento la visualización de un mensaje o cualquier cosa que no suponga un riesgo en su aplicación.

## 2. Comprobar la presencia de la variable en la URL

Si no pasa ningún argumento en la URL y llama a su página get\_recibe.php: [http://127.0.0.1/get\\_recibe.php](http://127.0.0.1/get_recibe.php), se producirá el siguiente error:

Notice: Undefined index: apellido in C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\ get\_recibe.php  
on line 14

El apellido recibido es:

Notice: Undefined index: nombre in C:\Program Files\EasyPHP-DevServer-13.1VC11\data\localweb\ get\_recibe.php

on line 15

El nombre recibido es:

De hecho, `$_GET['apellido']` y `$_GET['nombre']` no existen, ya que las variables apellido y nombre no se pasan como argumentos en la URL.

Para evitar este error, debe comprobar si `$_GET['apellido']` y `$_GET['nombre']` están vacíos.

Vamos a utilizar la función **isset()**, que permite comprobar si una variable está vacía o no.

La página `get_recibe.php` se convierte en:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con GET</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php

if (isset($_GET['nombre']) && isset($_GET['apellido'])) {
    echo "El apellido recibido es:".$_GET["apellido"]."<br />";
    echo "El nombre recibido es:".$_GET["nombre"]."<br />";
}
else {
    echo "El nombre o el apellido no se especifica.";
}

?>
</body>
</html>
```

Ahora, si vuelve a comprobar la siguiente URL: `http://127.0.0.1/get_recibe.php`, aparecerá el siguiente mensaje:

El nombre o el apellido no se especifica.

No debe confiar en el usuario, porque es capaz de modificar su URL.

### 3. Comprobar el valor de la variable en la URL

Si la página PHP tiene como argumento una variable que se utiliza en un bucle, puede ser muy peligroso que el usuario cambie este valor.

Por ejemplo, si escribe la siguiente URL: `http://127.0.0.1/get_recibe.php?número=5`

con `get_recibe.php`:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con GET</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php

if (isset($_GET["numero"])) {
    for ($i=1;$i <= $_GET["numero"];$i++) {
        echo "<?Hola!?" . "<br />" ;
    }
}
else {
    echo "El número no se especifica." ;
}

?>
</body>
</html>

```

Da como resultado:

```

iHola!
iHola!
iHola!
iHola!
iHola!

```

Ahora bien, si alguien escribe en la URL `número=azerty`, el código no muestra nada. Pero si escribe `número=121654165131321`, su código va a intentar hacer el bucle 121654165131321 veces y esto puede provocar un error o bloquear su sitio Web.

Para evitar que esto ocurra, debe convertir `$_GET["numero"]` en numérico y comprobar que está dentro de un cierto rango de valores.

El código de la página `get_recibe.php` se convierte en:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con GET</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php

```

```

if (isset($_GET["numero"])) { //comprueba que el argumento número
    //está especificado
    $numero = (int)$_GET["numero"]; //transformación en numérico
    if ($numero > 0 && $numero < 100) { //número comprendido
entre 1 y 99
        for ($i=1;$i <= $_GET["numero"];$i++) {
            echo ";Hola!."<br />";
        }
    }
} else {
    echo "El número no está especificado.";
}
}
?>
</body>
</html>

```

## 4. Información complementaria

### a. Argumentos con el mismo nombre

Si pasa en la URL dos argumentos con el mismo nombre, **\$\_GET** va a tener en cuenta el último nombre.

Por ejemplo:

La página `get_recibe.php` con la URL: `http://127.0.0.1/get_recibe.php?nombre=Juan&nombre=Roberto`

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con GET</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php

if (isset($_GET["nombre"])) { //comprueba que el argumento nombre
    //está especificado
    echo "El nombre es:". $_GET["nombre"];
}
?>
</body>
</html>

```

Da como resultado:

El nombre es:Roberto

## b. Argumentos de tipo tabla

PHP permite pasar en la URL los argumentos en forma de tabla.

Es posible que no quiera introducir un índice, como en este ejemplo: `http://127.0.0.1/get_recibe.php?dato[ ]=Juan&dato[ ]=Gómez`

En este caso, las claves de la tabla son 0 y 1 respectivamente.

También puede introducir un índice, por ejemplo: `http://127.0.0.1/get_recibe.php?dato[nombre]=Juan&dato[apellido]= Gómez`

En este caso, las claves de la tabla son nombre y apellido respectivamente.

Por ejemplo:

La página `get_recibe.php` con la URL: `http://127.0.0.1/get_recibe.php?dato[nombre]=Juan&dato[apellido]= Gómez`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con GET</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <?php

if (isset($_GET["dato"])) { //comprueba que el argumento dato está
                           //especificado
    echo "Los datos son:" ;
    foreach ($_GET["dato"] as $valor) {
        echo $valor." ";
    }
}
?>
</body>
</html>
```

Da como resultado:

Los datos son:Juan Gómez

## c. Argumentos con caracteres especiales

No puede pasar en una URL caracteres especiales, como & o ? o un espacio.

Y no puede escribir, por ejemplo: `http://127.0.0.1/get_recibe.php?argumento=Juan&Gómez`

El servidor Web entenderá que hay dos variables argumento y Gómez, cuando lo que quiere hacer es pasar como argumento Juan&Gómez.

Para realizar esto, debe codificar la URL, es decir, transformar los caracteres especiales en un código que el servidor Web pueda entender.

Hay dos funciones en PHP que permiten hacerlo: **urlencode()** y **rawurlencode()**.

Estas funciones toman como argumento la URL que debe codificar y devuelven la URL sin los caracteres especiales. Se diferencian en que la función **rawurlencode()** transforma los espacios en %20, mientras que la función **urlencode()** los transforma en +.

Por ejemplo:

```
<?php  
  
$argumento = "Juan&Gómez Pablo";  
echo urlencode($argumento). "<br />";  
echo rawurlencode($argumento). "<br />";  
  
?>
```

Da como resultado:

Juan%26Gómez+Pablo

Juan%26Gómez%20Pablo

Para pasar como argumento el valor "Juan&Gómez Pablo", debe escribir la siguiente URL:  
[http://127.0.0.1/get\\_recibe.php?argumento=Juan%26Gómez%20Pablo](http://127.0.0.1/get_recibe.php?argumento=Juan%26Gómez%20Pablo)

Otro ejemplo:

La página get\_envia.php contiene un enlace que llama a la página get\_recibe.php con el argumento Juan&Gómez.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">  
    <head>  
        <title>Ejercicio con GET</title>  
        <meta http-equiv="Content-Type" content="text/html;  
charset=iso-8859-1" />  
    </head>  
    <body>  
        <?php  
        $argumento = "Juan&Gómez"  
        ?>  
        Aquí está el enlace con Juan&Gómez como argumento: <a  
        href="get_recibe.php?argumento=<?php echo urlencode($argumento)?>">  
        get_recibe.php con argumento</a>  
    </body>
```

```
</html>
```

Si observamos el código fuente en el navegador, tenemos:

```
<a href="get_recibe.php?argumento=Juan%26Gómez">  
get_recibe.php con argumento</a>
```

La página get\_recibe.php:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">  
    <head>  
        <title>Ejercicio con GET</title>  
        <meta http-equiv="Content-Type" content="text/html;  
charset=iso-8859-1" />  
    </head>  
    <body>  
        <?php  
        echo "El argumento es:". $_GET[ "argumento" ];  
        ?>  
    </body>  
</html>
```

Da como resultado:

El argumento es:Juan&Gómez

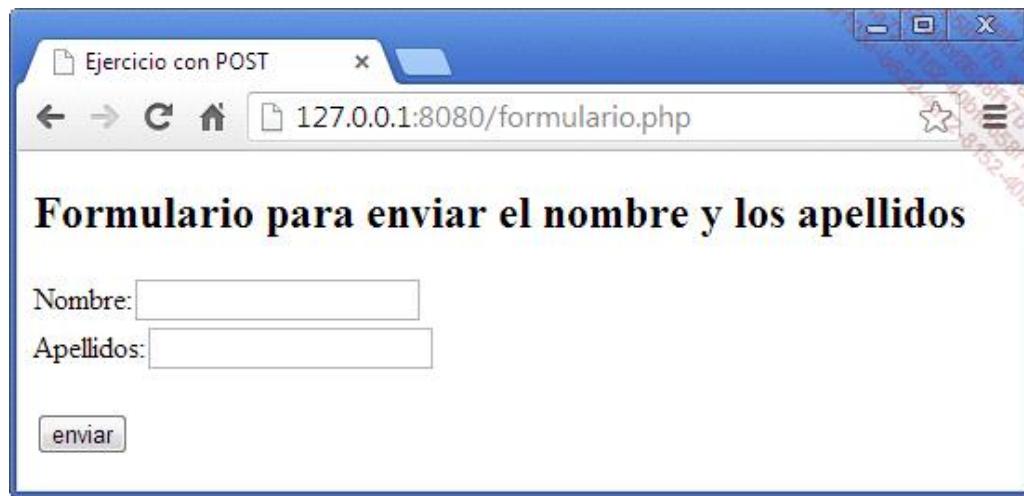
# El método POST

## 1. Utilización del método POST

El método POST transmite datos de una página PHP a otra. A diferencia del método GET, estos datos no están visibles en la URL. De ahí que este método sea el más utilizado.

Todos los datos que están contenidos en un formulario se envían a la otra página PHP a través del método POST y se reciben en una tabla superglobal **`$_POST`**.

Para entenderlo mejor, vamos a crear una página PHP con el nombre formulario.php que contiene un formulario con el nombre y el apellido.



El código de la página formulario.php es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejercicio con POST</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <form action="recibe_post.php" method="POST" name="formulario">
      <h2>Formulario para enviar el nombre y los apellidos</h2>
      Nombre: <input type="text" name="nombre" /><br />
      Apellido: <input type="text" name="apellido" /><br />
      <input type="submit" name="enviar" value="Enviar" />
    </form>
  </body>
</html>
```

Observe que en la etiqueta `<form>` está el atributo `method` con el valor `POST`, que es obligatorio para transmitir los valores que se han introducido en el formulario. Si escribe `method=GET`, los datos que se han introducido se transmiten a la URL.

A continuación, la etiqueta <form> debe contener la acción, es decir, el nombre de la página PHP al que se llama cuando hace clic en **enviar** (que es de tipo submit).

Cuando hace clic en **enviar**, todos los valores de los objetos que están en el formulario se envían a la página `recibe_post.php`, que los recibe en la tabla **\$\_POST**.

El código de la página `recibe_post.php` es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con POST</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <h2>Página de recepción del nombre y apellido</h2>
        Nombre: <?php echo $_POST["nombre"];?><br />
        Apellido: <?php echo $_POST["apellido"];?>
    </body>
</html>
```

Si escribe en la página `formulario.php` Juan en el nombre y Gómez en el apellido, haga clic en **enviar** y obtendrá lo siguiente:

Página de recepción del nombre y apellido

Nombre:Juan

Apellido:Gómez

## 2. Los diferentes elementos del formulario

Vamos a explicar de qué manera se transmiten de una página PHP a otra los elementos de un formulario. En todos los ejemplos siguientes, el código HTML de los elementos del formulario están en la página `formulario.php`, que contiene el siguiente código:

```
<form action="recibe_post.php" method="POST" name="formulario">
    <input type="submit" name="enviar" value="Enviar" />
</form>
```

El código HTML se va a añadir entre la etiqueta <form> y la etiqueta <input>.

El código PHP que permite recibir datos con `$_POST` se va a añadir en la página `recibe_post.php` entre las etiquetas <body> y </body>.

### a. Campo de tipo texto



El campo texto se escribe en HTML de la siguiente manera:

```
<input type="texto" name="campol" value="valor1" />
```

El atributo `value` es opcional. Representa el valor que se muestra en la zona de texto. En todas las zonas del formulario, el atributo `name` es obligatorio, ya que representa la clave de la tabla `$_POST` que permite recuperar el valor de la zona de texto.

El código PHP que permite recibir el valor de la zona de texto es:

```
<?php  
echo $_POST[ "campol" ];  
?>
```

Da como resultado:

valor1

La página completa `formulario.php` es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">  
  <head>  
    <title>Ejemplo con POST</title>  
    <meta http-equiv="Content-Type" content="text/html;  
charset=iso-8859-1" />  
  </head>  
  <body>  
    <form action="recibe_post.php" method="POST" name="formulario">  
      <h2>Formulario para enviar un área de texto</h2>  
      Nombre: <input type="text" name="nombre" value="" /><br />  
      <input type="submit" name="enviar" value="Enviar" />  
    </form>  
  </body>
```

```
</html>
```

La página completa `recibe_post.php` es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejemplo con POST</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
<h2>Página de recepción de la zona de texto</h2>
Nombre: <?php echo $_POST["nombre"];?><br />
</body>
</html>
```

Cuando hace clic en **enviar**, la página `formulario.php` transmite el valor que ha introducido en la zona de texto a la página `recibe_post.php`, que recupera este valor en la tabla **`$_POST`** y lo muestra.

### b. Campo de tipo contraseña



El campo de tipo contraseña se escribe en HTML de la siguiente manera:

```
<input type="password" name="campol" value="valor1" />
```

El atributo `value` es opcional. Representa el valor en la zona de texto.

El código PHP que permite recibir el valor de la zona de texto es:

```
<?php
echo $_POST["campol"];
?>
```

Da como resultado:

valor1

### c. Área de texto



Este campo se escribe en HTML de la siguiente manera:

```
<textarea name="campo1">valor1</textarea>
```

El valor se escribe por defecto entre las etiquetas <textarea> y </textarea>, y no en value.

El código PHP que permite recibir el valor de la zona de texto es:

```
<?php  
echo $_POST[ "campo1" ];  
?>
```

Da como resultado:

valor1

### d. Lista desplegable de elección simple



Este campo se escribe en HTML de la siguiente manera:

```
<select name="pais">
    <option value="F">Francia</option>
    <option value="E">España</option>
    <option value="R">Rusia</option>
</select>
```

El atributo value es opcional. Representa el valor que se ha transmitido a la tabla **\$\_POST**.

El código PHP que permite recibir el valor de la lista desplegable es:

```
<?php
echo $_POST["pais"];
?>
```

Si selecciona España, da como resultado:

E

Si no introduce el valor en la opción, se transmite el valor entre las etiquetas `<option>` y `</option>`.

Por ejemplo, en la página formulario.php:

```
<select name="pais">
    <option>Francia</option>
    <option>España</option>
    <option>Rusia</option>
</select>
```

El código PHP que permite recibir el valor de la lista desplegable es:

```
<?php
```

```
echo $_POST["pais"];
?>
```

Si selecciona España, da como resultado:

España

Puede introducir el atributo `selected="selected"` en la opción que permite mostrar un país por defecto.

Por ejemplo, para mostrar España:

```
<select name="pais">
    <option>Francia</option>
    <option selected="selected">España</option>
    <option>Rusia</option>
</select>
```

#### e. Lista desplegable de elección múltiple



Este campo se escribe en HTML de la siguiente manera:

```
<select name="pais[]" multiple="multiple">
    <option value="E">España</option>
    <option value="F">Francia</option>
    <option value="R">Rusia</option>
    <option value="A">Alemania</option>
</select>
```

El atributo `value` es opcional. Representa el valor que se ha transmitido en la tabla `$_POST`. El atributo `multiple="multiple"` también es obligatorio para indicar que la lista es de tipo múltiple.

El nombre se escribe con corchetes para indicar que el valor que se ha enviado con POST es de tipo tabla. De hecho, como hay muchas opciones, no podrá recuperar estos valores en una variable. Por lo tanto, la tabla se utiliza para recuperar los valores que ha seleccionado.

El código PHP que permite recibir el valor de la lista desplegable es:

```
<?php  
if (isset($_POST["pais"])) { //para que no haya error si  
    //no selecciona ningún país  
    echo "Los países seleccionados son:";  
    print_r ($_POST["pais"]);  
} else {  
    echo "ningún país seleccionado.";  
}  
?>
```

Si selecciona Francia y España, da como resultado:

Los países seleccionados son:Array ( [0] => F [1] => E )

El atributo value no es obligatorio, al igual que en las listas de selección simple:

Por ejemplo, en la página formulario.php:

```
<select name="pais[]" multiple="multiple">  
    <option>Francia</option>  
    <option>España</option>  
    <option>Rusia</option>  
    <option>Alemania</option>  
</select>
```

El código PHP que permite recibir el valor de la lista desplegable es:

```
<?php  
if (isset($_POST["pais"])) { //para que no haya error si  
    //no selecciona ningún país  
    echo "Los países seleccionados son:";  
    foreach ($_POST["pais"] as $valor) {  
        echo $valor . " ";  
    }  
} else {  
    echo "ningún país seleccionado.";  
}  
?>
```

Si selecciona Francia y España, da como resultado:

Los países seleccionados son: Francia España

Puede introducir el atributo `selected="selected"` en varias opciones y así mostrar el país.

Por ejemplo, para mostrar España y Rusia:

```
<select name="pais[]" multiple="multiple">
    <option selected="selected">España</option>
    <option>Francia</option>
    <option selected="selected">Rusia</option>
    <option>Alemania</option>
</select>
```

#### f. Lista de casillas de selección



Este campo se escribe en HTML de la siguiente manera:

```
<input type="checkbox" name="pais[]" value="E" />España<br />
<input type="checkbox" name="pais[]" value="F" />Francia<br />
<input type="checkbox" name="pais[]" value="R" />Rusia<br />
<input type="checkbox" name="pais[]" value="A" />Alemania<br />
```

El atributo `value` es opcional. Representa el valor que se ha transmitido a la tabla `$_POST`.

El nombre se escribe con corchetes para indicar que el valor que se ha enviado con POST es de tipo tabla. De hecho, como hay varias opciones, no se permite recuperar estos valores en una variable. Por lo tanto, se utiliza una tabla para recuperar los valores que ha seleccionado.

El código PHP que permite recibir el valor de la lista desplegable es:

```

<?php
if (isset($_POST["pais"])) { //para que no haya error si
    //no selecciona ningún país
    echo "Los países seleccionados son:";
    print_r ($_POST["pais"]);
} else {
    echo "ningún país seleccionado.";
}
?>

```

Si selecciona Rusia y Alemania, da como resultado:

Los países seleccionados son:Array ( [0] => R [1] => A )

El atributo value no es obligatorio, al igual que en las listas, ya que puede recuperar la elección del usuario con ayuda de las claves de la tabla:

Por ejemplo, en la página formulario.php:

```

<input type="checkbox" name="pais['España']" />España<br />
<input type="checkbox" name="pais[Francia]" />Francia<br />
<input type="checkbox" name="pais['Rusia']" />Rusia<br />
<input type="checkbox" name="pais['Alemania']" />Alemania<br />

```

El código PHP que permite recibir el valor de la lista desplegable es:

```

<?php
if (isset($_POST["pais"])) { //para que no haya error si
    //no selecciona ningún país
    echo "Los países seleccionados son:";
    foreach ($_POST["pais"] as $clave => $valor) {
        echo $clave."->".$valor." ";
    }
} else {
    echo "ningún país seleccionado.";
}
?>

```

Si selecciona Rusia y Alemania, da como resultado:

Los países seleccionados son:'Rusia'->on 'Alemania'->on

Puede introducir el atributo checked="checked" en varias de las casillas de selección que permiten seleccionar el país.

Por ejemplo, para mostrar España y Rusia:

```

<input type="checkbox" name="pais['España']" checked="checked" />España

```

```
checked="checked"/>España<br />
<input type="checkbox" name="pais['Francia']" />Francia<br />
<input type="checkbox" name="pais['Rusia']" checked="checked"/>Rusia<br />
<input type="checkbox" name="pais['Alemania']" />Alemania<br />
```

## g. Lista de botones de opción



Este campo se escribe en HTML de la siguiente manera:

```
<input type="radio" name="pais" value="E" />España<br />
<input type="radio" name="pais" value="F" />Francia<br />
<input type="radio" name="pais" value="R" />Rusia<br />
<input type="radio" name="pais" value="A" />Alemania
```

El valor `value` es obligatorio. Representa el valor que se ha transmitido en la tabla `$_POST` y no existe otra manera de conocer el valor de la opción que ha seleccionado.

El atributo `name` debe ser el mismo en todo el grupo de sus botones de opción.

El código PHP que permite recibir el valor del botón de opción es:

```
<?php
if (isset($_POST["pais"])) { //para que no haya error si
    //no selecciona ningún país
    echo "El país seleccionado es:";
    echo $_POST["pais"];
} else {
    echo "ningún país seleccionado.";
}
?>
```

Si selecciona Rusia, da como resultado:

El país seleccionado es:R

Puede introducir el atributo checked="checked" en un botón de opción para que aparezca marcado un país por defecto. Así el usuario está obligado a elegir un país, ya que no puede eliminar la selección de todos los botones de opción.

Por ejemplo, para mostrar España:

```
<input type="radio" name="pais" value="F" />Francia<br />
<input type="radio" name="pais" value="E" checked="checked"/>España<br />
<input type="radio" name="pais" value="R" />Rusia<br />
<input type="radio" name="pais" value="A" />Alemania
```

## **h. Los campos ocultos**

Este campo se escribe en HTML de la siguiente manera:

```
<input type="hidden" name="campol" value="valor1" />
```

Este campo no aparece en la pantalla. Almacena valores invisibles para el usuario. Atención, el usuario puede mostrar todavía el código fuente de la página Web y ver el valor que se ha almacenado en este campo.

El atributo value es obligatorio si quiere almacenar información en este campo oculto. El atributo name es obligatorio, ya que representa la clave de la tabla **\$\_POST** que permite recuperar el valor de la zona de texto.

El código PHP que permite recibir el valor del campo oculto es:

```
<?php
echo $_POST[ "campol" ];
?>
```

Da como resultado:

valor1

## **i. El botón submit**

Este botón se escribe en HTML de la siguiente manera:

```
<input type="submit" name="boton" value="enviar" />
```

Este campo es obligatorio en el formulario para poder enviar los datos con POST.

El código PHP que permite recibir el valor del botón es:

```
<?php
echo $_POST[ "boton" ];
?>
```

Da como resultado:

enviar

El atributo name no es obligatorio para enviar el formulario. Solo es necesario si hay varios botones de tipo submit. De hecho, puede enviar un formulario con JavaScript.

Ejemplo de página formulario.php con dos botones submit:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejemplo con POST</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <form action="recibe_post.php" method="POST" name="formulario">
      <h2>Formulario de envío de radio-buttons</h2>
      elección del país: <br /><input type="radio" name="pais" value="F"
      />Francia<br />
      <input type="radio" name="pais" value="E" />España<br />
      <input type="radio" name="pais" value="R" />Rusia<br />
      <input type="radio" name="pais" value="a" />Alemania<br /><br />
      <input type="submit" name="enviar" value="enviar" />
      <input type="submit" name="enviar" value="cancelar" />
    </form>
  </body>
</html>
```

La página completa `recibe_post.php` es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejemplo con POST</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <?php
      echo $_POST[ "enviar" ];
    ?>
```

```
?>
</body>
</html>
```

Si hace clic en **enviar**, da como resultado:

enviar

O bien si hace clic en **cancelar**, da como resultado:

cancelar

Esto permite ejecutar diferentes acciones según el botón en el que hace clic.

A continuación mostramos un ejemplo para enviar un formulario en Java-Script:

La página formulario.php se convierte en:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo con POST</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
        <SCRIPT language="JAVASCRIPT">
            function enviar_formulario() {
                document.formulario.action="recibe_post.php";
                document.formulario.submit();
            }
        </SCRIPT>
    </head>
<body>
<form method="POST" name="formulario">
<h2>Formulario de envío de radio-buttons</h2>
elección del país: <br /><input type="radio" name="pais" value="F"
/>Francia<br />
    <input type="radio" name="pais" value="E" />España<br />
    <input type="radio" name="pais" value="R" />Rusia<br />
    <input type="radio" name="pais" value="A" />Alemania<br /><br />
<input type="button" name="enviar" value="enviar"
OnClick="enviar_formulario();"/>
</form>
</body>
</html>
```

En este caso, es un botón normal de tipo button, el que ejecuta la función Java-Script `enviar_formulario()`. Esta función define la acción y presenta el formulario. Esto es muy útil para validar campos antes de enviar el formulario.

El código de la página completa `recibe_post.php` permite saber si se ha hecho clic en un botón de tipo submit:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo con POST</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php
if (isset($_POST["enviar"])) {
    echo "Procede de una página con un botón submit llamado enviado";
}
else {
    echo "Procede de una página con un código javascript";
}
?>
</body>
</html>
```

Da como resultado:

Procede de una página con un código javascript

#### j. El botón reset

Este botón se escribe en HTML de la siguiente manera:

```
<input type="reset" name="boton" value="borrar" />
```

Este botón sirve para borrar todos los valores que se han introducido, comprobado o seleccionado en el formulario.

#### k. Formulario completo

A continuación se muestra el código que contiene todos los controles de entrada posibles de un formulario:

Ejercicio con POST

127.0.0.1:8080/formulario.php

## Formulario general de envío

Nombre:

Contraseña:

Sexo:  Mujer  Hombre  No sabe

Nacionalidad:

Paises que desea visitar:

Francia  
 España  
 Rusia  
 Alemania

Idiomas:

Francés  
 Español  
 Russo  
 Alemán

Comentario:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
    <title>Ejercicio con POST</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
</head>
<body>
<form action="recibe_post.php" method="POST" name="formulario">
    <h2>Formulario general de envío</h2>
    nombre:<input type="text" name="nombre" /><br />
    contraseña:<input type="password" name="password" /><br /> <br />

    sexo:<input type="radio" name="sexo" value="F" />Mujer
        <input type="radio" name="sexo" value="M" />Hombre
        <input type="radio" name="sexo" value="N" checked="checked"/>No sabe<br /><br />
```

```

nacionalidad:<select name="nacionalidad">
    <option value="F">Francesa</option>
    <option value="E">Española</option>
    <option value="R">Rusa</option>
    <option value="A">Alemana</option>
</select><br /><br />

pa&iacute;ses que desa visitar:<br /> <select name="pais[]" multiple="multiple">
    <option>Francia</option>
    <option>España</option>
    <option>Rusia</option>
    <option>Alemania</option>
</select><br /><br />

idiomas:<br /><input type="checkbox" name="idioma[]"
value="F" />Franc&eacute;s<br />
<input type="checkbox" name="idioma[]" value="E" />Espa ol<br />
<input type="checkbox" name="idioma[]" value="R" />Ruso<br />
<input type="checkbox" name="idioma[]" value="A" />Alem n<br />

<input type="hidden" name="campo_oculto" value="campo_oculto" /><br />
Comentario:<br /><textarea name="comentario"></textarea><br />

<input type="submit" name="enviar" value="Enviar" />
<input type="reset" name="anular" value="Borrar" />
</form>
</body>
</html>

```

La p gina PHP recibe\_post.php que permite mostrar todos los datos introducidos es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo con POST</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<h2>P gina de recepc n de las zonas del formulario</h2>
nombre: <?php echo $_POST["nombre"];?><br />
contrase a: <?php echo $_POST["password"];?><br />
sexo: <?php echo $_POST["sexo"];?><br />
nacionalidad: <?php echo $_POST["nacionalidad"];?><br />
pa s a visitar: <?php if (isset($_POST["pais"])) { //para que no haya
//error si no selecciona ning n pa s
        foreach ($_POST["pais"] as $clave => $valor) {
            echo $valor." ";
        }
    } else {
        echo "ning n pa s seleccionado.";
    }?>

```

```
<br />
idiomas:<?php if (isset($_POST["idioma"])) { //para que no
// haya error si no selecciona ningún país
    foreach ($_POST["idioma"] as $clave => $valor) {
        echo $valor." ";
    }
} else {
    echo "ningún idioma seleccionado.";
}>?>
<br />
campo oculto: <?php echo $_POST["campo_oculto"];?><br />
comentario: <?php echo $_POST["comentario"];?><br />
</body>
</html>
```

## Otros métodos

### 1. El método \$\_REQUEST

**\$\_REQUEST** es una tabla asociativa que reagrupa los métodos **\$\_GET**, **\$\_POST** y **\$\_COOKIE**. Esta tabla también es superglobal; por lo tanto, es accesible en todos los scripts PHP. Si los datos se envían con POST, GET o COOKIE, se pueden recuperar con **\$\_REQUEST**.

En el siguiente ejemplo, la página formulario.php contiene dos botones : uno para enviar los datos con POST y el otro con GET.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
    <title>Ejemplo con REQUEST</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    <SCRIPT lenguaje="JAVASCRIPT">
        function enviar_formulario() {
            //llama la página receive_post.php con transmisión de las
            //variables en la URL

            document.location.href="receive_post.php?nombre="+document.formulario.
            nombre.value+"&apellido="+document.formulario.apellido.value;
        }
    </SCRIPT>
</head>
<body>
<form action="receive_post.php" method="POST" name="formulario">
<h2>Formulario para enviar el nombre y los apellidos</h2>
Nombre: <input type="text" name="nombre" /><br />
Apellido: <input type="text" name="apellido" /><br />
<input type="submit" name="enviarPOST" value="Enviar por POST" /> &nbsp;
<input type="button" name="enviarGET" value="Enviar por GET"
onClick="enviar_formulario()"/>
```

```
</form>
</body>
</html>
```

El método `$_REQUEST`, que recupera los datos de la tabla `$_GET` y de la tabla `$_POST`, se puede utilizar para recibir los datos de las dos tablas al mismo tiempo.

La página `recibe_post.php` es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejemplo con REQUEST</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <h2>Recuperación de datos por $_POST y $_GET</h2>
        <?php
            if (isset($_REQUEST["apellido"])) {
                echo "Su apellido es:".$_REQUEST['apellido']."<br />";
            }
            if (isset($_REQUEST["nombre"])) {
                echo "Su nombre es:".$_REQUEST['nombre'];
            }
        ?>
    </body>
</html>
```

Si hace clic en **enviar por GET** o en **enviar por POST**, da como resultado:

Recuperación de datos por \$\_POST y \$\_GET

Su apellido es:MORALES

Su nombre es:David

-  La función `import_request_variables` permite crear de manera automática una variable por cada dato que se ha recuperado por GET, POST o COOKIE, y que está obsoleta desde la versión PHP5.4.

## Zonas con el mismo nombre

Si en un formulario dos o más zonas tienen el mismo nombre, se utilizará la última.

Por ejemplo:

```
<form action="recibe_post.php" method="POST" name="formulario">
<h2>Formulario para enviar el nombre y los apellidos</h2>
Apellido: <input type="text" name="nombre" /><br />
Nombre: <input type="text" name="nombre" /><br />
<input type="submit" name="enviarPOST" value="enviar por POST" />
</form>
```

El valor que se introduce en la zona junto al nombre se transmite por \$ POST, ya que las dos zonas se denominan nombre.

## Varios formularios en la misma página

Si hay varios formularios en la misma página, estos formularios son independientes y cada botón submit envía los valores del propio formulario.

Por ejemplo:

```
<form action="recibe_post.php" method="POST" name="formulario">
<h2>Formulario para enviar el nombre y los apellidos</h2>
Apellido: <input type="text" name=apellido1 /><br />
Nombre: <input type="text" name=nombre1 /><br />
<input type="submit" name="enviar1" value="enviar 1" />
</form>

<form action="recibe_post.php" method="POST" name="formulario2">
<h2>Formulario para enviar el nombre y los apellidos</h2>
Apellido: <input type="text" name=apellido2 /><br />
Nombre: <input type="text" name=nombre2 /><br />
<input type="submit" name="enviar2" value="enviar 2" />
</form>
```

Si pulsa en el botón **enviar1**, solo se envían los valores de los campos **nombre1** y **apellido1**.

# Control de datos y redirección de páginas

## 1. Introducción

Los datos introducidos en un formulario no son seguros. De hecho, los usuarios pueden escribir letras cuando solicita una edad o introducir un código SQL que puede causar problemas en su página web. Esta última acción se denomina inyección SQL. Para evitar que un usuario pueda hacer inoperativo su sitio Web, es necesario proteger los datos introducidos. Se puede comprobar en JavaScript (lado cliente) que los datos introducidos se corresponden con lo que quiere, pero no es el propósito ni es suficiente. También debe codificar las comprobaciones en PHP (lado servidor) antes de guardar los datos en la base de datos.

## 2. Datos obligatorios

Esta prueba consiste en comprobar si el usuario ha llenado un campo del formulario.

Las zonas de tipo texto, textarea, hidden y password se comprueban de la siguiente manera:

```
<?php  
//Suponiendo que el formulario contenga el campo nombre  
if ($_POST['nombre']!='') {  
    echo "Su nombre es".$_POST['nombre'];  
}  
else  
{  
    echo "El nombre no se ha introducido.";  
}  
?>
```

En las áreas de tipo radio-button, checkbox y lista, la comprobación es:

```
<?php  
//Suponiendo que el formulario contenga una lista llamada pais  
if (isset($_POST['pais']) == true) {  
    echo "Los países son:";  
    print_r ($_POST['pais']);  
}  
else  
{  
    echo "El país no se ha seleccionado.";  
}  
?>
```

## 3. Eliminación de espacios no deseados

Algunas veces los usuarios añaden espacios al final de su apellido y luego lo recuperan con el espacio con **\$\_POST** y guardan este valor en la base de datos. Esto puede causar algunos problemas si intenta comparar los apellidos. De hecho, "GÓMEZ" es distinto de " GÓMEZ ".

Para evitar esto, añada la función **trim()**, que elimina los espacios y a continuación el valor.

```
<?php
//Suponiendo que el formulario contenga el campo apellido
if (trim($_POST['apellido']) != '') {
    echo "Su apellido es:".trim($_POST['apellido']);
}
?>
```

## 4. Longitud máxima

Esta vez el problema es la longitud de la cadena de caracteres que recupera con POST. Por ejemplo, el apellido no debe contener más de 20 caracteres, ya que la base de datos no lo acepta. Puede realizar una comprobación inicial introduciendo el atributo `maxlength=20` en la etiqueta `<input type= "text" />`, pero no es suficiente. También debe realizar el control del servidor en PHP:

```
<?php
//Suponiendo que el formulario contenga un campo apellido
$apellido = trim($_POST['apellido']);
if (strlen($apellido) <= 20) {
    echo "Su apellido es:".$apellido;
}
else
{
    echo "El apellido contiene más de 20 caracteres.";
}
?>
```

## 5. Caracteres permitidos

Una vez que haya comprobado que sus datos no están vacíos y que la longitud es correcta, debe revisar que no contengan caracteres no permitidos. Por ejemplo, la edad no debe contener letras y el correo electrónico debe incluir un punto (.) y @. Para realizar esto correctamente, utilice las expresiones regulares que hemos visto con anterioridad.

Por ejemplo, para comprobar que la contraseña contiene letras y cifras entre 4 y 8 caracteres:

```
<?php

//Suponiendo que el formulario contenga un área password
$password = $_POST['password'];
if (!preg_match('`^[[[:alnum:]]]{4,8}$`',$password)) {
    echo "La contraseña no es válida.";
}
else
{
    echo "La contraseña es válida.";
}
```

```
?>
```

Por ejemplo, para comprobar si un número de teléfono es válido:

```
<?php

//Suponiendo que el formulario contenga una zona teléfono
$telefono = $_POST[telefono];
if (preg_match("#^0[1-9]([- ]?[0-9]{2}){4}$#", $telefono))
{
    echo 'El número es válido.';
}
else
{
    echo "El número no es válido.";
}

?>
```

Por ejemplo, para comprobar si una fecha es válida:

```
<?php

//Suponiendo que el formulario contenga un área fecha
$fecha = $_POST['fecha'];
/* Divide la fecha en tres partes y asigna los días a $dd,
los meses a $mm y el año a $yyyy */
list($dd, $mm, $yyyy) = explode("/", $fecha);
/* Validación de la fecha con la función checkdate*/
if(! checkdate($mm, $dd, $yyyy))
{
    echo "Fecha no válida";
}
else {
    echo "Fecha válida";
}
?>
```

## 6. Magic quotes

Los datos que recupera por `$_GET` o `$_POST` se pueden utilizar para insertarlos en una base de datos. Para ello, utilice el lenguaje SQL y así podrá crear una consulta SQL como la siguiente:

```
$sql = "INSERT INTO client (apellido, nombre, comentario) VALUES
('GÓMEZ','ROBERTO','correcto');
```

Puede surgir un problema si el usuario introduce una cadena de caracteres que contiene un apóstrofo, por ejemplo:

La variable \$sql se convierte en:

```
$sql = "INSERT INTO client (apellido, nombre, poblacion) VALUES  
( 'GÓMEZ', 'ROBERTO', 'L'Hospitalet' )";
```

Se produce un error porque hay un apóstrofo después de la letra L.

Magic quotes es una funcionalidad PHP que se puede argumentar en el archivo php.ini, y permite proteger automáticamente los datos que han transmitido GET, POST o COOKIE.

Esta función evita los caracteres especiales, es decir, añade el carácter \ (barra invertida) delante de ' (apóstrofos), " (comillas), \ (barras invertidas) y NULL.

Sin embargo, esta funcionalidad está obsoleta desde la versión 5.3.0 de PHP y ya no existe desde la versión 5.4. Se recomienda que usted mismo asigne los caracteres especiales. Para desactivar magic quotes, introduzca el valor magic\_quotes\_gpc = Off en el archivo php.ini.

La función **addslashes()** desempeña la misma tarea, ya que evita los caracteres especiales.

Para evitar que un usuario guarde scripts maliciosos, utilice la función **htmlspecialchars()**, que convierte los caracteres especiales como < o > en &lt; o &gt;.

Esta función toma como argumento la cadena de caracteres que hay que codificar, y como argumentos opcionales la opción, el juego de caracteres y la doble codificación.

La opción puede contener tres valores:

- ENT\_NOQUOTES: ninguna conversión.
- ENT\_COMPAT: conversión de las comillas, pero no el apóstrofo.
- ENT\_QUOTES: conversión del apóstrofo, pero no las comillas.

El juego de caracteres es UTF-8, en lugar de ISO-8859-1 desde la versión PHP5.5 por defecto.

La doble codificación permite codificar una cadena que ya está codificada si el valor es true.

La función **htmlspecialchars\_decode()** permite la conversión inversa.

En el siguiente ejemplo, el código que recibe los datos está protegido correctamente:

```
<?php  
//Suponiendo que el formulario contenga un área password  
$password = htmlspecialchars(addslashes(trim($_POST['password'])));  
echo "Puede poner la variable $password en su consulta SQL.";  
$sql = "INSERT INTO client (apellido, password) VALUES  
( 'GÓMEZ', '".$password."' );  
  
?>
```

También puede utilizar la función **htmlentities()**, que equivale a la función **htmlspecialchars()**, salvo que transforme todos los caracteres especiales HTML.

## 7. Redirección de página

A veces es muy útil llamar una página PHP sin hacer clic en un botón. Para ello puede redireccionar una página PHP con la función **header()**.

Para redireccionar una página PHP o HTML, debe utilizar el atributo **location** de la función **header()**.

Por ejemplo:

```
<?php
header("Location:http://www.google.es"); // Redirecciona a
                                         // http://www.google.es
?>
```

O bien:

```
<?php
header("Location:redireccion.php"); // Redirecciona la página
                                         // redireccion.php
?>
```

O al pasar los argumentos:

```
<?php
header("Location:redireccion.php?apellido=Gómez&nombre=Juan");
// Redirecciona la página redireccion.php que pasa apellido y nombre
// como argumento
?>
```

Para llamar las páginas PHP en URL absoluta, debe utilizar las variables globales **\$\_SERVER[ 'HTTP\_HOST' ]** que dan el nombre del servidor y **\$\_SERVER[ 'PHP\_SELF' ]** que dan la ruta y el nombre de la página PHP actual:

```
<?php
$url_absoluta = "http://".$_SERVER['HTTP_HOST'].rtrim(dirname($_SERVER
['PHP_SELF']), '/\\').'/redireccion.php';
header("Location:".$url_absoluta); // Redirecciona la página
                                         // redireccion.php
?>
```

La función **header()** envía las consultas http a la página HTML y las acciones se explican en el siguiente sitio Web:  
<http://php.net/manual/es/function.header.php>



Su página PHP no debe contener código HTML y por tanto ningún `echo` antes de utilizar la función `header`, ya que provoca un error.

# Ejercicios

## 1. Enunciados

### **Ejercicio 1 (fácil)**

Cree una página login.php que contenga un formulario con una zona inicio de sesión y una zona contraseña. Añada el botón "comprobar", de tipo submit, que llama la página verif\_login.php. Esta página muestra "inicio de sesión correcto" si el apellido es igual a "Gómez" y la contraseña "alibaba". De lo contrario, esta página redirecciona a la página inicio de sesión.php con el mensaje "inicio de sesión incorrecto".

### **Ejercicio 2 (dificultad media)**

Cree una página tabla.php que contenga un formulario con una lista de selección simple que contenga los países y debajo una zona de tipo <div> que muestre las ciudades según el país que seleccione. El formulario contiene un botón de tipo "submit" que permite volver a llamar la misma página según el país que seleccione. Se trata de realizar esta página sin necesidad de utilizar JavaScript. Cuando muestra la página por primera vez, debe mostrar el primer país de la lista con sus correspondientes ciudades. Las dos tablas son:

```
$pais = array('Francia','Italia','Alemania','Rusia');
```

Y

```
$ciudades['Francia'][0] = "París";
$ciudades['Francia'][1] = "Lyon";
$ciudades['Francia'][2] = "Marsella";
$ciudades['Italia'][0] = "Roma";
$ciudades['Italia'][1] = "Milán";
$ciudades['Italia'][2] = "Nápoles";
$ciudades['Alemania'][0] = "Berlín";
$ciudades['Alemania'][1] = "Múnich";
$ciudades['Alemania'][2] = "Fráncfort";
$ciudades['Rusia'][0] = "Moscú";
$ciudades['Rusia'][1] = "San Petersburgo";
$ciudades['Rusia'][2] = "Nizhny-Novgorod";
```

### **Ejercicio 3 (dificultad media)**

Retome el ejercicio 1 y muestre en la página inicio de sesión.php el número de veces que el usuario ha intentado iniciar sesión antes de utilizar los inicios de sesión y contraseñas correctos. Muestre también todos los inicios de sesión y contraseñas que se han intentado introducir.

## 2. Soluciones

### **Solución del ejercicio 1**

login.php:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
    <title>Ejercicio inicio de sesión</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
</head>
<body>
<h2>Escriba su inicio de sesión y contraseña</h2>
<form action="verif_login.php" method="POST">
login:<input type="text" name="login" /><br />
contraseña:<input type="text" name="password" /><br />
<input type="submit" name="enviar" value="validar"/>
<br /><br />
<?php
if (isset($_GET['message']) && $_GET['message'] == '1') {
    echo "<span style='color:#ff0000'>inicio de sesión incorrecto
</span>";
}
?>
</form>
</body>
</html>
```

verfi\_login.php:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
    <title>Ejercicio inicio de sesión</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
</head>
<body>
<?php
if ($_POST['login'] == 'Gómez' && $_POST['password'] ==
'alibaba') {
    echo "<h2>inicio de sesión correcto</h2>";
}
else {
    header("location:login.php?message=1");
}
?>
</body>
</html>
```

## **Solución del ejercicio 2**

tabla.php:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio país</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <h2>Seleccione un país</h2>
        <form action="tabla.php" method="POST">
            <?php
                $pais = array('Francia','Italia','Alemania','Rusia');
                $ciudades['Francia'][0] = "París";
                $ciudades['Francia'][1] = "Lyon";
                $ciudades['Francia'][2] = "Marsella";
                $ciudades['Italia'][0] = "Roma";
                $ciudades['Italia'][1] = "Milán";
                $ciudades['Italia'][2] = "Nápoles";
                $ciudades['Alemania'][0] = "Berlín";
                $ciudades['Alemania'][1] = "Múnich";
                $ciudades['Alemania'][2] = "Fráncfort";
                $ciudades['Rusia'][0] = "Moscú";
                $ciudades['Rusia'][1] = "San Petersburgo";
                $ciudades['Rusia'][2] = "Nizhny-Novgorod";
            ?>
            pais:<select name="pais" >
            <?php
                if (isset($_POST['pais'])) {
                    $pais_seleccionado = $_POST['pais'];
                }
                else {
                    $pais_seleccionado = "Francia";
                }

                foreach ($pais as $valor) {
                    if ($valor == $pais_seleccionado) {
                        echo "<option value='".$valor."' selected='selected'
>".$valor."</option>";
                    }
                    else {
                        echo "<option value='".$valor."'>".$valor."</option>";
                    }
                }
            ?>
            </select>
            <br />
            <div>
            <?php
```

```

foreach ($ciudades as $clave => $valor) {
    if ($clave == $pais_seleccionado) {
        foreach ($valor as $valor_seleccionado) {
            echo $valor_seleccionado."<br />";
        }
    }
}
?>
</div>
<br />
<input type="submit" name="enviar" value="validar"/>
</form>
</body>
</html>

```

### Solución del ejercicio 3

login.php:

```

<?php
session_start();
if (isset($_SESSION['num_vez'])) {
    $_SESSION['num_vez'] = $_SESSION['num_vez']+1;
}
else {
    //inicio de sesión num_vez
    $_SESSION['num_vez'] = 0;
    //inicio de sesión login_password
    $_SESSION['login_password']="";
}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio login</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<h2>Escriba su inicio de sesión y su contraseña</h2>
<form action="verif_login.php" method="POST">
    inicio de sesión:<input type="text" name="login" /><br /><br />
    contraseña:<input type="text" name="password" /><br /><br />
    <input type="submit" name="enviar" value="validar"/>
<br /><br />
<?php
if (isset($_GET['message']) && $_GET['message'] == '1') {
    echo "<span style='color:#ff0000'>inicio de sesión incorrecto
</span>";
}

```

```

?>
<br />
Ha intentado <?php echo $_SESSION['num_vez'];?> veces.
<br />
<?php
if ($_SESSION['login_password']!="") {?>
Los inicios de sesión y contraseña intentados son:<?php echo
substr($_SESSION['login_password'],0,strlen($_SESSION
['login_password'])-2);
//¿quita el; el último espacio?>
<?php } ?>
</form>
</body>
</html>

```

verif\_login.php:

```

<?php
session_start();
$_SESSION['login_password'] = $_SESSION['login_password'].$_POST
['login']." y ".$_POST['password'].", ";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio inicio de sesión</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
<body>
<?php

if ($_POST['login'] == 'Gómez' && $_POST['password'] == 'alibaba') {
    echo "<h2>inicio de sesión correcto</h2>";
}
else {
    header("location:login.php?message=1");
}

?>
</body>
</html>

```

## La librería GD

Para crear una imagen en PHP, debe utilizar las funciones de la librería GD. Una librería es un archivo de extensión dll que contiene numerosas funciones PHP.

Para activar esta librería, en primer lugar hay que abrir el archivo PHP.ini (menú de configuración -> PHP) y quitar el punto y coma que está delante de la línea extension=php\_gd2.dll, en la parte inferior del archivo. A continuación, reinicie el servidor Web (menú reiniciar).

**Atención:** si usted alberga su sitio Web en un proveedor de servicios, debe asegurarse de que esta librería está activa, porque no siempre es así.

# Creación de una imagen

## 1. Header

Header indica al navegador que la página PHP reenvía una imagen, y no una página HTML. También indica el tipo de imagen que se ha creado: JPG o PNG.

Si tiene una imagen con mucho color, como por ejemplo una foto, es mejor utilizar el formato JPG o bien el formato PNG, que gestiona la transparencia.

El código PHP que debe insertar al inicio de la página es:

```
<?php  
header("Content-type: imagen/png");  
?>
```

## 2. Creación de una imagen vacía

Para crear una imagen vacía debe utilizar la función `imagecreate()`, que tiene dos parámetros: ancho y alto.

Por ejemplo:

```
<?php  
header("Content-type: imagen/png");  
$imagen = imagecreate(300,150);  
?>
```

Este código crea una imagen de 300 x 150 píxeles. La función `imagecreatecolor()` equivale a `imagecreate()`, pero sin el límite de 256 colores.

La variable `$imagen` es un objeto que permite manipular una imagen. Este concepto se ha explicado en el capítulo referente a los archivos con la función `fopen()`.

## 3. Creación y visualización de una imagen completa

Antes de mostrar la imagen, debe cambiar el color de fondo, pues de lo contrario será invisible. Para cambiar el color de fondo de la imagen, utilice la función `imagecolorallocate()`, que toma como parámetros el recurso y el código del color rojo, verde y azul.

Por ejemplo:

```
<?php  
header("Content-type: image/png");  
$imagen = imagecreate(300,150);  
$color_fondo = imagecolorallocate($imagen, 0, 255, 0);  
?>
```

Esta función colorea el fondo de la imagen y almacena este color en una variable (`$color_fondo`) para poder utilizarlo más adelante.

Para mostrar la imagen, utilice la función `imagepng()` tomando como parámetro el recurso. Para mostrar una imagen JPG, utilice la función `imagejpeg()`. Compruebe que Notepad++ lo codifica correctamente en ANSI; de lo contrario, no se podrá visualizar la imagen.

Por ejemplo:

```
<?php
header("Content-type: image/png");
$imagen = imagecreate(300,150);
$color_fondo = imagecolorallocate($imagen, 0, 255, 0);
imagepng($imagen);
?>
```

Inserte el código en un archivo `Imagen.php`.

Esta imagen, representada con un rectángulo verde en otra página PHP, se visualizará nombrando la imagen como una imagen normal. Introduzca su nombre en el atributo `src` de la etiqueta `img`:

```

```

Para guardar la imagen en el disco, agregue en el segundo parámetro opcional la ruta y el nombre de la imagen en la función `imagepng()`.

Por ejemplo:

```
<?php
$imagen = imagecreate(300,150);
$color_fondo = imagecolorallocate($imagen, 0, 255, 0);
imagepng($imagen, "fuente/Imagen1.png");
?>
```

Observe que se ha eliminado la función `header()` y ya no se muestra la imagen, aunque esté guardada.

Para crear una imagen desde una imagen que ya existe, utilice la función `imagecreatefromjpeg()`, y tome como parámetros el nombre de la imagen de tipo JPG.

Por ejemplo, una imagen `Koala.jpg` ubicada en el mismo lugar que su página PHP:

```
<?php
header("Content-type: image/jpeg");
$imagen = imagecreatefromjpeg("Koala.jpg");
imagejpeg($imagen);
?>
```

Todo esto tiene la ventaja de crear imágenes dinámicas, cuya forma y contenido pueden cambiar dependiendo de los datos incluidos en la base de datos.

Para terminar el script, agregue la función `imagedestroy($resorigen)`, que libera del servidor la memoria que ha ocupado la imagen.

```
<?php
header("Content-type: image/jpeg");
$imagen = imagecreatefromjpeg("Koala.jpg");
imagejpeg($imagen);
imagedestroy($imagen);
?>
```

# Texto y color

## 1. El color

Esta función es `imagecolorallocate()`, y se ha tratado anteriormente. Pone color de fondo a una imagen y almacena este color en una variable. Toma como parámetros el recurso y el código RGB.

La sintaxis es:

```
$color = imagecolorallocate ($recurso, $rojo, $verde, $azul);
```

Las variables `$rojo`, `$verde` y `$azul` van de 0 a 255. Puede encontrar el código del color que necesite en los programas Paint o Photoshop.

El siguiente ejemplo muestra un rectángulo azul:

```
<?php
header ("Content-type: image/png");
$imagen = imagecreate(300,150);
$color_fondo = imagecolorallocate($imagen, 0, 0, 255);
imagepng($imagen);
imagedestroy($imagen);
?>
```

## 2. El texto

La función que permite escribir el texto es `imagestring()`, que toma como parámetro el recurso, el tamaño de la fuente entre 0 y 5, las coordenadas x e y, la cadena de caracteres y el color de la cadena.

La sintaxis es:

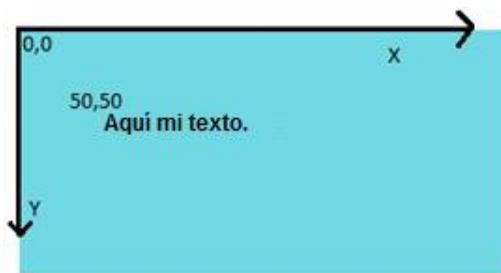
```
imagestring($recurso, $tamaño_tipo de letra, $x, $y, $cadena, $color_cadena);
```

El siguiente ejemplo muestra un texto en negro dentro de un rectángulo azul claro:

```
<?php
header ("Content-type: image/png");
$imagen = imagecreate(300,150);
$color_fondo = imagecolorallocate($imagen, 110, 210, 220);
//azul claro
$negro = imagecolorallocate($imagen, 0, 0, 0);
imagestring($imagen, 3, 50, 50, "Aquí mi texto.", $negro);
imagepng($imagen);
imagedestroy($imagen);
?>
```

La primera llamada a la función `imagecolorallocate()` cambia el color de fondo. Las siguientes llamadas solo almacenan un color en una variable.

Aquí se representa el resultado con las coordenadas x e y. Las coordenadas 0,0 corresponden al ángulo superior izquierdo, y las coordenadas 50,50 a la posición de inicio de la cadena de caracteres.



La función `imagestringup()` permite escribir el texto en forma vertical. Toma los mismos parámetros que la función `imagestring()`.

### 3. La transparencia

Debe utilizar el formato PNG para administrar la transparencia. La función `imagecolortransparent()` convierte un color en color transparente. Toma como parámetros el recurso y el color que va a convertir en transparente.

La sintaxis es:

```
imagecolortransparent($recurso, $color);
```

Este ejemplo muestra un texto negro en un rectángulo de color azul claro, pero el azul claro se convierte en transparente.

```
<?php
header("Content-type: image/png");
$imagen = imagecreate(300,150);
$color_fondo = imagecolorallocate($imagen, 110, 210, 220); //azul claro
$negro = imagecolorallocate($imagen, 0, 0, 0);
imagestring($imagen, 3, 50, 50, "Este es mi texto.", $negro);
imagecolortransparent($imagen, $color_fondo);
imagepng($imagen);
imagedestroy($imagen);
?>
```

Este ejemplo muestra el texto "Este es mi texto" en un fondo transparente.

### 4. Cambiar el tamaño de una imagen

La función `imagecopyresampled()` cambia el tamaño de la imagen y la ubica en otra imagen, en un punto concreto de coordenadas.

Esta función toma los siguientes parámetros:

- La imagen de destino: la imagen se crea con `imagecreate()`.

- La imagen de origen: imagen cuya miniatura se va a crear.
- La posición horizontal en la cual se ubicará la imagen en miniatura.
- La posición vertical en la cual se ubicará la imagen en miniatura.
- El ancho de la imagen en miniatura.
- El alto de la imagen en miniatura.
- El ancho de la imagen de origen: puede tomar solo una parte de la fuente.
- El alto de la imagen de origen.

En el siguiente ejemplo, el código PHP permite crear una imagen en miniatura de la imagen Koala.jpg y la llama mini\_Koala.jpg.

```
<?php

$imagen_origen = imagecreatefromjpeg("Koala.jpg"); // El origen es la
// imagen Koala.jpg
$destino = imagecreatetruecolor(102, 77); // Creación de la miniatura
// vacía

$ancho_origen = imagesx($imagen_origen); // cambia el ancho de la imagen
$alto_origen = imagesy($imagen_origen); // cambia el alto de la imagen
$ancho_destino = 102;
$alto_destino = 77;

// Creación de la miniatura
imagecopyresampled($destino, $imagen_origen, 0, 0, 0, 0,
$ancho_destino, $alto_destino, $ancho_origen,
$alto_origen);

// Guardar la miniatura con el nombre "mini_Koala.jpg"
imagejpeg($destino, 'mini_Koala.jpg');
echo "Mostrar la miniatura: <img src='mini_Koala.jpg' name=
'miniatura' />";
imagedestroy($imagen_origen);
?>
```

## 5. Superponer las imágenes

Para mostrar una imagen sobre otra, utilice la función `imagecopy()`, que toma los siguientes parámetros:

- La imagen de destino.
- La imagen de origen: imagen que hay que superponer.
- La posición horizontal de la imagen de destino.
- La posición vertical de la imagen de destino.
- La posición horizontal de la imagen de origen.
- La posición vertical de la imagen de origen.
- El ancho de la imagen de origen.

- El alto de la imagen de origen.

En el siguiente ejemplo, el código PHP coloca una imagen pequeña ordenador.png sobre la imagen Koala.jpg:

```
<?php
header ("Content-type: image/jpeg");

// Creación de dos imágenes como objeto
$origen = imagecreatefrompng("ordenador.png"); // El ordenador es
                                                // el origen
$destino = imagecreatefromjpeg("Koala.jpg"); // El Koala es
                                                // el destino

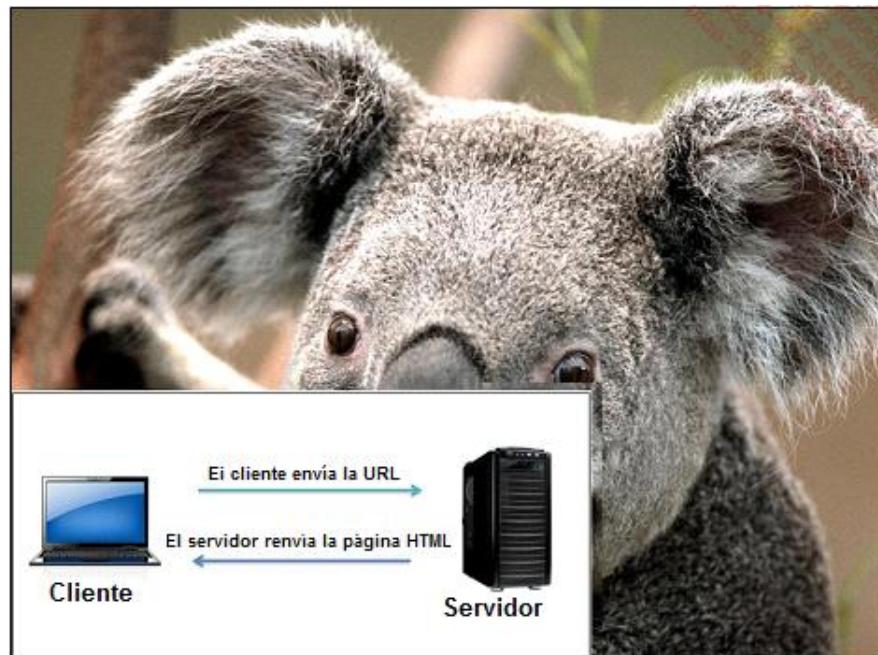
$ancho_origen = imagesx($origen); //ancho de la imagen origen
$alto_origen = imagesy($origen); //alto de la imagen origen
$ancho_destino = imagesx($destino); //ancho de la imagen destino
$alto_destino = imagesy($destino); //alto de la imagen,destino

// Colocar el logo abajo a la izquierda
$x = 0;
$y = $alto_destino - $alto_origen;

// Colocar la imagen origen en la imagen destino
imagecopy($destino, $origen, $x, $y, 0, 0, $ancho_origen,
$alto_origen);

// Mostrar la imagen final
imagejpeg($destino);
imagedestroy($destino);
?>
```

Da como resultado:



La función `imagecopymerge()` equivale a la función `imagecopy()`, pero funde la imagen original a través de un parámetro complementario de 0 a 100 (0 es completamente transparente).

El mismo ejemplo con un fundido se convierte en:

```
<?php
header ("Content-type: image/jpeg");

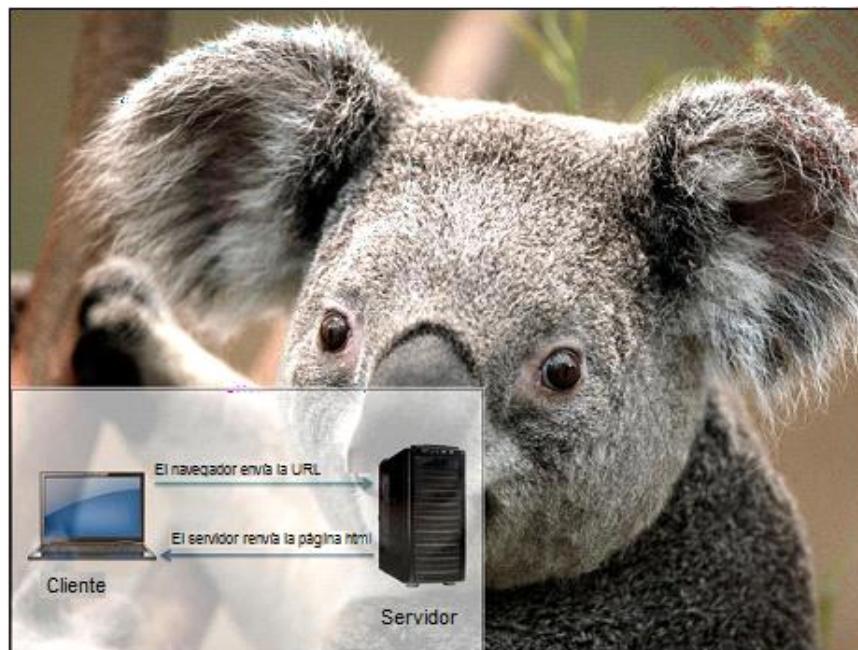
// Creación de dos imágenes como objeto
$origen = imagecreatefrompng("ordenador.png"); // El ordenador es
                                                // el origen
$destino = imagecreatefromjpeg("Koala.jpg"); // El Koala es
                                                // el destino

$ancho_origen = imagesx($origen); //ancho de la imagen origen
$alto_origen = imagesy($origen); //alto de la imagen origen
$ancho_destino = imagesx($destino); //ancho de la imagen destino
$alto_destino = imagesy($destino); //alto de la imagen destino

// Colocar el logo abajo a la izquierda
$x = 0;
$y = $alto_destino - $alto_origen;

// Colocar la imagen origen en la imagen destino con un fundido
imagecopymerge($destino, $origen, $x, $y, 0, 0, $ancho_origen, $alto_origen, 65);
// Mostrar la imagen final
imagejpeg($destino);
imagedestroy($destino);
?>
```

Da como resultado:



## Las formas

Las funciones de la librería GD dibujan formas como círculos, rectángulos, polígonos, etc.

A continuación mostramos algunas funciones con \$recurso de una imagen que se ha creado anteriormente:

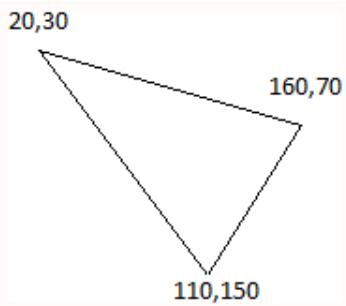
- `imagesetpixel($recurso, $x, $y, $color)`: píxel del color \$color en las coordenadas \$x,\$y.
- `imageline($recurso, $x1, $y1, $x2, $y2, $color)`: línea de color \$color entre las coordenadas \$x1,\$y1 y las coordenadas \$x2,\$y2.
- `imagefill($recurso, $x, $y, $color)`: rectángulo desde las coordenadas \$x,\$y hasta el ángulo inferior derecho del recurso rellenado con el color \$color.
- `imagerectangle($recurso, $x1, $y1, $x2, $y2, $color)`: rectángulo de contorno que tiene el color \$color entre las coordenadas \$x1,\$y1 y las coordenadas \$x2,\$y2.
- `imagefilledrectangle($recurso, $x1, $y1, $x2, $y2, $color)`: rectángulo con el fondo que tiene el color \$color entre las coordenadas \$x1,\$y1 y las coordenadas \$x2,\$y2.
- `imageellipse($recurso, $x, $y, $ancho, $alto, $color)`: elipse con coordenadas del centro \$x y \$y, de ancho \$ancho, de alto \$alto y el contorno con el color \$color.
- `imagefilledellipse($recurso, $x, $y, $ancho, $alto, $color)`: elipse con coordenadas del centro \$x y \$y, de ancho \$ancho, de alto \$alto y cuyo fondo tiene el color \$color.
- `imagepolygon($recurso, $registro_puntos, $numero_puntos, $color)`: polígono con un número de puntos igual a \$numero\_puntos, donde las coordenadas de estos puntos se ubican en el registro \$registro\_puntos y cuyo contorno tiene el color \$color.
- `imageflip($recurso, $modo)`: devuelve la imagen con el modo suministrado. Los valores del modo son:  
IMG\_FLIP\_HORIZONTAL: devuelve la imagen horizontalmente.  
IMG\_FLIP\_VERTICAL: devuelve la imagen verticalmente.  
IMG\_FLIP\_BOTH: devuelve la imagen horizontal y verticalmente.  
Esta función sólo existe desde PHP 5.5.

Por ejemplo:

```
<?php

header("Content-type: image/png");
$recurso = imagecreate(300,150);
$blanco = imagecolorallocate($recurso, 255, 255, 255);
$negro = imagecolorallocate($recurso, 0, 0, 0);
//registro de 3 puntos con sus coordenadas x e y
$registro_puntos = array(20, 30, 160, 70, 110, 150);
//creación del polígono de 3 puntos en negro.
imagepolygon($recurso, $registro_puntos, 3, $negro);
//mostrar la imagen
imagepng($recurso);
imagedestroy($recurso);
?>
```

Y muestra, sin las coordenadas, lo siguiente:



- `imagefilledpolygon($recurso, $registro_puntos, $numero_puntos, $color)`: polígono con un número de puntos igual a `$numero_puntos`, donde las coordenadas de estos puntos se ubican en el registro `$registro_puntos` y cuyo fondo tiene el color `$color`.
- `imagesetthickness($recurso, $grosor)`: aumenta el grosor de la línea un número de píxeles igual a `$grosor`.

Existen numerosas funciones en esta librería, que se explica en este enlace:  
<http://www.php.net/manual/es/ref.image.php>

# Ejemplos

## 1. Ejemplo 1

Un ejemplo común consiste en mostrar un gráfico que representa el número de visitas diarias a un sitio Web. Estos datos pueden proceder de una base de datos MySQL, pero se indican de forma fija en el ejemplo.

```
<?php
header("Content-type: image/png");
$registro_visitas = array(520, 458, 642, 741, 254, 657, 356, 912, 259, 712);

$anchoImagen = 450;
$altoImagen = 400;
$imagen = imagecreate($anchoImagen, $altoImagen);
$blanco = imagecolorallocate($imagen, 255, 255, 255);
$negro = imagecolorallocate($imagen, 0, 0, 0);
$rojo = imagecolorallocate($imagen, 255, 0, 0);

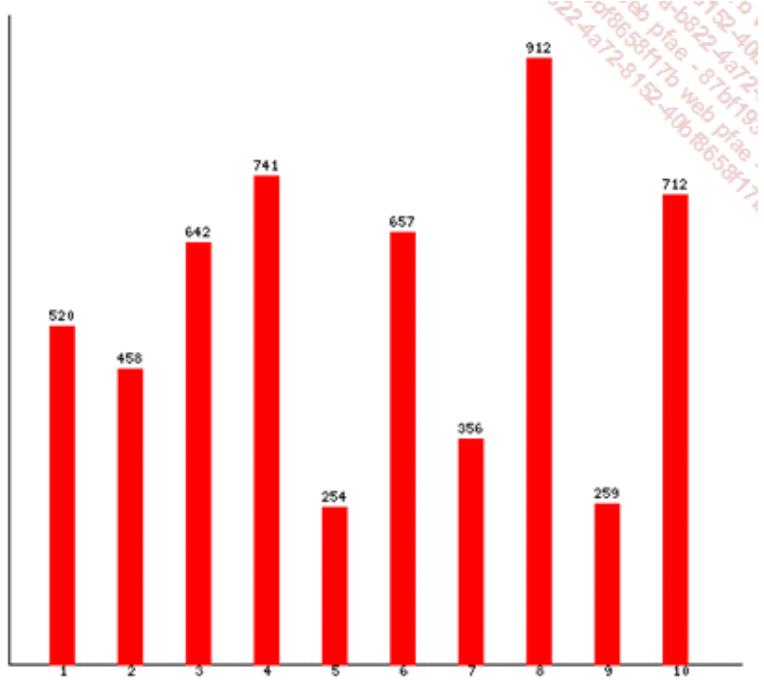
// trazo horizontal para representar el eje de los días
imageline($imagen, 10, $altoImagen-10, $anchoImagen-10,
$altoImagen-10, $negro);
// Mostrar el número de días
for ($dia=1; $dia<=10; $dia++) {
    imagestring($imagen, 0, $dia*40, $altoImagen-10, $dia, $negro);
}

// trazo vertical que representa el número de visitas
imageline($imagen, 10, 10, 10, $altoImagen-10, $negro);

// el número máximo de visitas proporcional a lo alto de la imagen
$visitas_maximas = 1000;
// trazado de rectángulos
for ($dia=1; $dia<=10; $dia++) {
    $altoRectangulo = round((($registro_visitas[$dia-1]*$altoImagen)/$visitas_maximas));
    imagefilledrectangle($imagen, $dia*40-6, $altoImagen-$altoRectangulo,
$altoRectangulo, $dia*40+8, $altoImagen-10, $rojo);
    imagestring($imagen, 0, $dia*40-6, $altoImagen-$altoRectangulo-10,
$registro_visitas[$dia-1], $negro);
}

imagePng($imagen);
imagedestroy($imagen);
?>
```

Da como resultado:



## 2. Ejemplo 2

En este ejemplo se utiliza la función `imagerotate()` para rotar la imagen y cambiar el color de fondo:

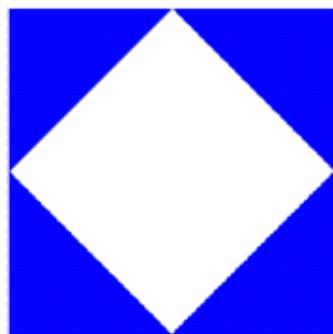
```
<?php
header('Content-type: image/gif');

$imagen = imagecreate(130, 130);
$blanco = imagecolorallocate($imagen, 255, 255, 255);
$azul = imagecolorallocate($imagen, 0, 0, 255);

//rotación de la imagen de 45° y fondo en azul
$imagen = imagerotate($imagen, 45, $azul);

imagegif($imagen);
imagedestroy($imagen);
?>
```

Da como resultado:





# Presentación

## 1. Introducción

Una base de datos es un conjunto estructurado de datos que administra un equipo. Si desea guardar la información en su sitio Web, como el nombre o los apellidos o lo necesario para crear un blog, está obligado a utilizar una base de datos. En teoría, puede utilizar un archivo de texto, pero en la práctica esto no es recomendable, porque puede provocar muchos problemas, por ejemplo, si varias personas están tratando de escribir al mismo tiempo.

Hay varios programas de bases de datos, como Oracle o SQL Server, pero no son gratuitos y solo resultan realmente útiles en la gestión de un gran volumen de datos. En este capítulo vamos a ver la base de datos MySQL, que es gratuita y está incluida en EasyPHP. Puede instalar MySQL en un servidor distinto al servidor Web, pero en la práctica se instala en el mismo lugar.

Una base de datos permite almacenar datos y también clasificarlos, de esta manera se pueden encontrar rápidamente con el lenguaje SQL (*Structured Query Language*). El lenguaje SQL se utiliza para ejecutar acciones en la base de datos, como crear tablas, añadir o eliminar datos...

Una consulta es un registro de escritura del lenguaje SQL.

Por ejemplo, para leer los apellidos y el nombre de la tabla cliente:

```
SELECT apellido, nombre FROM cliente;
```

Tiene la ventaja de ser común a todas las bases de datos. Por supuesto, hay algunas diferencias entre las bases de datos, pero la mayoría de las consultas que se ejecutan son válidas en todas partes.

Estas consultas se van a ejecutar desde el código PHP y pueden devolver datos en una tabla para mostrarlos posteriormente.

## 2. Estructura

Una base de datos está formada por tablas. Una tabla es un conjunto de campos (columnas).

Los datos se almacenan en cada registro de la tabla.

Ejemplo de tabla Persona:

<b>id_person</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>
1	Carolina	Gómez del Pozo	37
2	Estefanía	Morales HonHon	35
3	Luna	Morales	18
4	Luis	Del Morán	42

Esta tabla contiene cuatro campos:

- ID\_PERSON: identificador único de la persona

- APELLIDOS: apellidos de la persona
- NOMBRE: nombre de la persona
- EDAD: edad de la persona

Por lo tanto, esta tabla contiene cuatro líneas o registros, y no existe un límite para el número de registros, excepto el impuesto por el tamaño de su disco duro.

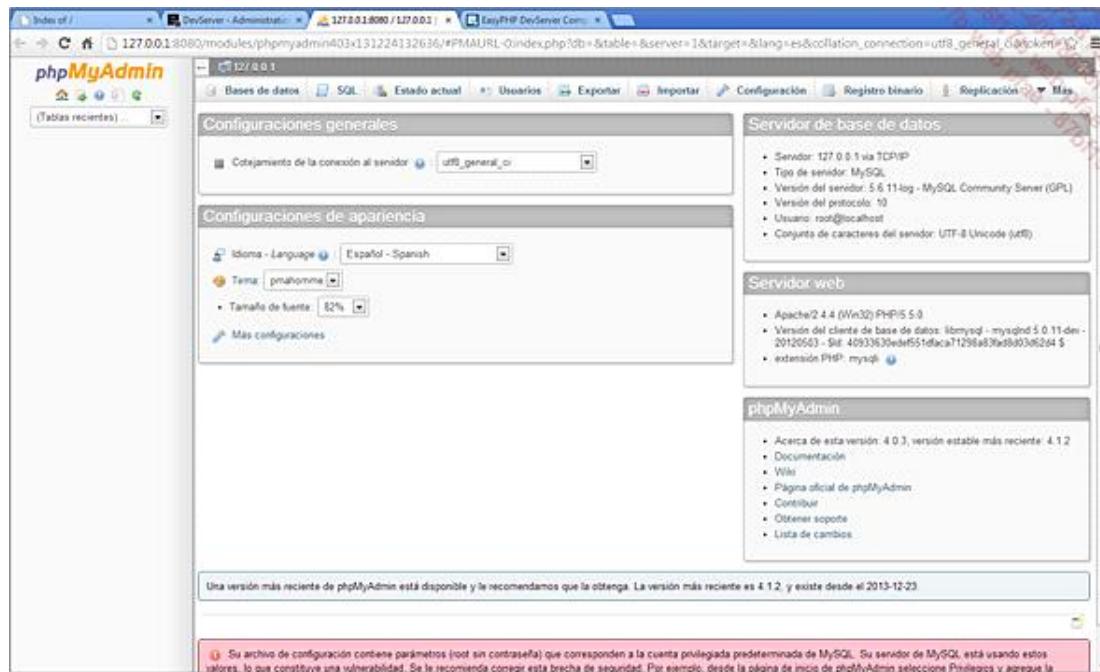
# PHPMyAdmin

PHPMyAdmin se incluye en EasyPHP para gestionar su interfaz Web de base de datos. Esta interfaz cuenta con diferentes menús para crear bases de datos, tablas, campos, añadir o eliminar datos...

Así, puede administrar su base de datos sin tener que escribir el comando SQL.

Para acceder a PHPMyAdmin, haga clic con el botón derecho del ratón sobre la "e" de EasyPHP en la barra de tareas y en **Administración**. Haga clic en **Abrir** del módulo **Administración MySQL: PhpMyAdmin 4.0.3**.

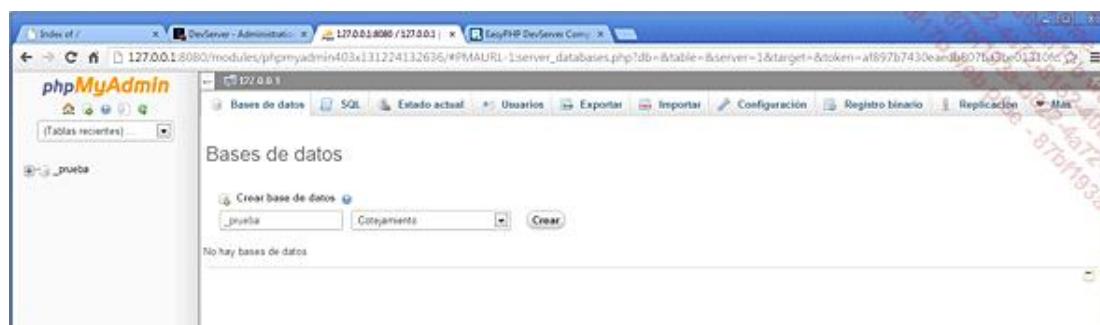
Accederá a una interfaz Web con la siguiente URL:  
<http://127.0.0.1/módulos/phpmyadmin403x130626101038/#PMAURL-0:índice. deb=&tabla=&server=1&target=&token=5b0148d8936b4b993c28008d6d4bd01b>



Ahora va a aprender a crear una base de datos y a utilizar todas las herramientas de PHPMyAdmin.

- Haga clic en la ficha **Bases de datos**.
- Introduzca el nombre de la base de datos que desee crear en la zona introducida y mantenga por defecto el idioma utilizado (cotejamiento). Haga clic en **Crear**.

A la izquierda aparecerá su base de datos. En el siguiente ejemplo, la base de datos se llama prueba.



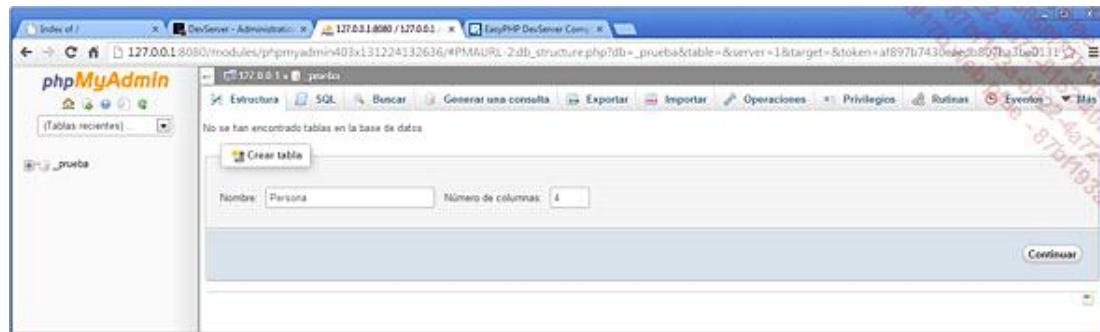
 Nunca introduzca caracteres especiales en los nombres de la base, de la tabla, de los campos o cualquier otro objeto en la base de datos. Así podrá evitar complicaciones.

→ Haga clic en \_prueba.

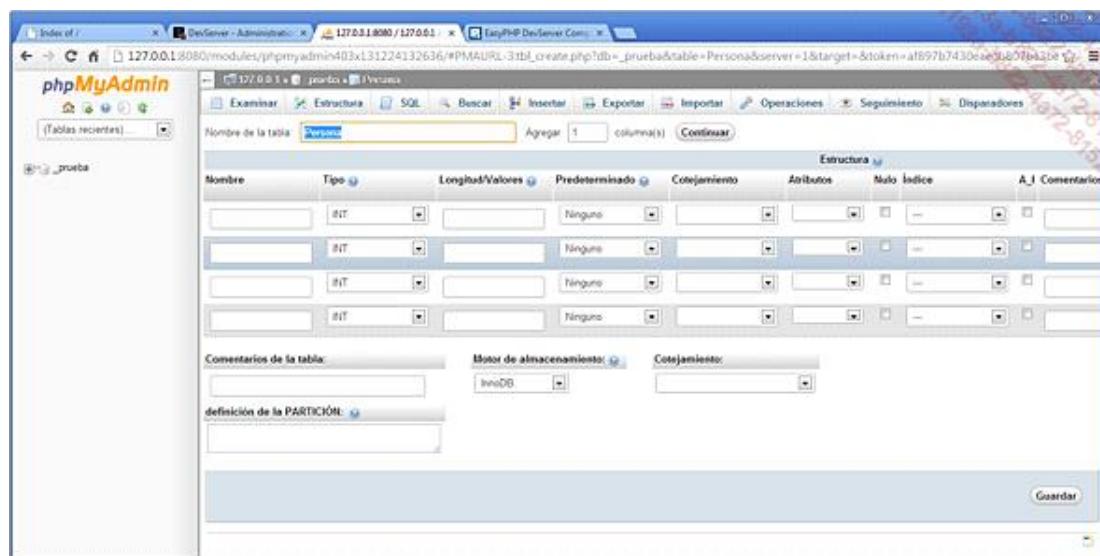
Ahora tenemos que crear una tabla que muestre el número de campos que contiene. Es un valor indicativo, porque todo puede cambiar después.

→ Escriba "Persona" en el cuadro de texto a la derecha del nombre y el número 4 en el cuadro de texto del número de campos. De hecho, la tabla Persona contendrá cuatro campos:

Id\_persona (Int), apellidos (Varchar 20), Nombre (Varchar 20) y Edad (Int).



→ Haga clic en **Continuar**.



Aparecen cuatro registros, cada uno de ellos en un campo distinto.

La columna **Nombre** contiene el nombre del campo.

La columna **Tipo** contiene el tipo de campo. Los tres tipos más utilizados son INT (entero de -2147483648 a 2147483647), VARCHAR (cadena de caracteres de longitud máxima de 255 caracteres) y Date.

Los otros tipos son:

- TINYINT: número entero codificado en un byte. Valores de -128 a 127 (0 a 255 en unsigned) (char).
- SMALLINT: número entero codificado con dos bytes. Valores de -32768 a 32767 (0 a 65535 en unsigned) (small int).
- MEDIUMINT: número entero codificado con tres bytes. Valores de -8388608 a 8388607 (0 a 16777215 en unsigned).

- BIGINT: número entero codificado con ocho bytes. Valores de -9223372036854775808 a 9223372036854775807 (0 a 18446744073709551615 en unsigned).
- DECIMAL: número decimal en forma de cadena de caracteres (el tamaño puede variar: 1 cifra => 1 carácter).
- FLOAT: número decimal de simple precisión, codificado con cuatro bytes. Valores de -3.402823466x10<sup>38</sup> a -1.175494351x10<sup>-38</sup> y de 1.175494351x10<sup>-38</sup> a 3.402823466x10<sup>38</sup>.
- DOUBLE: número decimal de doble precisión, codificado con ocho bytes. Valores de -1.7976931348623157x10<sup>308</sup> a -2.2250738585072014x10<sup>-308</sup> y de 2.2250738585072014x10<sup>-308</sup> a 1.7976931348623157x10<sup>308</sup>.
- BOOL: booleano con un valor verdadero o falso (se considera el valor 0 como falso).
- BIT(M): campo de tipo bit en el que M indica el número de bytes por cada valor (entre 1 y 64).
- SERIAL es un alias de BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE.
- DATETIME: fecha en formato aaa-mm-dd hh:mm:ss comprendida entre 01/01/1000 y 31/12/9999.
- TIMESTAMP: sin información complementaria, equivale a DATETIME pero sin los separadores. Puede crear un TIMESTAMP incompleto (por ejemplo AAMMDDHH).

 En PHP, un TIMESTAMP es un entero que representa el número de segundos que han transcurrido desde el 1 de Enero de 1970, mientras que en MySQL, como acaba de ver, es una fecha formateada.

- TIME: hora en formato hh:mm:ss.
- YEAR: año en formato aa o aaaa.
- CHAR: cadena de caracteres de longitud fija. Es una longitud comprendida entre 1 y 255.
- TINYTEXT: texto de longitud variable que puede tener hasta 255 caracteres.
- TEXT: texto de longitud variable que puede tener hasta 65535 caracteres.
- MEDIUMTEXT: texto de longitud variable que puede tener hasta 16777215 caracteres.
- LONGTEXT: texto de longitud variable que puede tener hasta 4294967295 caracteres.
- BLOB: dato binario que puede almacenar archivos o texto. En este caso, los tipos TINYBLOB, BLOB, MEDIUMBLOB y LONGBLOB son idénticos a sus homólogos TEXT, con la diferencia de que las búsquedas en un tipo BLOB tienen en cuenta las mayúsculas y minúsculas.

 Existen otros tipos de datos como ENUM o SPATIAL, aunque apenas se utilizan. Encontrará más información en <http://dev.mysql.com>

La columna **Longitud/Valores** permite concretar:

- En el tipo CHAR o VARCHAR, el número máximo de caracteres.
- En el tipo INT, el número de cifras significativas.
- En el tipo TIMESTAMP, la longitud de la fecha.

La columna **Predeterminado** permite definir el valor por defecto que hay que insertar si no se ha informado en la consulta.

La columna **Cotejamiento** permite definir el juego de caracteres del campo.

Atributos	Nulo	Índice	A_I	Comentarios	MIME-type	Transformación del navegador	Opciones de transformación

La columna **Atributos** permite definir si el número está firmado o no (UNSIGNED) o si la cadena de caracteres es sensible a mayúsculas y minúsculas (BINARY).

La columna **Nulo** permite definir la autorización del valor Null. Si selecciona esta casilla, el valor Null está autorizado. La base de datos guarda Null en un campo cuando no se ha informado el valor.

La columna **Índice** puede contener cuatro valores:

- **UNIQUE**: el valor del campo debe ser único.
- **INDEX**: la base de datos va a optimizar los datos de este campo. Normalmente los identificadores están indexados, ya que las consultas tienen filtros o uniones en estos identificadores.
- **PRIMARY**: el valor del campo es único y tiene un índice. Solo puede haber una «clave primaria» por cada tabla.
- **FULLTEXT**: índice de los campos de tipo texto.

La columna **A\_I (Auto\_incremento)** permite especificar que este campo se incrementa automáticamente en la base de datos.

La columna **Comentarios** permite añadir en el campo un comentario. Solo es visible en PHPMyAdmin.

La columna **MIME-type** permite definir el tipo si quiere guardar un archivo en la base de datos.

La columna **Transformación del navegador** permite elegir entre una gran selección de transformaciones que ya están predefinidas. Hay transformaciones globales y transformaciones de tipo MIME ligadas. Las transformaciones globales se pueden utilizar para cualquier tipo MIME. Se tendrá en cuenta, en su caso, el tipo MIME. Las transformaciones de tipo ligadas funcionan generalmente con algunos tipos MIME. Puede utilizar transformaciones de tipos MIME en las que la función no se ha definido. No hay control a la hora de asegurarse de que ha seleccionado la transformación correcta, así que tenga cuidado con el formato de su resultado.

La columna **Opciones de transformación** es un campo de texto libre. Introduzca aquí las opciones concretas de la función de transformación.

En el ejemplo de la tabla Persona, el campo id\_persona es de tipo INT, PRIMARY y Auto-incremental, el campo Apellidos es de tipo VARCHAR(20), el campo Nombre es de tipo VARCHAR(20) y finalmente el campo Edad es de tipo INT.

The screenshot shows the 'Estructura' (Structure) tab of the phpMyAdmin interface. A new table named 'Personas' is being created. It contains four columns: 'id\_persona' (Type: INT, Null: No, Default: Ninguno), 'Nombre' (Type: VARCHAR, Null: Yes, Default: Ninguno), 'Apellido' (Type: VARCHAR, Null: Yes, Default: Ninguno), and 'Edad' (Type: INT, Null: Yes, Default: Ninguno). The 'Apellido' column is currently selected. The 'Cotejamiento' (Collation) dropdown is set to 'InnoDB'. At the bottom right, there is a 'Guardar' (Save) button.

El motor de almacenamiento y la definición de partición permiten establecer la manera en que la base de datos va a almacenar los datos en sus archivos. Estos son argumentos muy sensibles que afectan solo al administrador de la base de datos.

- Haga clic en le botón **Guardar**.

The screenshot shows the 'Listado' (List) tab of the phpMyAdmin interface. The 'personas' table is listed with one row. The columns are 'Apellido' (latin1\_swedish\_ci), 'Nombre' (latin1\_swedish\_ci), 'Edad' (int(11)), and 'id\_persona' (int(11)). The 'Apellido' column is highlighted with a yellow border. Below the table, there is a 'Crear tabla' (Create table) form with fields for 'Nombre' and 'Número de columnas'.

Observe que PHPMyAdmin muestra el código SQL que equivale a crear la tabla con sus correspondientes campos.

- Pulse en la pestaña **Estructura** y despliegue la arborescencia de la tabla persona, en la parte de la izquierda.

**ALTER TABLE "persona" CHANGE "id\_persona" "id\_persona" INT( 11 ) NOT NULL AUTO\_INCREMENT ;**

#	Nombre	Tipo	Colejamiento	Atributos	Null	Predeterminado	Extra	Acción
1	<b>id_persona</b>	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar  Eliminar  Primaria  Único  Índice  Espacial <input type="checkbox"/> Texto completa <input type="checkbox"/> Valores distintos
2	Nombre	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice  Espacial <input type="checkbox"/> Texto completa <input type="checkbox"/> Valores distintos
3	Apellidos	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice <input type="checkbox"/> Espacial <input type="checkbox"/> Texto completa <input type="checkbox"/> Valores distintos
4	Edad	int(11)			No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice <input type="checkbox"/> Espacial <input type="checkbox"/> Texto completa <input type="checkbox"/> Valores distintos

Vista de impresión Vista de relaciones Planteamiento de la estructura de tabla Hacer seguimiento a la tabla Mover columnas

**Agregar:**   Al final de la tabla  Al comienzo de la tabla  Después de **id\_persona**

**Información:**

Espacio utilizado	Estadísticas de la fila
Datos 14 KB	Formato Compact
Índice 0 B	Contenido latin1_swedish_ci
Total 14 KB	Índice automático siguiente 1
	Creación 24-12-2013 a las 19:11:27

Deabajo puede modificar () o eliminar () los elementos de la estructura de la tabla.

A la izquierda puede seleccionar su base de datos (\_prueba), las tablas de su base de datos, los campos y los índices. Por ahora solo hay una, que es la tabla Persona.

Arriba hay diferentes pestañas que le permiten realizar acciones en su tabla.

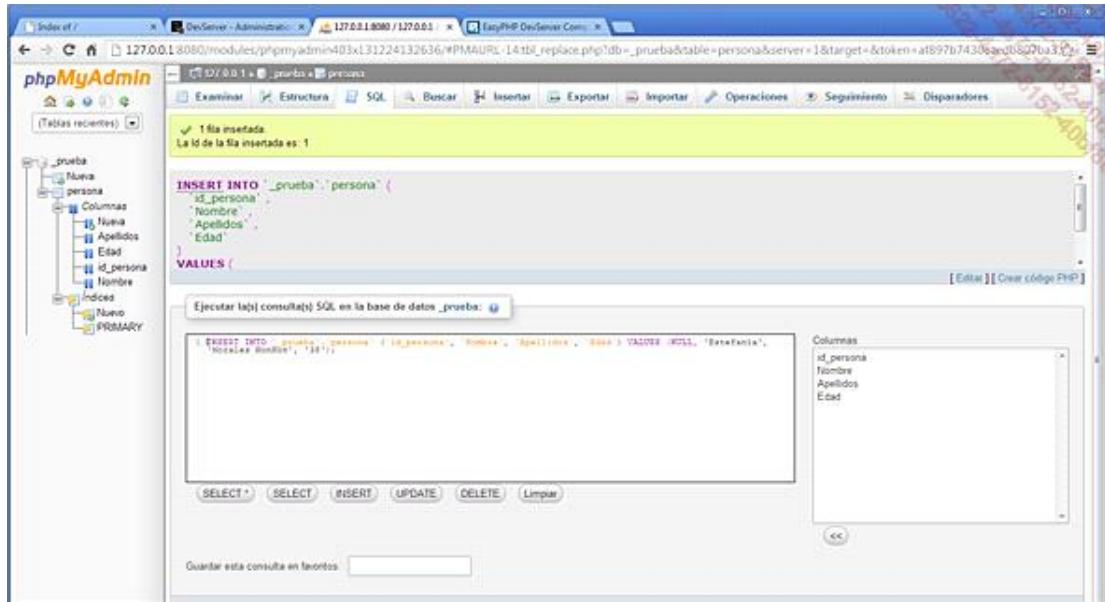
**Pestaña Insertar:**

The screenshot shows the phpMyAdmin interface for the 'persona' table in the '\_prueba' database. The left sidebar displays the database structure with a tree view of tables ('\_prueba', 'Nueva persona'), columns ('Columnas', 'id\_persona', 'Nombre', 'Apellidos', 'Edad'), and indices ('Índices', 'Nuevo', 'PRIMARY'). The main area is titled 'Insertar' and contains two sets of input fields for inserting data into the 'persona' table. The first set of fields includes 'id\_persona' (int(11)), 'Nombre' (varchar(20) with value 'Estefanía'), 'Apellidos' (varchar(20) with value 'Morales HonHon'), and 'Edad' (int(11) with value '38'). The second set of fields is identical and has a checked 'Ignorar' (Ignore) checkbox. Both sets have a 'Continuar' (Continue) button. Below these is a section for inserting multiple rows, with dropdowns for 'Insertar como una nueva fila' (Insert as a new row), 'y luego' (and then), and 'Volver' (Go back). A 'Continuar' button is also present here. At the bottom, there is a field 'Continuar inserción con' (Continue insertion with) containing '2' and a dropdown menu.

Esta pestaña se utiliza para introducir valores en la tabla. Puede introducir valores de dos en dos.

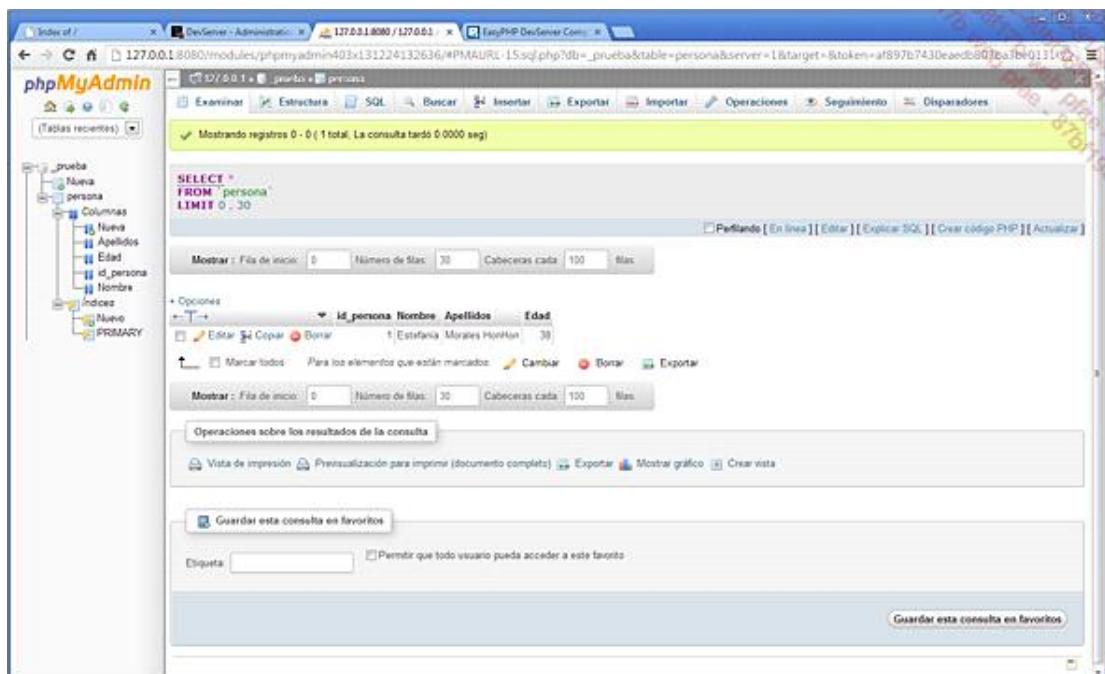
En el ejemplo anterior, el valor de `id_persona` no se rellena porque es autoincremental; es por tanto, la base de datos la que asignará un número.

A continuación, introduzca el apellido, el nombre y la edad y haga clic en **Continuar**.



PHPMyAdmin muestra el código SQL que equivale a la inserción de esta nueva persona.

Ahora haga clic en la pestaña **Examinar** para que aparezca el registro que acaba de insertar.



Puede modificar ( ) o eliminar ( ) datos de la tabla.

Puede visualizar una versión imprimible, haciendo clic en **Previsualización para imprimir**.

Index of / DevServer - Administration 127.0.0.1:8080/modules/phpmyadmin403x13

# Resultado SQL

**Servidor:** 127.0.0.1  
**Base de datos:** \_prueba  
**Tiempo de generación:** 24-12-2013 a las 19:20:44  
**Generado por:** phpMyAdmin 4.0.3 / MySQL 5.6.11-log  
**consulta SQL:** SELECT \* FROM `persona` LIMIT 0, 30 ;  
**Filas:** 1

id_persona	Nombre	Apellidos	Edad
1	Estefania	Morales HonHon	38

**Imprimir**

Puede exportar este contenido en formato SQL con el botón **Exportar**.

También puede crear una vista (sección SQL avanzado - Otros objetos de MySQL en este capítulo) y un gráfico que representa los datos de la tabla, lo que no es demasiado útil con un solo dato.

Para terminar, puede guardar esta consulta con un marcador, lo que le permitirá reutilizarlo desde la pestaña **SQL**.

La pestaña **SQL** permite ejecutar consultas SQL.

Index of / DevServer - Administration 127.0.0.1:8080/modules/phpmyadmin403x13224132636/#PMAURL:16tbl\_sql.php?db=\_prueba&table=persona&server=1&target=&token=a897b7430eardb807ba3be0c12

Ejecutar la(s) consulta(s) SQL en la base de datos \_prueba:

```
SELECT * FROM `persona`;
```

Columnas:  
id\_persona  
Nombre  
Apellidos  
Edad

SELECCIONAR | INSERTAR | ACTUALIZAR | BORRAR | LIMPIAR

Guardar esta consulta en favoritos:

Delimitador:  Mostrar esta consulta otra vez  Mantener la caja de texto con la consulta  Continuar

La pestaña **Buscar** permite buscar datos en la tabla desde varios filtros.

La pestaña **Exportar** permite realizar copias de seguridad de su base de datos con formatos diferentes, como un archivo de Excel, un archivo de texto o un archivo SQL. Se recomienda elegir el formato SQL, ya que permite guardar los datos y la estructura de la base de datos. También será más fácil para importarlo a otra base de datos más adelante.

La pestaña **Importar** permite importar datos o la estructura de una base de datos.

La pestaña **Seguimiento** permite crear una versión nueva y volver a este estado si hay una mala gestión.

La pestaña **Operaciones** permite cambiar el nombre de la tabla, mover la tabla a otra base de datos y copiar los datos o la estructura en otra base de datos.

Para terminar, volveremos a la pestaña **Disparadores** a lo largo de este capítulo.

# El lenguaje SQL

## 1. Presentación

En este apartado, no se muestra la forma de instalar el servidor MySQL. Va a utilizar PHPMyAdmin y especialmente la pestaña SQL para ejecutar las diferentes consultas.

Se recomienda no poner caracteres especiales en los nombres de bases de datos, tablas o campos. Debe tratar de usar solo caracteres no acentuados, cifras y el símbolo \_. Si añade espacios en sus nombres, tiene que delimitarlos con comillas o apóstrofes.

Hemos visto en la presentación de PHPMyAdmin que una base de datos contiene tablas que a su vez contienen campos. Por lo tanto, la sintaxis para hacer referencia a un campo en la consulta es:

```
nombre_base_de_datos.nombre_tabla.nombre_campo
```

Cuando ejecuta una consulta SQL con la pestaña SQL de PHPMyAdmin, debe saber en qué lugar se ubica en relación con la base de datos. Si ya ha seleccionado la base de datos, se mostrará el nombre del servidor y el nombre de su base de datos en la parte superior de la página PHPMyAdmin:



En este caso, no es necesario recordar el nombre de su base de datos en la consulta, puesto que ya está dentro de ella.

Por ejemplo, escriba:

```
SELECT * FROM Persona
```

Y no:

```
SELECT * FROM _prueba.Persona
```

En el resto de este capítulo, se utiliza la base de datos \_prueba, que contiene la tabla Persona con los campos Id\_person, Nombre, Apellidos y Edad.

## 2. Leer datos

La instrucción que permite leer en una tabla es SELECT.

Por ejemplo:

```
SELECT * FROM Persona
```

- **SELECT:** selecciona o lee
- **\*:** todos los campos
- **FROM:** de o en
- **Persona:** nombre de la tabla

Por lo tanto, esta consulta significa: leer todos los campos de la tabla Persona.

Seleccione la tabla Persona y copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

```

    SELECT *
    FROM `persona`
    WHERE 1
    LIMIT 0 , 30
  
```

	Id_persona	Nombre	Apellidos	Edad
	1	Estefania	Morales HonHon	38

Para mostrar solo algunos campos, únicamente tiene que nombrarlos separados por comas.

Por ejemplo:

```
Select Nombre, Apellidos FROM Persona
```

Da como resultado:

+ Opciones	
Nombre	Apellidos
Estefania	Morales HonHon

### 3. Escribir datos

La instrucción que permite escribir en una tabla es **INSERT INTO**.

Por ejemplo:

```
INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
('Luis','Del Morán',38)
```

- **INSERT INTO:** indica una consulta de inserción.

- Persona: nombre de la tabla.
- (Nombre, Apellidos, Edad): nombre de los campos en los que quiere guardar un valor.
- VALUES: palabra clave que precede a la lista de valores.
- ('Luis', 'Del Morán', 38): valores que han entrado en el mismo orden que los nombres de los campos. No olvide escribir los valores de tipo carácter o fecha con apóstrofes o comillas.

Observe que no se ha añadido el campo Id\_persona a la consulta. Esto es normal, ya que este campo es autoincremental; por lo tanto, la base de datos informa automáticamente este campo.

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

The screenshot shows the PHPMyAdmin interface with the SQL tab selected. The query entered is:

```
INSERT INTO Persona(`Nombre`, `Apellidos`, `Edad`)
VALUES(
    'Luna', 'García de Paredes Morales', 6
)
```

The status bar at the top indicates "1 fila insertada" (1 row inserted) and "La id de la fila insertada es: 4 (La consulta tardó 0.0080 seg)" (The inserted row ID is 4 (The query took 0.0080 seconds)).

Haga clic en la pestaña **Examinar** para comprobar que se ha añadido su registro.

The screenshot shows the PHPMyAdmin interface with the Examinar tab selected. The table displays the following data:

	+ Opciones	→ ←	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar  Copiar  Borrar		1	Estefanía	Morales HonHon	38
<input type="checkbox"/>	Editar  Copiar  Borrar		4	Luna	García de Paredes Mo	6

No está obligado a indicar el nombre del campo y su valor si el campo es autoincremental y si acepta el valor NULL.

También puede insertar varios registros al mismo tiempo.

Por ejemplo:

```
INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES ('Estefanía',
'Morales HonHon',37), ('Luna', 'Morales',6), ('María', 'López',25)
```

Ejecute esta consulta en la pestaña **SQL** y haga clic en **Examinar**:

The screenshot shows the PHPMyAdmin interface with the Examinar tab selected. The table displays the following data:

	+ Opciones	→ ←	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar  Copiar  Borrar		1	Estefanía	Morales HonHon	38
<input type="checkbox"/>	Editar  Copiar  Borrar		4	Luna	García de Paredes Mo	6
<input type="checkbox"/>	Editar  Copiar  Borrar		5	David	Morales	50
<input type="checkbox"/>	Editar  Copiar  Borrar		6	Carlos	Sánchez López	45
<input type="checkbox"/>	Editar  Copiar  Borrar		7	María	Malasaña Agora	30

## 4. Filtrar datos

La instrucción que permite filtrar datos en una tabla es WHERE.

Esta instrucción permite recuperar solo cierta información.

Por ejemplo, para recuperar las personas mayores de 48 años:

```
SELECT * FROM Persona WHERE Edad > 48
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

The screenshot shows the PHPMyAdmin interface with the SQL tab selected. The query entered is:

```
SELECT *
FROM persona
WHERE EDAD > 48
LIMIT 0 , 30
```

The results pane displays one row of data:

	id_persona	Nombre	Apellidos	Edad
	5	David	Morales	50

Below the results, there are buttons for Editar, Copiar, Borrar, Cambiar, Borrar, and Exportar.

Puede añadir una o varias condiciones después de la palabra clave WHERE. Deben estar separadas por AND (y) u OR (o bien).

Los distintos operadores son:

- =: igual a.
- <>: diferente a.
- <: estrictamente inferior a.
- <=: inferior o igual a.
- >: estrictamente superior a.
- >=: superior o igual a.
- LIKE: comparación parcial. Este operador se utiliza con el símbolo %, que significa cualquier número de caracteres.
- BETWEEN valor1 AND valor2: filtra entre los valores valor1 y valor2.
- IN (valor1, valor2...): filtra entre los valores.
- IS NULL o IS NOT NULL: comparación con NULL.
- NOT: para invertir la comparación. Puede añadirlo delante BETWEEN, IN y LIKE).

Por ejemplo, para obtener todas las personas cuyo apellido no contiene la letra u y la edad está comprendida entre 20 y 47 años:

```
SELECT * FROM Persona WHERE Nombre NOT LIKE '%u%' AND Edad BETWEEN 20 AND 47
```

Da como resultado:

✓ Mostrando registros 0 - 2 ( 3 total, La consulta tardó 0.0000 seg)

```
SELECT *
FROM PERSONA
WHERE nombre NOT LIKE 'u%'
AND edad
BETWEEN 20
AND 47
LIMIT 0 , 30
```

Mostrar : Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

Ordenar según la clave: Ninguna

+ Opciones

	← T →	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>		1	Estefanía	Morales HonHon	38
<input type="checkbox"/>		6	Carlos	Sánchez López	45
<input type="checkbox"/>		7	Maria	Malasaña Agora	30

Otro ejemplo consiste en obtener todas las personas cuyo apellido está más allá de DEL en orden alfabético y la letra a está presente en el nombre:

```
SELECT * FROM Persona WHERE Apellidos > 'Luna' OR Nombre LIKE '%r%'
```

Da como resultado:

```

SELECT *
FROM PERSONA
WHERE nombre > 'Luna'
AND apellidos LIKE '%r%
LIMIT 0 , 30

```

Mostrar : Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

+ Opciones

<input type="checkbox"/>	<input type="button" value="Editar"/>	<input type="button" value="Copiar"/>	<input type="button" value="Borrar"/>	id_persona	Nombre	Apellidos	Edad
				7	María	Malasaña Agora	30

## 5. Los alias

Un alias permite volver a nombrar un campo o una tabla en la consulta que está en curso.

Su sintaxis es:

```
nombre_tabla AS nombre_alias
```

La palabra clave AS es opcional.

Por ejemplo:

```
Select Nombre, Apellidos, Edad AS Edad_de_la_persona FROM Persona
```

Da como resultado:

+ Opciones		
Nombre	Apellidos	Edad_de_la_persona
Estefanía	Morales HonHon	38
Luna	García de Paredes Mo	6
David	Morales	50
Carlos	Sánchez López	45
María	Malasaña Agora	30

Observe que el campo Edad se ha convertido en Edad\_de\_la\_persona, que solo es válido en esta consulta.

Ejemplo sin utilizar la palabra clave AS:

```
Select Nombre Nombre_de_la_persona FROM Persona
```

Da como resultado:

+ Opciones
<u>Nombre_de_la_persona</u>
Estefanía
Luna
David
Carlos
María

Un alias puede ser muy útil cuando utiliza uniones en su consulta y cuando las tablas tienen el mismo nombre de campos. A continuación veremos un ejemplo.

## 6. Ordenar datos

La instrucción que permite ordenar datos es ORDER BY. Solamente se puede aplicar en SELECT.

Esta palabra clave va seguida de la expresión por la que se va a ordenar.

Esta expresión puede ser:

- Un campo (columna)
- Un alias
- Una expresión (Max, Min...) de una columna

Por ejemplo, para ordenar por nombre:

```
SELECT * FROM Persona ORDER BY Nombre
```

Da como resultado:

+ Opciones	← T →	▼	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/> Editar  Copiar  Borrar			6	Carlos	Sánchez López	45
<input type="checkbox"/> Editar  Copiar  Borrar			5	David	Morales	50
<input type="checkbox"/> Editar  Copiar  Borrar			1	Estefanía	Morales HonHon	38
<input type="checkbox"/> Editar  Copiar  Borrar			4	Luna	García de Paredes Mo	6
<input type="checkbox"/> Editar  Copiar  Borrar			7	María	Malasaña Agora	30

El orden se realiza de manera ascendente (ASC). Para invertir el orden aplicado, debe añadir la palabra clave DESC.

Por ejemplo, para ordenar la siguiente edad de mayor a menor:

```
SELECT * FROM Persona ORDER BY Edad DESC
```

Da como resultado:

+ Opciones		id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar	Copiar	5	David	Morales
<input type="checkbox"/>	Editar	Copiar	6	Carlos	Sánchez López
<input type="checkbox"/>	Editar	Copiar	1	Estefanía	Morales HonHon
<input type="checkbox"/>	Editar	Copiar	7	María	Malasaña Agora
<input type="checkbox"/>	Editar	Copiar	4	Luna	García de Paredes Mo

Puede ordenar siguiendo varios campos, es decir, el orden se realiza a partir del segundo campo si el valor es el mismo que el del primero.

Por ejemplo, para ordenar según la edad y, en caso de edades iguales, por apellidos:

```
SELECT * FROM Persona ORDER BY Edad,Apellidos
```

Da como resultado:

+ Opciones		id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar	Copiar	4	Luna	García de Paredes Mo
<input type="checkbox"/>	Editar	Copiar	7	María	Malasaña Agora
<input type="checkbox"/>	Editar	Copiar	1	Estefanía	Morales HonHon
<input type="checkbox"/>	Editar	Copiar	6	Carlos	Sánchez López
<input type="checkbox"/>	Editar	Copiar	5	David	Morales

## 7. Eliminar datos

La instrucción que permite eliminar datos es **DELETE**.

Por ejemplo:

```
DELETE FROM Persona WHERE id_persona = 5
```

- **DELETE**: indica una consulta de eliminación.
- **Persona**: nombre de la tabla.
- **WHERE id\_persona = 5**: filtra para eliminar la persona con **id\_persona = 5**.

**Atención:** Si olvida la condición que está contenida en la cláusula WHERE, se eliminan todos los registros de la tabla.

## 8. Modificar datos

La instrucción que permite modificar datos es UPDATE.

Por ejemplo:

```
UPDATE Persona SET Nombre = 'Nanie' WHERE id_persona = 1
```

- UPDATE: indica una consulta de modificación.
- Persona: nombre de la tabla.
- SET: palabra clave que precede a los nombres de los campos que va a asignar.
- Nombre = 'Nanie': campo y valor que hay que asignar.
- WHERE id\_persona = 1: filtra para modificar la persona donde el id\_persona es igual a 1, es decir, Estefania Morales Honhon.

Ejecute esta consulta en la pestaña SQL y haga clic en **Examinar**:

	+ Opciones	→ ←	▼	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar  Copiar  Borrar			1	Nanie	Morales HonHon	38
<input type="checkbox"/>	Editar  Copiar  Borrar			4	Luna	García de Paredes Mo	6
<input type="checkbox"/>	Editar  Copiar  Borrar			5	David	Morales	50
<input type="checkbox"/>	Editar  Copiar  Borrar			6	Carlos	Sánchez López	45
<input type="checkbox"/>	Editar  Copiar  Borrar			7	María	Malasaña Agora	30

Observe que en el nombre de la persona con un id\_person=1 se ha modificado su Nombre de Estefanía a Nanie.

Puede modificar varios campos (columnas) a la vez. Solo tiene que añadir los campos con su valor, separados por comas.

Por ejemplo:

```
UPDATE Persona SET Apellidos = 'Manrique Adán', Nombre = 'Luis',
Edad = 45 WHERE id_person = 4
```

Ejecute esta consulta en la pestaña **SQL** y haga clic en **Examinar**:

			<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>
<input type="checkbox"/>		Editar		Copiar		Borrar
1	Nanie	Morales HonHon	38			
4	Luis	Manrique Adán	45			
5	David	Morales	50			
6	Carlos	Sánchez López	45			
7	María	Malasaña Agora	30			

Esta consulta ha modificado el apellido, el nombre y la edad de la persona cuyo id\_persona es igual a 4.

Observe que, al instar consultas de tipo INSERT, debe escribir los valores de tipo VARCHAR, TEXT o DATE con apóstrofes o dobles comillas.

Si no introduce la palabra clave WHERE, no hay filtro y por lo tanto las modificaciones se realizan en todos los registros.

Puede modificar varios registros al mismo tiempo.

Por ejemplo, para modificar la edad a 55 años de todas aquellas personas mayores o igual a de 50 años:

```
UPDATE Persona SET Edad = 55 WHERE Edad >= 50
```

Ejecute esta consulta en la pestaña **SQL** y haga clic en **Examinar**:

			<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>
<input type="checkbox"/>		Editar		Copiar		Borrar
1	Nanie	Morales HonHon	38			
4	Luis	Manrique Adán	45			
5	David	Morales	55			
6	Carlos	Sánchez López	45			
7	María	Malasaña Agora	30			

## 9. Las uniones

En una consulta de tipo SELECT, a veces es muy útil traer información de dos tablas que se corresponden entre sí.

Por ejemplo, cree una tabla Idiomas que contenga dos campos:

- Id: Int, autoincremental, Primary
- Etiqueta: Varchar (20)

Haga clic en su base de datos (\_prueba) y escriba "Idiomas" en la zona "Crear nueva tabla en la base de datos \_prueba". Haga clic en **Continuar**, rellene la estructura de los dos campos introduciendo como índices el Id autoincremental y Primary. Haga clic en **Grabar**.

- Seleccione la tabla **Idioma** y haga clic en **Estructura**.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	Id	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar  Eliminar  Primaria  Único  Índice  Espacial  Texto completo  Más
2	Etiqueta	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice  Espacial  Texto completo  Más

- Haga clic en la pestaña **Insertar** para añadir: Francés, Inglés.

- Haga clic en **Examinar**:

+ Opciones		← T →	↓	Id	Etiqueta
<input type="checkbox"/>	Editar  Copiar  Borrar	1	Francés		
<input type="checkbox"/>	Editar  Copiar  Borrar	2	Inglés		

Para añadir la opción del idioma en la tabla Persona, haga clic en ésta en la tabla Persona y en la pestaña **Estructura**. Haga clic en **Continuar** para añadir un campo al final de la tabla. Este campo se llama "Id\_idioma" y es de tipo "int". Haga clic en **Grabar**.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/>	id_persona	int(11)			No	None		
<input type="checkbox"/>	Nombre	varchar(20)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	Apellidos	varchar(20)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	Edad	int(11)			No	None		
<input type="checkbox"/>	Id_idioma	int(11)			No	None		

Muestre los datos de la tabla Persona:

+ Opciones		→ T ←	↓	id_persona	Nombre	Apellidos	Edad	Id_idioma
<input type="checkbox"/>	Editar  Copiar  Borrar	1	Nanie	Morales HonHon	38		0	
<input type="checkbox"/>	Editar  Copiar  Borrar	4	Luis	Manrique Adán	45		0	
<input type="checkbox"/>	Editar  Copiar  Borrar	5	David	Morales	55		0	
<input type="checkbox"/>	Editar  Copiar  Borrar	6	Carlos	Sánchez López	45		0	
<input type="checkbox"/>	Editar  Copiar  Borrar	7	María	Malasaña Agora	30		0	

Si quiere que Nanie Morales tenga como idioma "Francés", modifique a 1 (haciendo clic en el lápiz) el valor de su Id\_idioma.

+ Opciones

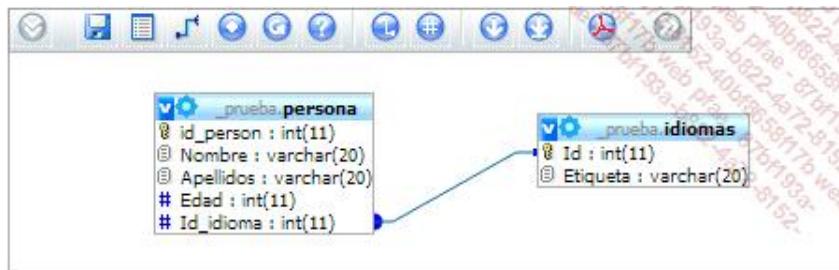
			<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>	<b>Id_idioma</b>
<input type="checkbox"/>		Editar		Copiar		Borrar	1
<input type="checkbox"/>		Editar		Copiar		Borrar	4
<input type="checkbox"/>		Editar		Copiar		Borrar	5
<input type="checkbox"/>		Editar		Copiar		Borrar	6
<input type="checkbox"/>		Editar		Copiar		Borrar	7

Si desea que Luis y David tengan como idioma el "Inglés", modifique a 2 (haciendo clic en el lápiz) el valor de su Id\_idioma.

+ Opciones

			<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>	<b>Id_idioma</b>
<input type="checkbox"/>		Editar		Copiar		Borrar	1
<input type="checkbox"/>		Editar		Copiar		Borrar	4
<input type="checkbox"/>		Editar		Copiar		Borrar	5
<input type="checkbox"/>		Editar		Copiar		Borrar	6
<input type="checkbox"/>		Editar		Copiar		Borrar	7

Las dos tablas pueden estar unidas de esta manera:



Las instrucciones que permiten realizar una unión entre dos tablas es JOIN y ON.

Por ejemplo, para realizar una unión entre la tabla Persona y la tabla Idiomas:

```
SELECT * FROM Persona JOIN Idiomas ON Persona.Id_idioma =
Idiomas.Id
```

Da como resultado:

+ Opciones						
<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>	<b>Id_idioma</b>	<b>Id</b>	<b>Etiqueta</b>
1	Nanie	Morales HonHon	38	1	1	Francés
4	Luis	Manrique Adán	45	2	2	Inglés
5	David	Morales	55	2	2	Inglés

- `SELECT *:` selecciona todos los campos de las tablas Persona e Idiomas.
- `Persona:` nombre de la tabla.
- `JOIN Idiomas:` unión en la tabla Idiomas.
- `Persona.Id_idioma = Idiomas.Id:` correspondencia entre los campos de las dos tablas.

Observe que solo se han mostrado tres personas. Esto ocurre porque JOIN realiza por defecto una unión interna, es decir, la consulta vuelve a enviar los datos que tienen una correspondencia entre las dos tablas. Como Estefanía, María y Carlos tienen un `Id_idioma` igual a 0 y 0 no existe en la tabla Idiomas, no aparecen. La unión interna se designa INNER JOIN.

Por el contrario, la unión externa devuelve todos los registros, incluso aquellos que no tienen correspondencia entre las dos tablas. Una unión externa se designa LEFT JOIN o RIGHT JOIN.

En el ejemplo anterior, si quiere mostrar todas las personas, incluidas aquellas que no tienen correspondencia en la tabla Idiomas, debe utilizar LEFT JOIN.

```
SELECT * FROM Persona LEFT JOIN Idiomas ON Persona.Id_idioma =
Idiomas.Id
```

Da como resultado :

+ Opciones						
<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>	<b>Id_idioma</b>	<b>Id</b>	<b>Etiqueta</b>
1	Nanie	Morales HonHon	38	1	1	Francés
4	Luis	Manrique Adán	45	2	2	Inglés
5	David	Morales	55	2	2	Inglés
6	Carlos	Sánchez López	45	0	NULL	NULL
7	María	Malasaña Agora	30	0	NULL	NULL

La instrucción LEFT JOIN devuelve todos los datos de la tabla a la izquierda (Persona) incluso si no tienen correspondencia en la tabla de la derecha (Idiomas). MySQL rellena las zonas sin correspondencia con los valores NULL. La unión externa se designa OUTER JOIN.

La instrucción RIGHT JOIN devuelve todos los Idiomas, incluso si no tienen correspondencia en la tabla Persona, pero esto no tiene mucho sentido funcional.

Existe otra manera de escribir la unión interna, aunque esta sintaxis está obsoleta:

```
SELECT * FROM Persona,Idiomas WHERE Persona.Id_idioma = Idiomas.Id
```

equivale a:

```
SELECT * FROM Persona JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id
```

Puede añadir varias uniones.



La sintaxis con las tres tablas llamadas tabla1, tabla2 y tabla3 con tabla1 unida a la tabla2 con el Id\_table2 y tabla2 unida a la tabla3 con el Id\_table3 es:

```
SELECT *
FROM Tabla1
INNER JOIN Tabla2 ON Tabla1.Id_table2 = Tabla2.Id
INNER JOIN Tabla3 ON Tabla2.Id_table3 = Tabla3.Id
```

Puede combinar los filtros (WHERE), el orden (ORDER BY) y los alias con las uniones.

Por ejemplo:

```
Select Nombre, apellidos, Etiqueta as Idiomas
FROM Persona
LEFT JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id
WHERE apellidos LIKE '%a%' ORDER BY Nombre
```

Muestra en orden alfabético los apellidos, el nombre y el idioma de las personas. El campo apellidos contiene la cadena de caracteres 'a':

+ Opciones		
NOMBRE	APELLIDOS	IDIOMA
Carlos	Sánchez López	NULL
David	Morales	Inglés
María	Malasaña Agora	NULL
Nanie	Morales HonHon	Francés

## 10. El agrupamiento

A continuación la tabla Persona se completa de la siguiente manera:

<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>	<b>Id_idioma</b>
1	Nanie	Morales HonHon	50	1
2	Luis	Manrique Adán	45	2
3	David	Morales	50	2
4	Carlos	Sánchez López	45	0
5	María	Malasaña Agora	30	0

La instrucción que permite agrupar datos es GROUP BY. Se utiliza con las funciones:

- SUM: suma
- MIN: mínimo
- MAX: máximo
- AVG: medio
- COUNT: número de elementos

Por ejemplo, para conocer la edad de la persona más mayor:

```
SELECT MAX(Edad) as Edad_Máxima FROM Persona
```

Da como resultado:



Aquí los datos no se agrupan, porque se trata de obtener la edad máxima entre todas las personas.

Si quiere obtener la edad máxima de las personas por idioma, debe agrupar en el identificador del idioma:

```
SELECT MAX(Edad) as Edad_Máxima, Etiqueta
FROM Persona
LEFT JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id
GROUP BY Id_idioma
```

Da como resultado:

+ Opciones	
Edad_Máxima	ETIQUETA
45	NULL
50	Francés
50	Inglés

Para obtener el número de personas por idioma:

```
SELECT COUNT(*) as Número_Personas, Etiqueta
FROM Persona
LEFT JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id
GROUP BY Id_idioma
```

Da como resultado:

+ Opciones	
Número_Personas	ETIQUETA
2	NULL
1	Francés
2	Inglés

En una consulta de agrupamiento, no puede utilizar el filtro WHERE. Utilice la cláusula HAVING para filtrar el resultado. Las condiciones que debe poner después de la palabra clave HAVING son las mismas que en la palabra clave WHERE.

Para obtener el número de personas por un idioma que no sea el inglés:

```
SELECT COUNT(*) as Número_Personas, Etiqueta
FROM Persona
LEFT JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id
GROUP BY Id_idioma
HAVING Etiqueta <>
```

Da como resultado:

+ Opciones	
Número_Personas	ETIQUETA
1	Francés
2	Inglés

Otro ejemplo consiste en mostrar la edad media de las personas que tienen la cadena de caracteres "a" en sus apellidos:

```
SELECT AVG(Edad) as Edad_Promedio FROM Persona WHERE Nombre like '%a%'
```

Da como resultado:

+ Opciones
<b>Edad_Promedio</b>
43.7500

Aquí no se ha utilizado HAVING porque no se agrupa con GROUP BY.

# SQL avanzado

## 1. Las funciones e instrucciones SQL

El objetivo de este capítulo no es que aprenda todas las funciones SQL, sino solo las más utilizadas.

Para el resto de este capítulo, la tabla Persona contiene estos datos:

	+ Opciones	→ ←	▼	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar  Copiar  Borrar			1	Nanie	Morales HonHon	55
<input type="checkbox"/>	Editar  Copiar  Borrar			2	David	Manrique Adán	48
<input type="checkbox"/>	Editar  Copiar  Borrar			5	María	Malasaña Agora	36
<input type="checkbox"/>	Editar  Copiar  Borrar			6	Roberto	Magalán	63
<input type="checkbox"/>	Editar  Copiar  Borrar			7	David	Olís De Las Heras	33
<input type="checkbox"/>	Editar  Copiar  Borrar			8	Rodolfo	Germán	64
<input type="checkbox"/>	Editar  Copiar  Borrar			9	Rula	Campos	42
<input type="checkbox"/>	Editar  Copiar  Borrar			10	Martín	De Julián	57
<input type="checkbox"/>	Editar  Copiar  Borrar			11	Manuel	Barberá	45

### a. Limitar datos

La instrucción que permite limitar el número de registros devueltos por SELECT es: LIMIT número\_registro.

Por ejemplo:

```
SELECT * FROM Persona ORDER BY Nombre LIMIT 3
```

Esta consulta significa: leer los tres primeros registros que contienen todos los campos de la tabla Persona ordenados por Nombre.

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

	+ Opciones	← →	▼	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar  Copiar  Borrar			2	David	Manrique Adán	48
<input type="checkbox"/>	Editar  Copiar  Borrar			7	David	Olís De Las Heras	33
<input type="checkbox"/>	Editar  Copiar  Borrar			11	Manuel	Barberá	45

También puede añadir el número de registro con el que quiere recuperar datos agregando este número antes del número de registros.

Por ejemplo, para recuperar los registros de la quinta a la sexta, ambas incluidas:

```
SELECT * FROM Persona ORDER BY Nombre LIMIT 4,2
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones			<b>id_persona</b>	<b>Nombre</b>	<b>Apellidos</b>	<b>Edad</b>
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	10	Martín	De Julián
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	Nanie	Morales HonHon
						55

### **b. Valores distintos**

La instrucción que permite recuperar únicamente los valores de registros distintos devueltos por SELECT es DISTINCT.

Por ejemplo:

```
SELECT DISTINCT Nombre FROM Persona ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>NOMBRE</b>
David
Manuel
Maria
Martin
Nanie
Roberto
Rodolfo
Rula

En realidad, dos personas se llaman David pero este nombre sólo aparece una vez con la palabra clave DISTINCT.

### c. Convertir en mayúsculas

La instrucción que permite convertir el valor de un campo en mayúsculas es `UPPER(nombre_del_campo)`.

Por ejemplo, para mostrar el apellido en mayúsculas:

```
Select Nombre, UPPER(apellidos) FROM Persona ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

NOMBRE	UPPER(APELLIDOS)
David	MANRIQUE ADÁN
David	OLÍS DE LAS HERAS
Manuel	BARBERÁ
Maria	MALASAÑA AGORA
Martín	DE JULIÁN
Nanie	MORALES HONHON
Roberto	MAGALÁN
Rodolfo	GERMÁN
Rula	CAMPOS

Puede crear un alias para UPPER(apellidos) de esta manera:

```
Select Nombre, UPPER(apellidos) as Apellidos_en_Mayúsculas FROM Persona
ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

NOMBRE	Apellidos_en_Mayúsculas
David	MANRIQUE ADÁN
David	OLÍS DE LAS HERAS
Manuel	BARBERÁ
Maria	MALASAÑA AGORA
Martín	DE JULIÁN
Nanie	MORALES HONHON
Roberto	MAGALÁN
Rodolfo	GERMÁN
Rula	CAMPOS

#### d. Convertir en minúsculas

La instrucción que permite convertir en minúsculas el valor de un campo es LOWER(nombre\_del\_campo).

Por ejemplo, para mostrar los apellidos en minúsculas:

```
Select Nombre, LOWER(Apellidos) as Apellidos_en_Minúsculas FROM Persona
ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
NOMBRE	Apellidos_en_Minúsculas
David	manrique adán
David	olís de las heras
Manuel	barberá
María	malasaña agora
Martín	de julián
Nanie	morales honhon
Roberto	magalán
Rodolfo	germán
Rula	campos

### e. Redondear un número decimal

La instrucción que permite redondear un número decimal es ROUND( ).

Esta función toma como argumento el nombre del campo y el número de cifras después de la coma.

Si tiene el campo precio\_iva = 31.2698754 y solo quiere las dos cifras después de la coma, escriba:

```
ROUND(precio_iva,2)
```

Y devuelve 31.27.

### f. Valor absoluto de un número decimal

La instrucción que permite tomar el valor absoluto de un número decimal es ABS( ).

Esta función toma como argumento el nombre del campo y devuelve el valor absoluto de este número.

Si tiene el campo temperatura = -31.2698754 y solo quiere el valor absoluto, escriba:

```
ABS(temperatura)
```

Y devuelve 31.2698754.

### g. Número aleatorio

La instrucción que permite devolver un número aleatorio entre 0 y 1 es RAND( ).

Por ejemplo:

```
SELECT rand()
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
rand()
0.7464317155503032

Por lo tanto, para obtener un número entero aleatorio entre 0 y 30:

```
SELECT round(rand()*30) as Número aleatorio
```

## **h. Longitud de un campo**

La instrucción que permite devolver la longitud de un campo es LENGTH(nombre\_del\_campo).

Por ejemplo, para mostrar la longitud del campo Nombre:

```
Select Nombre, LENGTH(Nombre) as Longitud_Nombre
FROM Persona ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

NOMBRE	Longitud_Nombre
David	5
David	5
Manuel	6
Maria	5
Martín	6
Nanie	5
Roberto	7
Rodolfo	7
Rula	4

## **i. Eliminar los espacios de un campo**

La instrucción que permite eliminar los espacios de un campo es TRIM(nombre\_del\_campo).

Por ejemplo:

```
SELECT TRIM('    Ejemplo    ') as Campo1
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:



También existe la función LTRIM( ) para eliminar los espacios a la izquierda y RTRIM( ) para eliminar los espacios a la derecha.

### j. Extraer una subcadena de un campo

La instrucción que permite extraer una subcadena de un campo o una cadena de caracteres es SUBSTR( ).

Esta función toma como argumentos el nombre del campo, la posición de inicio y como opción la longitud de la subcadena que va a recuperar.

Por ejemplo, para mostrar los tres primeros caracteres del nombre:

```
SELECT SUBSTR(Nombre,1,3) as Nombre_Partido FROM Persona ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
Nombre_Partido
Dav
Dav
Man
Mar
Mar
Nan
Rob
Rod
Rul

### k. Concatenar varios campos

La instrucción que permite concatenar campos y cadenas de caracteres es CONCAT( ).

Esta función toma como argumento el nombre de los campos o de las cadenas de caracteres que quiere concatenar (juntar).

Por ejemplo, para mostrar el nombre y los apellidos separados por un espacio:

```
SELECT CONCAT(Nombre, ' ', Apellidos) as Nombre_Apellidos  
FROM Persona ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>Nombre_Apellidos</b>
David Manrique Adán
David Olís De Las Heras
Manuel Barberá
Maria Malasaña Agora
Martín De Julián
Nanie Morales HonHon
Roberto Magalán
Rodolfo Germán
Rula Campos

## I. Posición de una cadena de caracteres en un campo

La instrucción que permite devolver la posición de una cadena de caracteres en un campo es INSTR( ).

Esta función toma como argumentos el nombre del campo o de la cadena de caracteres de inicio y la cadena de caracteres que hay que buscar. Esta función devuelve 0 si no se ha encontrado la cadena.

Por ejemplo, para mostrar la posición de 'ar' en el nombre de las personas:

```
Select Nombre, INSTR(Nombre, 'ar') as posicion FROM Persona ORDER
BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
NOMBRE	Posición
David	0
David	0
Manuel	0
Maria	2
Martín	2
Nanie	0
Roberto	0
Rodolfo	0
Rula	0

## m. Añadir una secuencia de caracteres

La instrucción que permite devolver una cadena completada por la izquierda por una secuencia de caracteres es LPAD( ).

Esta función toma como argumentos el campo o la cadena de caracteres, la longitud total del campo con los caracteres que se han añadido y el carácter que hay que añadir.

Por ejemplo, para mostrar el carácter ":" antes de los apellidos de manera que el campo tenga 25 caracteres de longitud total:

```
SELECT LPAD(Apellidos,25,'::') FROM Persona
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
<b>LPAD(APELLIDOS,25,'::')</b>	
.....	Morales HonHon
.....	Manrique Adán
.....	Malasaña Agora
.....	Magalán
.....	Olís De Las Heras
.....	Germán
.....	Campos
.....	De Julián
.....	Barberá

De la misma manera la función RPAD( ) completa a la derecha.

#### n. Sustitución de una cadena de caracteres

La instrucción que permite sustituir una cadena de caracteres en un campo o en otra cadena de caracteres es REPLACE( ).

Esta función toma como argumentos la cadena de caracteres principal o el campo, la cadena de caracteres que se busca y la cadena de caracteres de sustitución.

Por ejemplo, para sustituir 'Da' por 'Da--' en el nombre de las personas:

```
SELECT REPLACE(Nombre , 'Da' , 'Da--') as Nombre_sustituido FROM Persona
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
Nombre_Sustituido
Nanie
Da-vid
María
Roberto
Da-vid
Rodolfo
Rula
Martín
Manuel

## o. Comprobar el valor de un campo

La instrucción que permite comprobar una expresión (como el valor de un campo por ejemplo) es **IF( )**.

Esta función toma como argumentos la condición, el valor que tiene que devolver si la condición es verdadera y el valor que tiene que devolver si la condición es falsa.

Por ejemplo, para mostrar "Senior" en personas de más de 60 años y "Otro" para las otras personas:

```
Select Nombre, Apellidos, Edad, IF(Edad > 60,'Senior','Otro')
as Categoría FROM Persona ORDER BY Nombre
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones				
NOMBRE	APELLIDOS	EDAD	Categoría	
David	Manrique Adán	48	Otro	
David	Olís De Las Heras	33	Otro	
Manuel	Barberá	45	Otro	
María	Malasaña Agora	36	Otro	
Martín	De Julián	57	Otro	
Nanie	Morales HonHon	55	Otro	
Roberto	Magalán	63	Senior	
Rodolfo	Germán	64	Senior	
Rula	Campos	42	Otro	

La otra instrucción que permite comprobar el valor de un campo es **CASE WHEN THEN**.

Esta instrucción puede contener varias veces la palabra clave **WHEN** y por lo tanto puede obtener varios resultados posibles según las condiciones que se han pasado en **WHEN**.

Por ejemplo, para mostrar "Senior" en personas de más de 60 años, "Medio" en personas entre 35 y 60 años y "Junior" en personas de menos de 35 años:

```

Select Nombre,
       Apellidos,
       Edad,
       CASE WHEN Edad > 60 THEN 'SENIOR'
             WHEN Edad BETWEEN 35 AND 60 THEN 'Medio'
             WHEN Edad < 35 THEN 'Junior'
         END as Categoría
FROM Persona
ORDER BY Nombre

```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones			
NOMBRE	APELLIDOS	EDAD	Categoría
David	Manrique Adán	48	Medio
David	Olís De Las Heras	33	Junior
Manuel	Barberá	45	Medio
Maria	Malasaña Agora	36	Medio
Martín	De Julián	57	Medio
Nanie	Morales HonHon	55	Medio
Roberto	Magalán	63	Senior
Rodolfo	Germán	64	Senior
Rula	Campos	42	Medio

#### p. Examinar la fecha actual

Las funciones CURDATE( ) y CURRENT\_DATE( ) devuelven la fecha actual en formato aaaa-mm-dd.

Las funciones CURTIME( ) y CURRENT\_TIME( ) devuelven la hora actual en formato hh:mm:ss.

Las funciones NOW( ), LOCALTIME( ) y CURRENT\_TIMESTAMP( ) devuelven la fecha y la hora actuales.

Por ejemplo:

```
SELECT CURDATE(), CURTIME(), NOW(), LOCALTIME(), CURRENT_TIMESTAMP()
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones				
CURDATE()	CURTIME()	NOW()	LOCALTIME()	CURRENT_TIMESTAMP()
2013-12-26	11:50:02	2013-12-26 11:50:02	2013-12-26 11:50:02	2013-12-26 11:50:02

#### q. Extraer la fecha de un campo date y hora

La instrucción que permite devolver la parte de la fecha de un campo que contiene la fecha y la hora es DATE( ).

Esta función toma como argumento el campo o la fecha completa.

Por ejemplo:

```
SELECT NOW() as fecha_hora, DATE(NOW()) as solo_fecha
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
Fecha_Hora	Solo_Fecha
2013-12-26 11:51:01	2013-12-26

#### r. Diferencia entre dos fechas

La instrucción que permite devolver la diferencia en número de días entre dos fechas es **DATEDIFF( )**.

Esta función toma como argumentos el primer campo o una fecha, y a continuación un segundo campo o una fecha.

Por ejemplo:

```
SELECT DATEDIFF('2013-05-07', '2013-02-02') as número_días
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
Número_Días	
94	

#### s. Añadir un intervalo de tiempo a una fecha

La instrucción que permite añadir un intervalo de tiempo a una fecha es **DATE\_ADD( )** o **ADDDATE( )**.

Esta función toma como argumentos la fecha o el campo y la palabra clave **INTERVAL** seguida del valor y de la unidad (días, mes, semana...).

Por ejemplo, para añadir un día a la fecha 10/01/2013:

```
SELECT DATE_ADD('2013-01-10', INTERVAL 1 DAY) as día_siguiente
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>Día_Siguiente</b>
2013-01-11

Las palabras clave de los intervalos son:

- MICROSECOND: microsegundos
- SECOND: segundos
- MINUTE: minutos
- HOUR: horas
- DAY: días
- WEEK: semanas
- MONTH: meses
- QUARTER: trimestres
- YEAR: años
- MINUTE\_SECOND: minutos y segundos en formato m:s
- HOUR\_SECOND: horas, minutos y segundos en formato h:m:s
- HOUR\_MINUTE: horas y minutos en formato h:m
- DAY\_SECOND : días, horas, minutos y segundos en formato d h:m:s
- YEAR\_MONTH : año y mes en formato a-m

Por ejemplo, para añadir 2 años y 1 mes a la fecha 10/01/2013:

```
SELECT DATE_ADD('2013-01-10', INTERVAL '2-1' YEAR_MONTH) as nueva_fecha
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>Nueva_Fecha</b>
2015-02-10

Si la fecha se muestra en hexadecimal, seleccione la opción **Mostrar el contenido binario en hexadecimal**, a la que se accede por el enlace **Opciones** situado encima del resultado de la consulta.

#### t. Añadir un intervalo de tiempo a una hora

La instrucción que permite añadir un intervalo de tiempo a una hora es ADDTIME( ).

Esta función toma como argumentos la hora o el campo que contiene la hora y el intervalo de tiempo que debe añadir en formato hh:mm:ss.μμμμμμ.

Por ejemplo, para añadir un minuto y dos segundos a 22h58m02s:

```
SELECT ADDTIME('22:58:02','00:01:02') as nueva_hora
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>Nueva_Hora</b>
22:59:04

#### **u. Sustracción de un intervalo de tiempo a una fecha**

La instrucción que permite sustraer un intervalo de tiempo a una fecha es DATE\_SUB( ) o SUBDATE( ).

Esta función toma los mismos argumentos que los de la función DATE\_ADD( ) y opera según el mismo principio, salvo que sustrae el intervalo de tiempo.

Por ejemplo, para sustraer 41 días de la fecha 10/01/2013:

```
SELECT DATE_SUB('2013-01-10', INTERVAL '41' DAY) as nueva_fecha
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>Nueva_Fecha</b>
2012-11-30

#### **v. Sustracción de un intervalo de tiempo a una hora**

La instrucción que permite sustraer un intervalo de tiempo a una hora es SUBTIME( ).

Esta función toma los mismos argumentos que los de la función ADDTIME( ) y funciona con el mismo principio, salvo que sustrae el intervalo de tiempo.

Por ejemplo, para sustraer 2 horas, 5 minutos y 20 segundos a 22h58m02s:

```
SELECT SUBTIME('22:58:02','02:05:20') as nueva_hora
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
<b>Nueva_Hora</b>
20:52:42

## w. Unir dos consultas

La instrucción que permite unir dos consultas es UNION.

Esta instrucción se ubica entre dos consultas que deben devolver el mismo número de columnas y del mismo tipo.

Por ejemplo, para unir esta consulta que devuelve los apellidos y el nombre de personas mayores de 50 años:

```
Select Nombre, Apellidos FROM Persona WHERE Edad > 50
```

Y esta consulta que devuelve los apellidos y el nombre de personas cuyo nombre es David:

```
Select Nombre, Apellidos FROM Persona WHERE Nombre = 'David'
```

Escriba lo siguiente:

```
Select Nombre, Apellidos FROM Persona WHERE Edad > 50  
UNION  
Select Nombre, Apellidos FROM Persona WHERE Nombre = 'David'
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
NOMBRE	APELLIDOS
Nanie	Morales HonHon
Roberto	Magalán
Rodolfo	Germán
Martín	De Julián
David	Manrique Adán
David	Olís De Las Heras

Puede introducir varios UNION.

## 2. Las subconsultas

Una subconsulta es una consulta llamada en otra consulta. Una subconsulta se escribe entre paréntesis y puede contener otras subconsultas.

Hay varios tipos de subconsultas:

### Subconsulta escalonada

Esta subconsulta solo devuelve una columna con un único registro.

Por ejemplo, la edad media de personas:

```
SELECT AVG(edad) FROM Persona
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
AVG(EDAD)
49.2222

He aquí una consulta que muestra las personas que tienen una edad menor que el resultado de la subconsulta anterior:

```
Select Nombre, Apellidos  
FROM Persona  
WHERE Edad < (  
    SELECT AVG(edad)  
    FROM Persona  
)
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones
NOMBRE APELLIDOS
David Manrique Adán
Maria Malasaña Agora
David Olis De Las Heras
Rula Campos
Manuel Barberá

### **Subconsulta que reenvía una lista**

Puede utilizar como criterio de comparación el operador IN. Este operador devuelve la cifra 1 si se presenta en la lista una expresión.

Por ejemplo:

```
SELECT 'Estefanía' IN ('Pablo','Estefanía','Luna')
```

Devuelve 1.

Otro ejemplo: mostrar las personas cuya edad es 48, 55, 36 años:

```
Select Nombre, Apellidos FROM Persona WHERE Edad IN (48,55,36)
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
NOMBRE	APELLIDOS
Nanie	Morales HonHon
David	Manrique Adán
María	Malasaña Agora

También puede poner una subconsulta en lugar de un grupo de cifras.

Por ejemplo, para mostrar las personas que tienen la misma edad que aquellas cuyo nombre comienza por 'D':

```
SELECT Apellidos, Nombre
FROM Persona
WHERE Edad IN (
    SELECT Edad
    FROM Persona
    WHERE Nombre LIKE 'D%.'
)
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

+ Opciones	
NOMBRE	APELLIDOS
David	Manrique Adán
David	Olis De Las Heras

De hecho, `SELECT Edad FROM Persona WHERE Nombre like 'D%'` devuelve la edad de David Manrique y David Olis.

También puede utilizar el operador NOT IN para excluir los elementos de una lista.

Para obtener las otras personas que no están en la consulta anterior:

```
Select Nombre, Apellidos
FROM Persona
WHERE Edad NOT IN (
    SELECT Edad
    FROM Persona
    WHERE Nombre LIKE 'D%'
)
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

NOMBRE	APELLIDOS
Nanie	Morales HonHon
María	Malasaña Agora
Roberto	Magalán
Rodolfo	Germán
Rula	Campos
Martín	De Julián
Manuel	Barberá

La palabra clave EXISTS permite probar la existencia de datos de una subconsulta.

Por ejemplo, si quiere mostrar todas las personas, siempre y cuando al menos una de ellas sea mayor o igual a 62 años:

```
Select Nombre, Apellidos, Edad FROM Persona WHERE EXISTS (SELECT *
FROM Persona WHERE Edad >= 62)
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

NOMBRE	APELLIDOS	EDAD
Nanie	Morales HonHon	55
David	Manrique Adán	48
María	Malasaña Agora	36
Roberto	Magalán	63
David	Olís De Las Heras	33
Rodolfo	Germán	64
Rula	Campos	42
Martín	De Julián	57
Manuel	Barberá	45

La consulta `SELECT * FROM Persona WHERE Edad >= 62` devuelve al menos un registro, por lo tanto `WHERE EXISTS` es verdadero y la consulta `Select Nombre, Apellidos, Edad FROM Persona` devuelve todo el mundo.

También puede utilizar NOT EXISTS para probar la ausencia de datos de una subconsulta.

### **Subconsulta correlacionada**

Una subconsulta correlacionada o anidada es una subconsulta que se ejecuta en cada registro de la consulta principal, ya que la columna de la subconsulta hace referencia a una columna de la consulta principal. Por lo tanto, cada registro de la consulta principal puede cambiar con el resultado del registro de la subconsulta.

Por ejemplo, para mostrar todas las personas con los mismos nombres:

```

SELECT Id_person, Nombre, Apellidos
FROM persona P1
WHERE Nombre IN (
    Select Nombre
    FROM persona P2
    WHERE P1.Id_person <> P2.Id_person
)

```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

The screenshot shows a table with three columns: ID\_PERSONA, NOMBRE, and APELLIDOS. The first row has ID\_PERSONA=7, NOMBRE=David, and APELLIDOS=Olís De Las Heras. The second row has ID\_PERSONA=2, NOMBRE=David, and APELLIDOS=Manrique Adán.

ID_PERSONA	NOMBRE	APELLIDOS
7	David	Olís De Las Heras
2	David	Manrique Adán

Observe que la cláusula WHERE de la subconsulta hace referencia a la tabla P1 de la consulta principal.

Esta consulta busca todos los registros que tienen los mismos apellidos en la tabla Persona, pero no el mismo Id\_person; de lo contrario, la consulta devolvería todo el mundo.

### 3. Los procedimientos almacenados y funciones

Los procedimientos almacenados y funciones permiten ejecutar instrucciones SQL almacenadas en la base de datos.

Una función devuelve un valor de manera sistemática, lo que no es obligatorio en el caso de los procedimientos.

Una función se utiliza básicamente en una consulta SQL, mientras que un procedimiento es un programa autónomo que se puede ejecutar con PHP.

Por tanto una función no puede tener:

- transacción, ya que la consulta utiliza la transacción.
- actualización (INSERT, UPDATE, DELETE...).
- gestión de errores.
- llamada al procedimiento.
- orden SQL de modificación de la estructura de la base de datos.

No se puede crear una tabla temporal.

Estas funcionalidades aparecen con MySQL 5 y permiten disponer de consultas SQL que se vuelven a utilizar y que además son más seguras.

Si tiene una consulta SQL que se ejecuta en 10 páginas PHP distintas, puede ser interesante tener en su lugar un procedimiento almacenado que contenga esta consulta y que sea este procedimiento el que se ejecute 10 veces. Si quiere modificar su consulta SQL, solo la tiene que cambiar una vez en el procedimiento que se haya almacenado.

PHPMyAdmin no permite crear un procedimiento almacenado o una función con sus menús. Por lo tanto, escriba el código SQL que permite crearlo.

Por ejemplo, para crear un procedimiento almacenado sin argumento llamado creacion\_persona:

```
CREATE PROCEDURE creacion_persona()
BEGIN
INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES ('Martín',
'Alonso', 64);
END//
```

Las palabras clave CREATE PROCEDURE significan que va a crear un procedimiento. A continuación escriba su nombre seguido de la palabra clave BEGIN y las instrucciones SQL seguidas de la palabra clave END. Atención, añada el delimitador // al final del procedimiento almacenado.

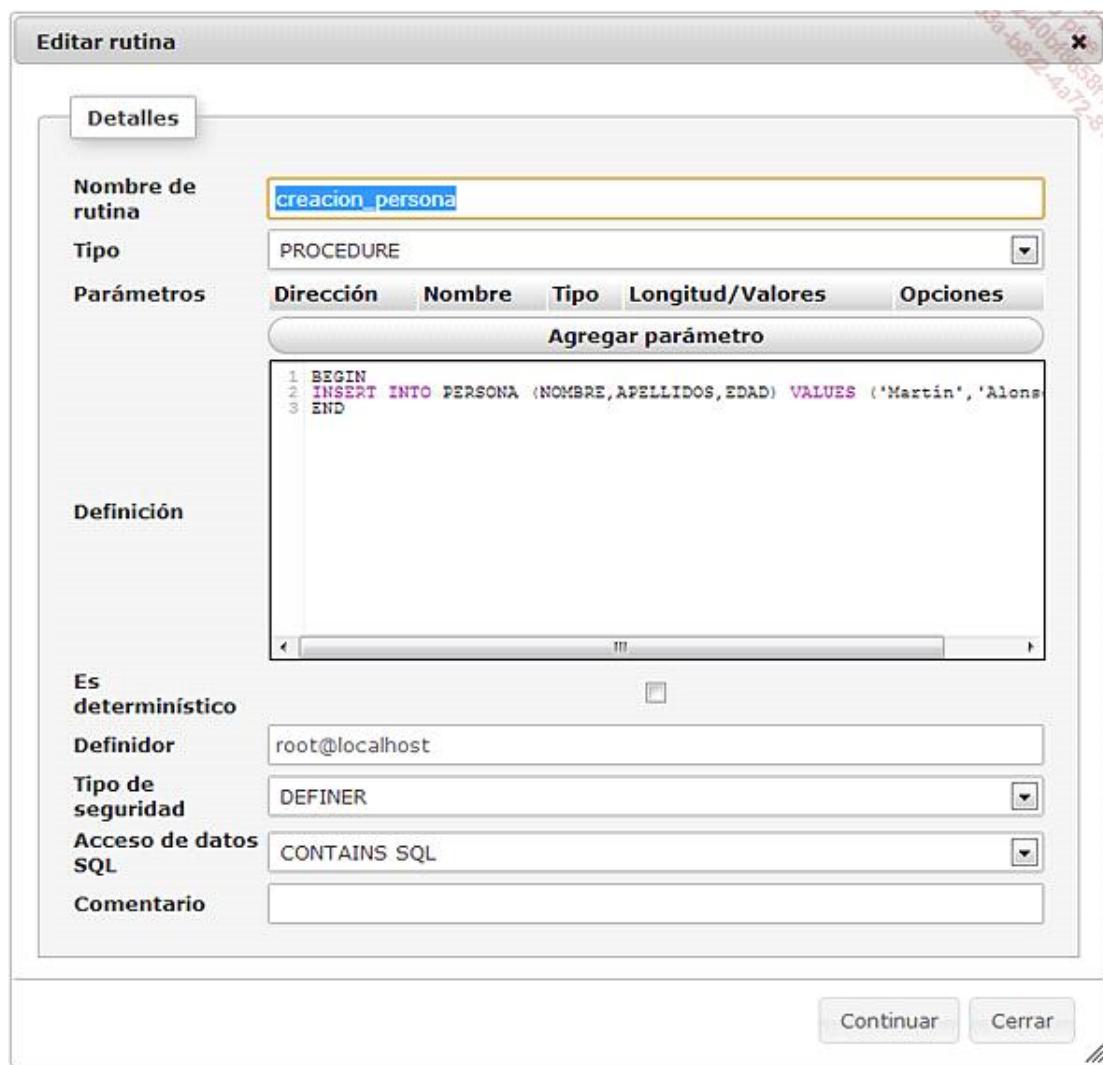
Copie esta consulta en la pestaña **SQL** de PHPMyAdmin, cambie el delimitador a // en la zona **Delimitador**, a la izquierda de **Continuar**, y haga clic en **Continuar**.

Haga clic en la pestaña **Estructura** y en el enlace **Rutinas**:

Rutinas			
Nombre	Acción	Tipo	Retorna
creacion_persona	Editar ▶ Ejecutar Exportar Eliminar	PROCEDURE	

Se muestra su procedimiento almacenado y varias herramientas le permiten modificarlo, ejecutarlo, exportarlo o eliminarlo.

Cuando hace clic en **Editar**, se abre una nueva ventana.



En esta ventana puede cambiar su nombre, tipo (procedimiento o función), definición o creador.

Un procedimiento o función se considera "determinista" si siempre devuelve el mismo resultado para los mismos parámetros de entrada.

El **Tipo de seguridad** permite elegir en DEFINER o INVOKER. En el primer caso, el procedimiento se almacena y ejecuta con los permisos del usuario que lo ha creado. Esto puede plantear un problema si exporta su procedimiento almacenado a una base de datos en la que no existe el usuario. En este caso es mejor seleccionar INVOKER que ejecuta el procedimiento almacenado con los permisos del usuario que llama a la función.

Respecto al **Acceso de datos SQL**, CONTAINS SQL, es necesario que la rutina contenga consultas SQL. READS SQL DATA permite indicar que la rutina no realiza ningún acceso en modo escritura, sino solo en modo lectura. MODIFIES SQL DATA permite indicar que la rutina no tiene consultas SQL. Estos parámetros se usan para mejorar el rendimiento.

Para ejecutar este procedimiento almacenado:

```
CALL creacion_persona();
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:

Seleccione la tabla Persona y haga clic en **Examinar**:

	+ Opciones						
	←	→					
	Editar	Copiar	Borrar	id_persona	Nombre	Apellidos	Edad
				1	Nanie	Morales HonHon	55
				2	David	Manrique Adán	48
				5	María	Malasaña Agora	36
				6	Roberto	Magalán	63
				7	David	Olís De Las Heras	33
				8	Rodolfo	Germán	64
				9	Rula	Campos	42
				10	Martín	De Julián	57
				11	Manuel	Barberá	45
				12	Martín	Alonso	64

El procedimiento almacenado ha insertado Martín Alonso.

Puede pasar los argumentos al procedimiento almacenado. Estos argumentos pueden ser de tipo:

- IN: variable de entrada del procedimiento almacenado utilizado dentro de esta.
- OUT: variable de salida del procedimiento almacenado. Esta variable se actualiza al final del procedimiento almacenado y se devuelve a PHP.
- INOUT: variable de entrada-salida del procedimiento almacenado.

Para hacer este procedimiento más versátil, va a tomar como argumentos los apellidos, el nombre y la edad de la persona que va a insertar.

Modifique el procedimiento almacenado de esta manera:

```
DROP PROCEDURE creacion_persona //
CREATE PROCEDURE creacion_persona
(IN apellidos VARCHAR(20), IN Nombre VARCHAR(20), IN Edad INT)
BEGIN
INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
(nombre,apellidos,edad);
END
//
```

DROP PROCEDURE significa «eliminar el procedimiento», ya que debe eliminarse antes de volver a crearlo.

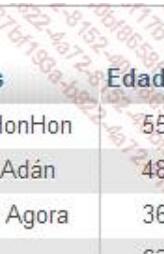
Copie este código en la pestaña **SQL** de PHPMyAdmin, cambie el delimitador a // en la zona **Delimitador**, a la izquierda del botón **Continuar** y pulse en el botón **Continuar**. Responda **Sí**, si recibe un mensaje para confirmar la eliminación del procedimiento almacenado.

Para ejecutar este procedimiento almacenado:

```
CALL creacion_persona('Emilio','Martin',15);
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**.

Seleccione la tabla Persona y haga clic en **Examinar**:



	+ Opciones	← T →	▼	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>		Editar		Copiar		Borrar	1
<input type="checkbox"/>		Editar		Copiar		Borrar	2
<input type="checkbox"/>		Editar		Copiar		Borrar	5
<input type="checkbox"/>		Editar		Copiar		Borrar	6
<input type="checkbox"/>		Editar		Copiar		Borrar	7
<input type="checkbox"/>		Editar		Copiar		Borrar	8
<input type="checkbox"/>		Editar		Copiar		Borrar	9
<input type="checkbox"/>		Editar		Copiar		Borrar	10
<input type="checkbox"/>		Editar		Copiar		Borrar	11
<input type="checkbox"/>		Editar		Copiar		Borrar	12
<input type="checkbox"/>		Editar		Copiar		Borrar	13

He aquí un ejemplo de procedimiento almacenado que devuelve el valor al cubo de un argumento:

```
CREATE PROCEDIMIENTO cubo(IN valor_entrada INT, OUT valor_salida BIGINT)
BEGIN
SELECT valor_entrada*valor_entrada*valor_entrada INTO valor_salida;
END //
```

La instrucción `SELECT... INTO variable` almacena el resultado de select en la variable.

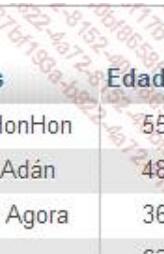
Para ejecutar este procedimiento almacenado:

```
CALL cubo(5,@RESULTADO);
SELECT @RESULTADO;
```

El valor de vuelta se ubica en la variable `@RESULTADO`. El nombre de esta variable tiene que estar obligatoriamente precedido por el símbolo `@`.

`SELECT @RESULTADO` permite mostrar el valor de la variable `@RESULTADO`.

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**:



+ Opciones
<b>@RESULTADO</b>
125

Los procedimientos almacenados se utilizan cuando se quiere ejecutar consultas SQL. En el ejemplo anterior, es más lógico crear una función:

```
CREATE FUNCTION funcion_cubo (valor_entrada INT) RETURNS BIGINT  
RETURN valor_entrada*valor_entrada*valor_entrada;
```

Las palabras clave CREATE FUNCTION significan que va a crear una función. A continuación escriba su nombre, seguido de los argumentos de entrada. Debe seleccionar el tipo de valor devuelto después de la palabra clave RETURNS. Para terminar, escriba el valor devuelto después de la palabra clave RETURN.

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**.

Para probar esta función:

```
SELECT funcion_cubo(5);
```

Copie esta consulta en la pestaña **SQL** de PHPMyAdmin y haga clic en **Continuar**.

Puede modificar, ejecutar, exportar o eliminar la función con ayuda de la misma pestaña que se utiliza en los procedimientos almacenados.



Existe un lenguaje que permite crear condiciones, bucles, variables y otros objetos en el procedimiento almacenado, pero no es el objetivo de este capítulo.

## 4. Otros objetos de MySQL

### a. Las tablas

Una vez que ha creado la tabla Persona en PHPMyAdmin, habrá observado que este código muestra:

```
CREATE TABLE '_prueba'.'Persona' (  
    'Id_person' INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    'Apellidos' VARCHAR(20) NOT NULL,  
    'Nombre' VARCHAR(20) NOT NULL,  
    'Edad' INT NOT NULL  
) ENGINE = MYISAM;
```

La palabra clave CREATE se utiliza para crear la tabla. Si quiere eliminarla, utilice la palabra clave DROP:

Para eliminar la tabla Persona:

```
DROP TABLE '_prueba'.'Persona';
```

Para modificar una tabla, puede utilizar la palabra clave ALTER, pero la sintaxis es mucho más compleja, ya que depende de lo que quiera hacer con los campos. Puede modificar, añadir o eliminar un campo, cambiar su tipo...

Tiene más información en el siguiente enlace: <http://dev.mysql.com/doc/refman/5.0/en/alter-table.html>

Las palabras clave siempre son CREATE, DROP y ALTER, sea para crear, eliminar, modificar una base, una tabla, un índice o cualquier otro objeto de la base de datos.

## b. Los índices

Los índices se utilizan para mejorar el rendimiento de una tabla. Cuando crea un índice en el campo de tipo identificador, la base de datos devuelve rápidamente el identificador entre todos los demás. Resulta imprescindible si tiene muchos registros, ya que la operación puede ser 100 veces más rápida.

Un índice se utiliza para:

- Encontrar rápidamente datos desde la cláusula WHERE.
- Leer registros en tablas con ayuda de las uniones.
- Ordenar o añadir datos.

Hemos visto en PHPMyAdmin cómo se crea un índice, pero también se puede crear en SQL. Su sintaxis es igual que en los otros objetos SQL, es decir, debe utilizar la palabra clave CREATE:

```
CREATE INDICE nombre_indice ON nombre_tabla (nombre_campo);
```

Este código crea un índice en el campo nombre\_campo de la tabla nombre\_tabla.

Hay varias opciones para que sea única, para crear un índice en varios campos al mismo tiempo, etc.

MyISAM representa el motor de almacenamiento de MySQL, es decir, la manera en que MySQL va a almacenar estos datos. Hay dos motores de almacenamiento:

- MyISAM: permite los índices en los campos de tipo fulltext y es muy rápido en las consultas de tipo SELECT o INSERT.
- InnoDB: soporta las transacciones y las claves extranjeras.

Una transacción permite asegurar que una secuencia de instrucción SQL se ha realizado correctamente. Si surge un problema tras una consulta, la transacción anula todas las demás consultas.

Tiene más información en el siguiente enlace: <http://dev.mysql.com/doc/refman/5.0/en/create-index.html>

## c. Las vistas

Una vista es una consulta SQL almacenada en el servidor que contiene una consulta de tipo SELECT y que se utiliza como una tabla.

Por ejemplo, para crear una vista llamada `vista_ejemplo` que muestra las cuatro primeras letras del nombre y los apellidos de las personas menores de 35 años y los apellidos y el nombre de las personas mayores de 35 años:

```
CREATE VIEW vista_ejemplo AS
SELECT SUBSTR(Nombre,1,4) as Raiz_nombre, Apellidos FROM Persona
WHERE Edad < 35
UNION
Select Nombre,Apellidos as Raiz_nombre, Apellidos FROM Persona
WHERE Edad > 35;
```

En PHPMyAdmin, tiene la nueva tabla `vista_ejemplo`. Haga clic en ella para mostrar sus datos:

+ Opciones	
Raíz_nombre	Apellidos
Davi	Olis De Las Heras
Emil	Martn
Nani	Morales HonHon
Davi	Manrique Adán
Mari	Malasaña Agora
Robe	Magalán
Rodo	Germán
Rula	Campos
Mart	De Julián
Manu	Barberá
Mart	Alonso

La sentencia `SELECT` contiene una cláusula `UNION`. El conjunto final es las personas con edad <35 años y >35 años.

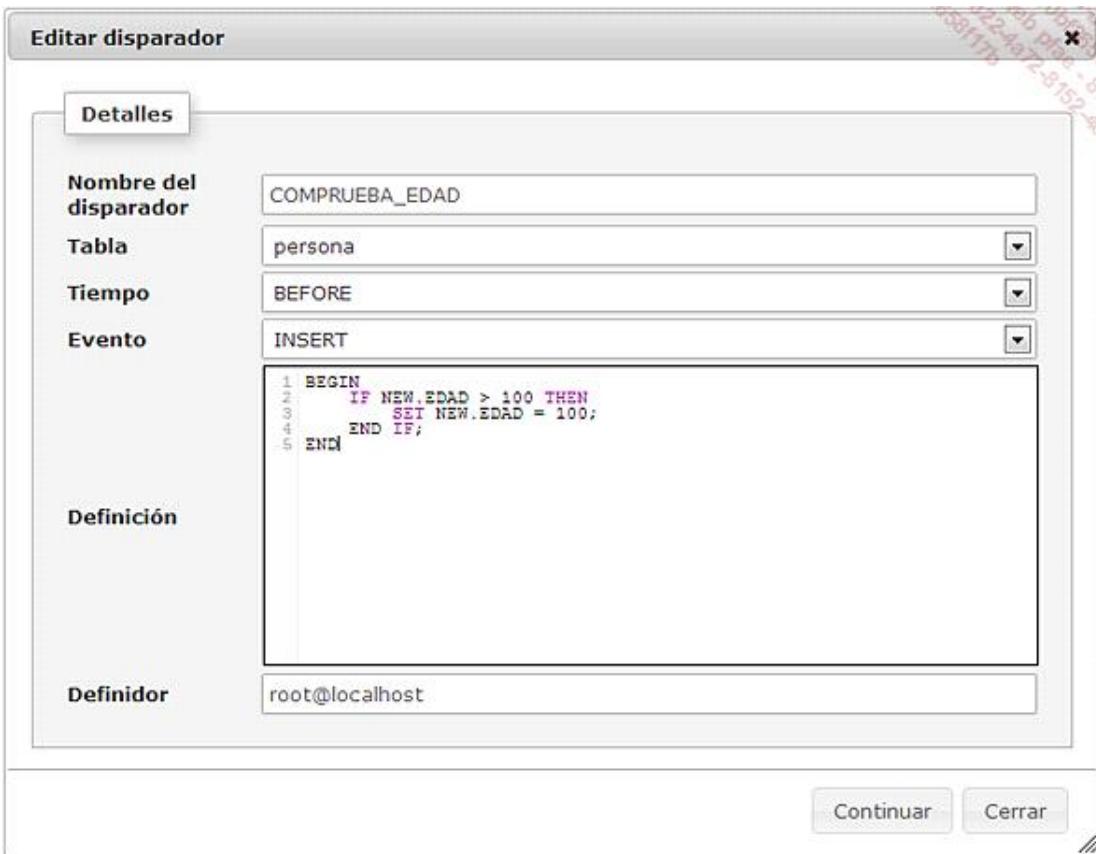
Puede utilizar esta vista como cualquier otra tabla.

#### d. Trigger

Un trigger es un objeto asociado a una tabla que se activa cuando se producen determinados eventos. Por ejemplo se puede comprobar el contenido de una tabla antes de realizar una inserción o eliminación.

Sólo se puede activar antes (before) o después (after) de un update, insert o delete.

Por ejemplo, para crear un trigger que se llama `COMPRUEBA_EDAD`, que establece la edad máxima en 100 en el supuesto de que fuera superior a 100, si se crea una nueva persona:



En lenguaje SQL:

```
CREATE TRIGGER `COMPROBAR_EDAD` BEFORE INSERT ON `Persona`
FOR EACH ROW BEGIN
    IF NEW.edad > 100 THEN
        SET NEW.edad = 100;
    END IF;
END
```

Si inserta una persona con una edad superior a 100, se le asignará 100 a su edad.

 Siempre es posible modificar la edad y asignar un valor superior a 100 porque el trigger está asociado al evento INSERT y no al evento UPDATE. Además no es posible tener dos triggers asociados al mismo evento y la misma acción sobre la misma tabla.

Encontrará más información en el enlace: <http://dev.mysql.com/doc/refman/5.0/en/triggers.html>

## Ejercicios SQL

Para realizar estos ejercicios, la tabla Persona y la tabla Idiomas se tienen que crear en la base de datos \_prueba. Estas tablas contienen los siguientes datos:

Tabla Persona:

	+ Opciones	← T →	id_persona	Nombre	Apellidos	Edad	Id_idioma
<input type="checkbox"/>	Editar  Copiar  Borrar		1	Nanie	Morales HonHon	55	1
<input type="checkbox"/>	Editar  Copiar  Borrar		2	David	Manrique Adán	60	2
<input type="checkbox"/>	Editar  Copiar  Borrar		3	María	Malasaña Agora	35	3
<input type="checkbox"/>	Editar  Copiar  Borrar		4	Roberto	Magalán	23	1
<input type="checkbox"/>	Editar  Copiar  Borrar		5	Manuel	Olís De Las Heras	20	2
<input type="checkbox"/>	Editar  Copiar  Borrar		6	Margarita	Germán	26	1

Tabla Idiomas:

	+ Opciones	← T →	Id	Etiqueta
<input type="checkbox"/>	Editar  Copiar  Borrar		1	Francés
<input type="checkbox"/>	Editar  Copiar  Borrar		2	Inglés
<input type="checkbox"/>	Editar  Copiar  Borrar		3	Alemán
<input type="checkbox"/>	Editar  Copiar  Borrar		4	Ruso

El script SQL que permite crear las tablas con sus datos es:

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

-- Base de datos: `_prueba`


-- Estructura de la tabla `idioma`


DROP TABLE IF EXISTS Idiomas;
CREATE TABLE IF NOT EXISTS Idiomas (
    Id int(11) NOT NULL AUTO_INCREMENT,
    Etiqueta varchar(20) NOT NULL,
    PRIMARY KEY (Id)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

-- Contenido de la tabla `idioma`


INSERT INTO Idiomas (Id, Etiqueta) VALUES
(1, 'Francés'),
(2, 'Inglés'),
(3, 'Alemán'),
```

```

(4, 'Ruso');

-- -----
-- Estructura de la tabla `persona`


DROP TABLE IF EXISTS Persona;
CREATE TABLE IF NOT EXISTS Persona (
    Id_person int(11) NOT NULL AUTO_INCREMENT,
    Nombre varchar(20) NOT NULL,
    Apellidos varchar(20) NOT NULL,
    Edad int(11) NOT NULL,
    Id_idioma int(11) NOT NULL,
    PRIMARY KEY (Id_person)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

-- Contenido de la tabla `persona`


INSERT INTO Persona (Id_person, Nombre, Apellidos, Edad, Id_idioma) VALUES
(1, 'Nanie', 'Morales HonHon', 55, 1),
(2, 'David', 'Manrique Adán', 60, 2),
(3, 'María', 'Malasaña Agora', 35, 3),
(4, 'Roberto', 'Magalán', 23, 1),
(5, 'Manuel', 'Olis De Las Heras', 20, 2),
(6, 'Margarita', 'Germán', 26, 1)

```

A continuación escriba el script SQL en la pestaña **SQL** de PHPMyAdmin. Este script elimina las tablas Idiomas y Persona y las vuelve a crear con sus datos.

## 1. Enunciados

### **Ejercicio 1 (muy fácil)**

Cree una consulta que permita mostrar los apellidos, el nombre y la edad de todas las personas mayores de 50 años.

+ Opciones		
NOMBRE	APELLIDOS	EDAD
Nanie	Morales HonHon	55
David	Manrique Adán	60

### **Ejercicio 2 (fácil)**

Cree una consulta que permita mostrar los apellidos, el nombre y la edad de todas las personas que hablen francés y cuyo nombre comience por 'Ma'.

+ Opciones		
NOMBRE	APELLIDOS	EDAD
Margarita	Germán	26

### **Ejercicio 3 (fácil)**

Cree una consulta que permita mostrar los apellidos, el nombre y la edad de las tres primeras personas con Nombre en orden alfabético.

+ Opciones		
NOMBRE	APELLIDOS	EDAD
David	Manrique Adán	60
Manuel	Olís De Las Heras	20
Margarita	Germán	26

### **Ejercicio 4 (dificultad media)**

Cree una consulta que permita mostrar la edad media y el idioma de las personas agrupadas por idioma.

+ Opciones	
Edad_Media	Idioma
34.6667	Francés
40.0000	Inglés
35.0000	Alemán

### **Ejercicio 5 (dificultad media)**

Cree una consulta que permita mostrar la edad media redondeada a la unidad de las personas agrupadas por idioma que tienen la cadena de caracteres 'es' contenida en su idioma.

+ Opciones	
Edad_media_redondeada	Idioma
35	Francés
40	Inglés

### **Ejercicio 6 (difícil)**

Cree una consulta que permita mostrar las tres primeras letras del nombre concatenadas con un espacio concatenado a su vez con las tres primeras letras de los apellidos de las personas mayores de 30 años. Y además, las tres últimas letras del nombre concatenadas con un espacio concatenado a su vez con las tres últimas letras de los apellidos de las personas menores de 30 años.

+ Opciones	
<b>Nombre_Apellidos</b>	
Nan Mor	
Dav Man	
Mar Mal	
rto lán	
uel ras	
ita mán	

### **Ejercicio 7 (dificultad media)**

Cree una consulta que permita mostrar la hora actual si el nombre contiene seis caracteres y la hora actual más una hora en otros casos.

+ Opciones	
<b>NOMBRE Horario</b>	
Nanie	13:55:19
David	13:55:19
María	13:55:19
Roberto	13:55:19
Manuel	12:55:19
Margarita	13:55:19

### **Ejercicio 8 (difícil)**

Cree una consulta que permita mostrar todos los idiomas que no habla David Manrique Adán.

+ Opciones	
<b>ETIQUETA</b>	
Francés	
Alemán	
Ruso	

### **Ejercicio 9 (muy difícil)**

Cree una consulta que permita mostrar la suma de edades agrupadas por idioma de todas las personas que hablan los idiomas que no hablan las personas de 18 a 25 años.

+ Opciones	
<b>Suma_Edad Idioma</b>	
80	Ingles
35	Alemán

## 2. Soluciones

### Solución del ejercicio 1

```
Select Nombre, Apellidos, Edad FROM Persona WHERE Edad > 50
```

### Solución del ejercicio 2

```
Select Nombre, Apellidos, Edad FROM Persona INNER JOIN Idiomas ON  
Persona.Id_idioma = Idiomas.Id WHERE Etiqueta = 'Francés'  
AND Nombre LIKE 'Ma%'
```

### Solución del ejercicio 3

```
Select Nombre, Apellidos, Edad FROM Persona ORDER BY Nombre LIMIT 3
```

### Solución del ejercicio 4

```
SELECT AVG(Edad) AS edad_media, Etiqueta AS Idiomas FROM Persona INNER JOIN  
Idiomas ON Persona.Id_idioma = Idiomas.Id GROUP BY Id_idioma
```

### Solución del ejercicio 5

```
SELECT ROUND(AVG(Edad)) AS edad_media_redondeada, Etiqueta AS Idiomas  
FROM Persona  
INNER JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id  
GROUP BY Id_idioma  
HAVING Etiqueta LIKE '%es%'
```

### Solución del ejercicio 6

```
SELECT CONCAT(SUBSTR(Nombre,1,3), ' ', SUBSTR(Apellidos,1,3))  
as Nombre_apellido  
FROM Persona WHERE Edad > 30  
UNION  
SELECT CONCAT(SUBSTR(Nombre,LENGTH(Nombre)-2), ' ', SUBSTR(Apellidos,  
LENGTH(Apellidos)-2)) as Nombre_apellido FROM Persona WHERE Edad < 30
```

### Solución del ejercicio 7

```
Select Nombre, IF(LENGTH(Nombre) = 6 ,CURRENT_TIME,ADDTIME(CURRENT_TIME,  
'01:00:00')) as horario FROM `Persona`
```

### **Solución del ejercicio 8**

```
SELECT Etiqueta FROM Idiomas WHERE Id NOT IN (  
SELECT Id_idioma FROM Persona WHERE Nombre='David'  
)
```

### **Solución del ejercicio 9**

```
SELECT SUM(Edad) as Suma_Edad, Etiqueta as Idiomas  
FROM Persona  
INNER JOIN Idiomas ON Persona.Id_idioma = Idiomas.Id  
GROUP BY Id_idioma  
HAVING Id_idioma NOT IN (  
SELECT Id_idioma FROM Persona WHERE Edad BETWEEN 25 AND 27  
)
```

# Acceso a las bases de datos con PHP

## 1. Introducción

Hemos visto anteriormente las diferentes consultas SQL; vamos a ver ahora cómo se ejecutan en PHP y a mostrar el resultado.

En esta parte del capítulo aprenderá a leer, insertar, modificar y eliminar datos desde formularios PHP.

Partiremos de la siguiente estructura de la tabla Persona.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	id_persona	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar  Eliminar  Primaria  Único  Índice  Espacial  Más
2	Nombre	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice  Espacial  Más
3	Apellidos	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice  Espacial  Más
4	Edad	int(11)			No	Ninguna		Cambiar  Eliminar  Primaria  Único  Índice  Espacial  Más

Y también los siguientes datos:

+ Opciones			
	← T →	▼	
			id_persona
	Editar  Copiar  Borrar		1
	Editar  Copiar  Borrar		2
	Editar  Copiar  Borrar		3
	Editar  Copiar  Borrar		4
	Editar  Copiar  Borrar		5
	Editar  Copiar  Borrar		6

Hay dos extensiones que utilizan las funciones para acceder a MySQL. Son `mysql_` y `mysqli_`. Estas dos extensiones son muy similares, pero `mysqli_` es más reciente (versión 5 de PHP) y tiene algunas funcionalidades complementarias. Por lo tanto, utilizaremos `mysqli_` en lo que queda de capítulo. En el próximo capítulo, presentaremos la extensión PDO (*PHP Data Object*). Es una extensión aún más reciente y completa a la hora de utilizar objetos.

## 2. Conexión

La función que permite conectarse a MySQL es `mysqli_connect()`.

Esta función toma como argumentos:

- El host: cadena de caracteres que contiene el nombre o la dirección IP del host, que corresponde a "localhost" o 127.0.0.1 si trabaja en modo local.
- El usuario: cadena de caracteres que contiene el nombre de usuario para conectarse a la base de datos. Corresponde a "root" si trabaja en modo local. Tenga cuidado, ya que este usuario tiene todos los derechos sobre su base de datos.
- La contraseña: cadena de caracteres que contiene la contraseña asociada al usuario. Por defecto se encuentra vacía.
- El nombre de la base de datos: cadena de caracteres opcional que contiene el nombre de su base de datos.

- El puerto: número de puerto opcional para conectarse al servidor MySQL, que por defecto equivale a 3306.

Esta función devuelve falso en caso de error, o un objeto mysqli que contiene el identificador de conexión en caso de éxito.

Por ejemplo:

```
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "_prueba");
?>
```

Otro ejemplo que gestiona los errores es:

```
<?php

$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información sobre el servidor:' . mysqli_GET_host_info($base);
}
else {
    printf('Error %d : %s.<br />', mysqli_connect_errno(),
    mysqli_connect_error());
} //%d representa un decimal que es el número de error que envía mysql
//connect errno. %s representa una cadena de caracteres, que es la etiqueta
//del error que envía mysql connect error

?>
```

La función mysqli\_GET\_host\_info(\$base) devuelve información de su servidor.

La función mysqli\_connect\_errno() devuelve el número del error en caso de un error de conexión.

La función mysqli\_connect\_error() devuelve el mensaje de error en caso de un error de conexión.

También se puede conectar al servidor y a la base de datos con ayuda de la función mysqli\_select\_db(). Esta función toma como argumento el objeto devuelto por la función mysqli\_connect().

Por ejemplo:

```
<?php
$base = mysqli_connect('127.0.0.1', 'root', '');
mysqli_select_db($base, '_prueba') ;
?>
```

### 3. Desconexión

La función que permite desconectarse de MySQL es mysqli\_close().

Por ejemplo:

```
if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}
```

No está obligado a cerrar la conexión, ya que PHP lo hace automáticamente al final del script. No obstante, le recomendamos que la cierre, ya que aumentará la velocidad de sus scripts.

## 4. Consultas no preparadas

### a. Leer datos

La función que permite ejecutar una consulta SQL es: `mysqli_query()`.

Esta función toma como argumentos:

- El objeto de la conexión: objeto que reenvía la función `mysqli_connect()`.
- La consulta: cadena de caracteres que contiene la consulta SQL.

Esta función devuelve `false` en caso de fallo y `true` o un identificador de sesión en caso de éxito.

La función que permite conocer el número de registros en el resultado de la consulta es: `mysqli_num_rows()`.

Esta función toma como argumento el resultado que ha devuelto la función `mysqli_query()` y devuelve el número de registros enviado por la consulta SQL.

Por ejemplo, para obtener el número de registros en la tabla Persona:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con mysqli</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:' . mysqli_get_host_info($base).
'<br />';
    // Ejecución de la consulta
```

```

$resultado = mysqli_query($base, 'SELECT * FROM Persona');
if ($resultado == FALSE) {
    echo "Error en la ejecución de la consulta.<br />";
}
else {
    // Examinar el número de registros
    echo 'Número de personas: '.mysqli_num_rows($resultado). '<br />';
}

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}

else {
    printf('Error %d : %s.<br />', mysqli_connect_errno(),
    mysqli_connect_error());
}

?>
</body>
</html>

```

Da como resultado:

Conexión realizada.  
 Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP  
 Número de personas: 6  
 Desconexión realizada.

 En adelante, ya no escribimos el código HTML del inicio de la página; solo indicamos el código PHP.

Si ahora quiere mostrar los datos de la tabla Persona, debe utilizar fetch, que permite leer el registro actual y desplazarse al siguiente registro.

Hay varias funciones que permiten utilizar fetch.

La función más actual es `mysqli_fetch_assoc()`.

Esta función toma como argumento el resultado que la función `mysqli_query()` ha devuelto y devuelve una tabla asociativa que tenga como índice el nombre de la columna de la consulta SQL.

Por ejemplo, para mostrar los apellidos y el nombre de la tabla Persona:

```

<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {

```

```

echo 'Conexión realizada.<br />';
echo 'Información del servidor:' . mysqli_get_host_info($base).
'<br />';
// Ejecución de la consulta
$resultado = mysqli_query($base, 'Select Nombre, Apellidos FROM Persona');
if ($resultado == FALSE) {
    echo "Error en la ejecución de la consulta.<br />";
}
else {
    //fetch en cada registro devuelto por la consulta
    while ($registro = mysqli_fetch_assoc($resultado)) {
        // Examinar apellidos y nombre de las personas
        echo "Apellidos:" . $registro['Apellidos'] . "
y nombre:" . $registro['Nombre'] . "<br />";
    }
}

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}
else {
    printf('Error %d : %s.<br /
>', mysqli_connect_errno(), mysqli_connect_error());
}
?>

```

Da como resultado:

```

Conexión realizada.
Información del servidor:MySQL host info: 127.0.0.1 a través de TCP/IP
Apellidos:Morales Honhon y nombre:Nanie
Apellidos:Manrique Adán y nombre:David
Apellidos:Malasaña Agora y nombre:María
Apellidos:Magalán y nombre:Roberto
Apellidos:Olís de las Heras y nombre:Manuel
Apellidos: Germán y nombre:Margarita
Desconexión realizada.

```

La variable \$registro es una tabla en la que los índices son los nombres de los campos que ha devuelto la consulta. Esta tabla se elimina con los nuevos valores de cada registro. Fetch permite pasar de registro en registro y, si no hay más registros, la condición del bucle while es falsa. Por lo tanto, el script sale del bucle.

Otra función que permite el fetch es mysqli\_fetch\_row( ).

Esta función toma como argumento el resultado que la función mysqli\_query( ) ha devuelto y devuelve una tabla indexada que tiene como índice un contador de 0 a n que representa las columnas en el orden establecido por la consulta.

Por ejemplo, para mostrar los apellidos y el nombre de la tabla Persona:

```

<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:'.mysqli_GET_host_info($base).
'<br />';
    // Ejecución de la consulta
    $resultado = mysqli_query($base, 'Select Nombre, Apellidos FROM Persona');
    if ($resultado == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
    }
    else {
        //fetch en cada registro devuelto por la consulta
        while ($registro = mysqli_fetch_row($resultado)) {
            // Examinar apellidos y nombre de las personas
            echo "Apellidos:".$registro[0]." y nombre:".$registro[1]."<br />";
        }
    }
}

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}
else {
    printf('Error %d : %s.<br />',mysqli_connect_errno(),
mysqli_connect_error());
}
?>

```

Da como resultado la misma información que antes.

Esta vez la tabla contiene el valor de los apellidos en `$registro[0]` y el valor del nombre en `$registro[1]`. Esta función es menos práctica, ya que el índice 0 corresponde a los apellidos y el índice 1 corresponde al nombre.

Otra función que permite el fetch es `mysqli_fetch_array()`. Toma como argumento complementario una constante que permite recuperar una tabla asociativa, una tabla indexada o las dos a la vez.

Las constantes son:

- `MYSQLI_ASSOC`: devuelve una tabla asociativa que equivale a `mysqli_fetch_assoc()`.
- `MYSQLI_NUM`: devuelve una tabla indexada que equivale a `mysqli_fetch_row()`.
- `MYSQLI_BOTH`: devuelve una tabla asociativa e indexada a la vez.

En el capítulo El objeto se explicarán estos conceptos.

Para terminar, la última función que permite el fetch es `mysqli_fetch_object()`.

Esta función toma como argumento el resultado que la función `mysqli_query()` ha enviado y devuelve un objeto con un atributo por cada campo devuelto por la consulta. Este atributo tiene como nombre y como valor respectivamente este y el del campo.

Por ejemplo, para mostrar los apellidos y el nombre de la tabla Persona:

```
<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:'.mysqli_GET_host_info($base).
'<br />';
    // Ejecución de la consulta
    $resultado = mysqli_query($base, 'Select Nombre, Apellidos FROM Persona');
    if ($resultado == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
    }
    else {
        //fetch en cada registro devuelto por la consulta
        while ($objeto = mysqli_fetch_object($resultado)) {
            // Examinar apellidos y nombre de las personas
            echo "Apellidos:".$objeto->Apellidos." y nombre:".$objeto->Nombre."<br />";
        }
    }
}

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}
else {
    printf('Error %d : %s.<br/ >',mysqli_connect_errno()
,mysqli_connect_error());
}
?>
```

Da como resultado:

Conexión realizada.  
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP  
Apellidos:Morales Honhon y nombre:Nanie  
Apellidos:Manrique Adán y nombre:David  
Apellidos:Malasaña Agora y nombre:María  
Apellidos:Magalán y nombre:Roberto  
Apellidos:Olís de las Heras y nombre:Manuel  
Apellidos: Germán y nombre:Margarita  
Desconexión realizada.

## b. Escribir datos

Para escribir datos, debe ejecutar una consulta de tipo INSERT.

Utilice en este caso la función `mysqli_query()`.

Esta función toma como argumentos:

- El objeto de conexión: objeto que la función `mysqli_connect()` ha devuelto.
- La consulta: cadena de caracteres que contiene la consulta SQL.

Por ejemplo, si quiere insertar una persona llamada Nadia González Pérez, de 31 años de edad:

```
<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:' . mysqli_GET_host_info($base) . '<br />';
    $sql = "INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
('Nadia', 'González Pérez', 31)";
    // Ejecución de la consulta
    $resultado = mysqli_query($base, $sql);
    if ($resultado == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
    }
    else {
        echo "Persona guardada.<br />";
    }

    if (mysqli_close($base)) {
        echo 'Desconexión realizada.<br />';
    }
    else {
        echo 'Error en la desconexión.';
    }
}
else {
    printf('Error %d :
%s.<br />', mysqli_connect_errno(), mysqli_connect_error());
}

?>
```

Ahora tiene un nuevo registro en la base de datos:

<input type="checkbox"/>	 Editar	 Copiar	 Borrar	7	Nadia	González Pérez	31
--------------------------	--	--	--	---	-------	----------------	----

Observe que el `id_person` no se ha añadido en la consulta. De hecho, es autoincremental; por lo tanto, la base de datos asignará un nuevo número al `id_person`.

Cuando inserta una nueva persona, no conoce su identificador. Para recuperar el último Id autoincremental que se ha añadido a la base de datos, debe utilizar la función `mysqli_insert_id()`. Esta función toma como argumento el objeto de conexión y devuelve el último identificador autoincremental que se ha añadido en la base de datos.

Por ejemplo, para insertar una persona llamada David Morales de 61 años de edad, añada después de `echo "Persona guardada.<br />"`; los siguientes registros:

```
$id = mysqli_insert_id($base);
echo "Su identificador es:".$id."<br />";
```

Da como resultado:

```
Conexión realizada.
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP
Persona guardada.
Su identificador de sesión es: 8.
Desconexión realizada.
```

### c. Eliminar datos

Para eliminar datos, ejecute una consulta de tipo `DELETE`.

Utilice en este caso la función `mysqli_query()`.

Esta función toma como argumentos:

- El objeto de conexión: objeto que la función `mysqli_connect()` ha devuelto.
- La consulta: cadena de caracteres que contiene la consulta SQL.

Por ejemplo, para eliminar una persona cuyo nombre es David:

```
<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:'.mysqli_GET_host_info($base).
'<br />';

    $sql = "DELETE FROM Persona WHERE Nombre = 'David'";
    // Ejecución de la consulta
    $resultado = mysqli_query($base, $sql);
    if ($resultado == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
    }
    else {
        echo "Persona eliminada.<br />";
    }
}
```

```

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}

else {
    printf('Error %d : %s.<br/>', mysqli_connect_errno(),
    mysqli_connect_error());
}
?>

```

Da como resultado:

Conexión realizada.  
 Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP  
 Persona eliminada.  
 Desconexión realizada.

 Aunque no exista la persona y aparezca el mensaje <<Persona eliminada>>, este mensaje se muestra cuando no hay error en la consulta. Si quiere saber el número de registros, debe utilizar la función **mysql\_affected\_rows** () .

#### d. Actualizar datos

Para modificar datos, ejecute una consulta de tipo UPDATE.

Utilice en este caso la función **mysqli\_query** () .

Esta función toma como argumentos:

- El objeto de conexión: objeto que la función **mysqli\_connect** () ha devuelto.
- La consulta: cadena de caracteres que contiene la consulta SQL.
- También puede utilizar la función **mysqli\_affected\_rows** (), que toma como argumento el resultado que ha devuelto la función **mysqli\_connect** () y devuelve el número de registros que la consulta ha modificado.

Por ejemplo, para modificar la persona cuyo nombre es David cambiando su apellido por MORALES y su edad de 61 a 62 años:

```

<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:' . mysqli_get_host_info($base).
    '<br />';

    $sql = "UPDATE Persona SET Apellidos = 'MORALES', Edad = 62 WHERE

```

```

Nombre = 'David';
// Ejecución de la consulta
$resultado = mysqli_query($base, $sql);
if ($resultado == FALSE) {
    echo "Error en la ejecución de la consulta.<br />";
}
else {
    $apellido_persona = mysqli_affected_rows($base);
    echo "Apellidos de personas modificadas:". $apellido_persona . "<br />";
}

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}
else {
    printf('Error %d : %s.<br />', mysqli_connect_errno(),
    mysqli_connect_error());
}
?>

```

Da como resultado:

Conexión realizada.  
 Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP  
 Número de personas modificadas:1.  
 Desconexión realizada.

## 5. Consultas preparadas

### a. Introducción

Una consulta preparada se utiliza mucho, ya que tiene varias ventajas. En primer lugar, evita la inyección SQL, ya que no se pueden ejecutar los datos de la consulta. En segundo lugar, si quiere ejecutar varias veces seguidas una consulta de tipo INSERT con distintos valores, no necesita reconstruir la consulta cada vez. Simplemente debe unir los nuevos valores y ejecutarla de nuevo. Por lo tanto, una consulta preparada es más segura y a veces más rápida que una consulta no preparada.

Debe tener estos datos:

+ Opciones							
		← T →	▼	id_persona	Nombre	Apellidos	Edad
<input type="checkbox"/>	Editar	Copiar	Borrar	1	Nanie	Morales HonHon	55
<input type="checkbox"/>	Editar	Copiar	Borrar	2	David	Manrique Adán	60
<input type="checkbox"/>	Editar	Copiar	Borrar	3	María	Malasaña Agora	35
<input type="checkbox"/>	Editar	Copiar	Borrar	4	Roberto	Magalán	48
<input type="checkbox"/>	Editar	Copiar	Borrar	5	Manuel	Olís De Las Heras	20
<input type="checkbox"/>	Editar	Copiar	Borrar	6	Margarita	Germán	26
<input type="checkbox"/>	Editar	Copiar	Borrar	7	Nadia	González Pérez	31

## b. Leer datos

Antes de ejecutar la consulta, debe prepararla, es decir, escribir la consulta sustituyendo los valores por signos de interrogación, que a continuación se sustituyen por valores.

La función que permite preparar la consulta es `mysqli_prepare()`.

Esta función toma como argumentos:

- El objeto de conexión: objeto que la función `mysqli_connect()` ha devuelto.
- La consulta: cadena de caracteres que contiene la consulta SQL con los signos de interrogación.
- Esta función devuelve un objeto `mysqli_stmt`, que las funciones `mysqli_stmt_bind_param()` y `mysqli_stmt_bind_result()` utilizan.

La función `mysqli_stmt_bind_param()` permite unir las variables a una consulta SQL.

Esta función tiene tres argumentos:

- El objeto consulta: resultado que la función `mysqli_prepare()` ha devuelto.
- Los tipos: cadena de caracteres que contiene los tipos de datos que pasan como argumentos en el mismo orden. Los tipos son:
  - `s` en las cadenas de caracteres,
  - `i` en los números enteros,
  - `d` en los números decimales,
  - `b` en los blob.
- Las variables: lista de variables que están unidas a los argumentos de la consulta sql, separadas por comas.
- Esta función devuelve `false` en caso de error y `true` en caso contrario.

Para ejecutar la consulta, utilice la función `mysqli_stmt_execute()`. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve `false` en caso de error y `true` en caso contrario.

En una consulta de tipo SELECT puede almacenar los valores devueltos por la consulta SQL. Utilice la función `mysqli_stmt_bind_result()`, que va a asociar las columnas de un resultado a unas variables.

Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y las

variables que corresponden a cada columna. Devuelve true en caso de éxito y false en caso de fallo.

Debe ejecutar la consulta preparada con la función `mysqli_stmt_execute()`. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve false en caso de error y true en caso contrario.

Para leer los valores en un bucle, utilice la función `mysqli_stmt_fetch()`. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve false en caso de error y true en caso contrario.

Para terminar, debe cerrar una consulta preparada con la función `mysqli_stmt_close()`. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto. Devuelve true en caso de éxito y false en caso de fallo.

Por ejemplo, para mostrar los apellidos y el nombre de personas mayores de 35 años:

```
<?php  
// Conexión a la base de datos  
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");  
if ($base) {  
    echo 'Conexión realizada.<br />';  
    echo 'Información del servidor:'.mysqli_GET_host_info($base).  
'<br />;  
  
    $sql = "Select Nombre, Apellidos FROM Persona WHERE Edad > ?";  
    // Preparación de la consulta  
    $resultado = mysqli_prepare($base, $sql);  
    // Enlace de argumentos.  
    $ok = mysqli_stmt_bind_param($resultado, 'i',$Edad);  
    $Edad=35; // Mientras no ejecute la consulta,  
              // puede inicializar la variable  
              // Ejecución de la consulta.  
    $ok = mysqli_stmt_execute($resultado);  
  
    if ($ok == FALSE) {  
        echo "Error en la ejecución de la consulta.<br />";  
    }  
    else {  
        // Asociación de variables de resultado.  
        $ok = mysqli_stmt_bind_result($resultado,$Apellido,$Nombre);  
        // Lectura de valores.  
        echo "Apellidos y nombre de personas que tengan una edad > 35<br />";  
        while (mysqli_stmt_fetch($resultado)) {  
            echo $Apellido.", ".$Nombre."<br />";  
        }  
        mysqli_stmt_close($resultado);  
    }  
  
    if (mysqli_close($base)) {  
        echo 'Desconexión realizada.<br />';  
    }  
    else {  
        echo 'Error en la desconexión.';  
    }  
}
```

```

    }
else {
    printf('Error %d : %s.<br/ >',mysqli_connect_errno(),
mysqli_connect_error());
}
?>

```

Da como resultado:

```

Conexión realizada.
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP
Apellidos y nombre de personas que tengan una edad >35
Morales Honhon, Nanie
Manrique Adán, David
Magalán, Roberto
Desconexión realizada.

```

### c. Escribir datos

Para escribir datos, ejecute una consulta de tipo INSERT.

Debe preparar la consulta con `mysqli_prepare()`, unir las variables a la consulta con `mysqli_stmt_bind_param()` y ejecutar con `mysqli_stmt_execute()`. Estas tres funciones se explican en la sección anterior.

Por ejemplo, para insertar una persona llamada Mónica Prieto, de 63 años:

```

<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:' . mysqli_GET_host_info($base).
'<br />';

    $sql = "INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES (?,?,?)";
    // Preparación de la consulta
    $resultado = mysqli_prepare($base, $sql);
    // Enlace de argumentos.
    $ok = mysqli_stmt_bind_param($resultado, 'ssi', $apellido, $nombre, $edad);
    $apellido = 'Prieto';
    $nombre = 'Mónica';
    $edad = 63;
    // Ejecución de la consulta.
    $ok = mysqli_stmt_execute($resultado);

    if ($ok == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
    }
    else {
        echo "Persona añadida.<br />";
    }
}

```

```

mysqli_stmt_close($resultado);
if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}

else {
printf('Error %d : %s.<br/>',mysqli_connect_errno(),
mysqli_connect_error());
}
?>

```

Da como resultado:

```

Conexión realizada.
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP
Persona añadida.
Desconexión realizada.

```

#### d. Modificar datos

Para obtener el número de registros que la consulta de tipo UPDATE ha modificado, utilice la función `mysqli_stmt_affected_rows()`.

Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve -1 en caso de error y el número de registros actualizados en caso contrario.

Por ejemplo, para modificar la persona cuyo apellido es MORALES, cambiando su apellido de nuevo a Morales y su edad de 62 a 61 años:

```

<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:' . mysqli_get_host_info($base).
'<br />';

    $sql = "UPDATE Persona SET Apellidos = ?, Edad = ? WHERE Apellidos =
'MORALES'";
    // Preparación de la consulta
    $resultado = mysqli_prepare($base, $sql);
    // Enlace de argumentos.
    $ok = mysqli_stmt_bind_param($resultado, 'si', $apellido, $edad);
    $apellido = 'Morales';
    $edad = 61; //Mientras no ejecute la consulta,
               //puede inicializar las variables
    // Ejecución de la consulta.
    $ok = mysqli_stmt_execute($resultado);
}

```

```

if ($ok == FALSE) {
    echo "Error en la ejecución de la consulta.<br />";
}
else {
    echo "Número de personas modificadas:
".mysqli_stmt_affected_rows($resultado).".<br />";
}
mysqli_stmt_close($resultado);
if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}
else {
    printf('Error %d : %s.<br />', mysqli_connect_errno(),
mysqli_connect_error());
}
?>

```

Da como resultado:

```

Conexión realizada.
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP
Número de personas modificadas:1.
Desconexión realizada.

```

## e. Eliminar datos

De la misma manera que para escribir o modificar datos, vamos a utilizar las funciones `mysqli_prepare()`, `mysqli_stmt_bind_param()` y `mysqli_stmt_execute()` para eliminar datos.

Por ejemplo, para eliminar a David Morales:

```

<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:'.mysqli_GET_host_info($base).
'<br />';

$sql = "DELETE FROM Persona WHERE Apellidos = ?";
// Preparación de la consulta
$resultado = mysqli_prepare($base, $sql);
// Enlace de argumentos.
$ok = mysqli_stmt_bind_param($resultado, 's',$apellido);
$apellido = 'Morales'; //Mientras no ejecute la consulta,
                    //puede inicializar la variable

```

```

// Ejecución de la consulta.
$ok = mysqli_stmt_execute($resultado);

if ($ok == FALSE) {
    echo "Error en la ejecución de la consulta.<br />";
}
else {
    echo "Persona eliminada.<br />";
}
mysqli_stmt_close($resultado);

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}

else {
    printf('Error %d : %s.<br/>', mysqli_connect_errno(),
    mysqli_connect_error());
}
?>

```

Da como resultado:

```

Conexión realizada.
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP
Persona eliminada.
Desconexión realizada.

```

## f. Almacenar un resultado

Si quiere saber el número de registros seleccionados en una consulta de tipo SELECT, hay una función que permite almacenar el resultado y otra que obtiene el número de registros de este resultado.

La función `mysqli_stmt_store_result()` permite almacenar en la memoria el resultado de una consulta. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve false en caso de error y true en caso contrario.

La función `mysqli_stmt_num_rows()` permite devolver el número de registros de una consulta preparada. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve 0 en caso de error y el número de registros de la consulta en caso contrario.

La función `mysqli_stmt_free_result()` permite liberar memoria con el resultado que se ha obtenido con `mysqli_stmt_store_result()`. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y no devuelve nada.

Por ejemplo, para mostrar el número de personas que son mayores de 35 años:

```
<?php
```

```

// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:' . mysqli_get_host_info($base);
    '<br />';

    $sql = "Select Nombre, Apellidos FROM Persona WHERE Edad > ?";
    // Preparación de la consulta
    $resultado = mysqli_prepare($base, $sql);
    // Enlace de argumentos.
    $ok = mysqli_stmt_bind_param($resultado, 'i', $Edad);
    $Edad = 35; //mientras no ejecute la consulta,
                //puede inicializar la variable
    // Ejecución de la consulta.
    $ok = mysqli_stmt_execute($resultado);

    if ($ok == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
    }
    else {
        // Asociación de variables de resultado.
        $ok = mysqli_stmt_bind_result($resultado, $Apellido, $Nombre);
        // Almacenamiento de valores.
        $ok = mysqli_stmt_store_result($resultado);
        echo "Número de personas que tengan una edad > 35:
        ".mysqli_stmt_num_rows($resultado). "<br />";
        //Liberación del resultado
        mysqli_stmt_free_result($resultado);
        mysqli_stmt_close($resultado);
    }
}

if (mysqli_close($base)) {
    echo 'Desconexión realizada.<br />';
}
else {
    echo 'Error en la desconexión.';
}

}
else {
    printf('Error %d : %s.<br />', mysqli_connect_errno(),
    mysqli_connect_error());
}
?>

```

Da como resultado:

Conexión realizada.  
 Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP  
 Número de personas que tengan una edad > 35: 2  
 Desconexión realizada.

Este método consume memoria; por lo tanto, debe tener cuidado con lo que almacena en ella.

## **g. Examinar los errores de una consulta preparada**

Las funciones que permiten recuperar los errores de una consulta preparada son `mysqli_stmt_errno()` y `mysqli_stmt_error()`.

La función `mysqli_stmt_errno()` permite devolver el número de error. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto y devuelve 0 si no hay error.

La función `mysqli_stmt_error()` permite devolver el mensaje de error. Esta función toma como argumento el objeto consulta que la función `mysqli_prepare()` ha devuelto.

Estas funciones muestran los errores una vez que se ejecuta la consulta. Para ver los errores de preparación, utilice las funciones `mysqli_errno()` y `mysqli_error()`.

Por ejemplo, si añade una persona con un identificador que ya existe:

```
<?php
// Conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");
if ($base) {
    echo 'Conexión realizada.<br />';
    echo 'Información del servidor:'.mysqli_GET_host_info($base).
'<br />';

    $sql = "INSERT INTO Persona (Id_person, nombre, edad) VALUES (?,?,?)";
    // Preparación de la consulta
    $resultado = mysqli_prepare($base, $sql);
    // Enlace de argumentos.
    $ok = mysqli_stmt_bind_param($resultado, 'isi',$id,$nombre,$edad);
    $id = 1; //Añadir un identificador que ya existe
    $nombre = 'Mónica';
    $edad = 63; //Mientras no ejecute la consulta,
                //puede inicializar las variables
    // Ejecución de la consulta.
    $ok = mysqli_stmt_execute($resultado);

    if ($ok == FALSE) {
        echo "Error en la ejecución de la consulta.<br />";
        echo "error : ".mysqli_stmt_error($resultado).'-'.
'.mysqli_stmt_error($resultado)."<br />";
    }
    else {
        echo "Persona añadida.<br />";
    }
    mysqli_stmt_close($resultado);
    if (mysqli_close($base)) {
        echo 'Desconexión realizada.<br />';
    }
    else {
        echo 'Error en la desconexión.';
    }
}

}
```

```
else {
    printf('Error %d : %s.<br/>',mysqli_connect_errno(),
    mysqli_connect_error());
}
?>
```

Da como resultado:

Conexión realizada.  
Información del servidor:MySQL host info: 127.0.0.1 con TCP/IP  
Error en la ejecución de la consulta.  
error: 1062 - Duplicate entry '1' for key 'PRIMARY'  
Desconexión realizada.

# PDO

En esta parte, vamos a poner en práctica algunas nociones de PDO. Para la correcta compresión de su sintaxis, es preciso asimilar los conceptos que se explican en el capítulo El objeto.

## 1. Introducción

PDO (*PHP Data Object*) es una librería de funciones PHP que permite acceder a cualquier base de datos, ya que aporta una capa de abstracción al acceso a las bases de datos. Es un lenguaje orientado a objetos, aunque el funcionamiento no es muy distinto al de mysqli. Siempre se debe escribir la consulta y ejecutarla.

Para activar esta librería, abra el archivo PHP.ini, al que se accede con el menú Configuration, PHP. A continuación, compruebe que no hay ningún punto y coma delante del registro:

```
;extension=php_openssl.dll  
;extension=php_pdo_firebird.dll  
;extension=php_pdo_mssql.dll  
extension=php_pdo_mysql.dll  
;extension=php_pdo_oci.dll  
;extension=php_pdo_odbc.dll
```

Esta librería se escribe en lenguaje objeto; por lo tanto, la sintaxis puede ser confusa. En el capítulo El objeto se explica más detalladamente la programación orientada a objetos.

Como requisito, es preciso haber creado la tabla Persona.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	id_persona	int(11)			No	Ninguna	AUTO_INCREMENT	Primaria  Único  Índice  Espacial
2	Nombre	varchar(20)	latin1_swedish_ci		No	Ninguna		Primaria  Único  Índice  Espacial
3	Apellidos	varchar(20)	latin1_swedish_ci		No	Ninguna		Primaria  Único  Índice  Espacial
4	Edad	int(11)			No	Ninguna		Primaria  Único  Índice  Espacial

Y también los siguientes datos:

+ Opciones					
	← T →	▼	id_persona	Nombre	Apellidos
<input type="checkbox"/>			1	Nanie	Morales HonHon
<input type="checkbox"/>			2	David	Manrique Adán
<input type="checkbox"/>			3	María	Malasaña Agora
<input type="checkbox"/>			4	Roberto	Magalán
<input type="checkbox"/>			5	Manuel	Olís De Las Heras
<input type="checkbox"/>			6	Margarita	Germán

La sección PHPMyAdmin explica cómo debe crear la base con los datos.

## 2. Conexión

Para conectarse a base de datos MySQL, debe crear una instancia de la clase PDO, es decir, crear un objeto que es

un elemento de la clase PDO, pero con algunos argumentos. Este concepto se explica de manera más concreta en el capítulo de programación orientada a objetos.

El objeto que permite conectarse a MySQL es PDO( ).

Este objeto toma como argumentos:

- La cadena de conexión: cadena de caracteres que contiene el SGBD utilizado y el nombre o la dirección IP del *host*, que corresponde a "localhost" o 127.0.0.1 si trabaja en modo local. Esta cadena también contiene el nombre de la base de datos.
- El usuario: cadena de caracteres que contiene el nombre de usuario para conectarse a la base de datos. Si trabaja en modo local, corresponde a "root". Atención, este usuario tiene todos los derechos sobre su base de datos.
- La contraseña: cadena de caracteres que contiene la contraseña asociada al usuario. Por defecto está vacía.

Por ejemplo, para conectarse a la base de datos \_prueba:

```
<?php  
$base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');  
?>
```

La variable \$base es un objeto; por lo tanto, no puede mostrar su valor.

Para comprobar si el código ha generado un error, debe escribir el código con la instrucción try {} catch (Exception \$e) {}.

Por ejemplo:

```
<?php  
try {  
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');  
}  
catch (Exception $e) {  
    die('Error : ' . $e->getMessage());  
}  
?>
```

Si en el bloque try ocurre un error, PHP pasa automáticamente al bloque catch y por lo tanto ejecuta la instrucción die(). La función die() equivale a la función exit(), es decir, termina el script actual mostrando un mensaje.

Para recuperar los errores que se producen, debe añadir el siguiente código:

```
$base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Este código va a activar las excepciones PDO.

Desde la versión 5.5 de PHP, es posible usar el bloque finally. Es bloque se ubica después del catch y siempre se ejecuta.

Por ejemplo:

```

<?php
try {
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Conexión ok.";
}
catch (Exception $e) {
    die('Error : ' . $e->getMessage());
}
finally {

    $base = null; //cierre de la conexión
}

?>

```

Da como resultado:

Conexión ok.

En todos los casos, el código pasa al bloc finally y cierra la conexión.

### 3. Consultas no preparadas

#### a. Leer datos

El método que permite ejecutar una consulta SQL es: `query()`.

Este método forma parte del objeto conexión que ha devuelto con `new PDO()`. Toma como argumento la consulta SQL en forma de cadena de caracteres.

Esta función devuelve un objeto resultado que contiene todo lo que la consulta SQL vuelve a enviar.

El método que permite conocer el número de registros en el resultado de la consulta es: `rowCount()`.

Este método forma parte del objeto `$resultado` que el método `query()` ha devuelto.

Por ejemplo, para obtener el número de registros en la tabla Persona:

```

<?php
try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Recuperación datos de la tabla Persona
    $resultado = $base->query('SELECT * FROM Persona');

    echo "Número de personas:". $resultado->rowCount();
}

```

```

    // Muestra el número de registros

    $resultado->closeCursor(); // Cierre de la consulta
}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}
?>

```

Da como resultado:

Número de personas:6

Si ahora quiere mostrar los datos de la tabla Persona, debe utilizar fetch, que permite leer el registro actual y desplazarse al siguiente registro.

El método que permite realizar el fetch de la consulta es: `fetch()`.

Este método forma parte del objeto `$resultado` que el método `query()` ha devuelto.

Por ejemplo, para mostrar los apellidos y los nombres contenidos en la tabla Persona:

```

<?php
try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Recuperación de datos de la tabla Persona
    $resultado = $base->query('SELECT Nombre, Apellidos FROM Persona');

    echo "Número de personas:". $resultado->rowCount();
    // Muestra el número de registros
    while ($datos = $resultado->fetch())
    {
        ?>
        <p>
        Apellidos : <?php echo $datos['Apellidos']; ?>,
        Nombre : <?php echo $datos['Nombre']; ?>
        </p>
        <?php
    }
    $resultado->closeCursor(); // Cierre de la consulta
}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}
?>

```

Da como resultado:

Número de personas:6  
Apellidos:Gómez del Pozo y nombre:Carolina  
Apellidos:Morales HonHon y nombre:Estefanía  
Apellidos:Morales y nombre:Luna  
Apellidos:Del Morán y nombre:Luis  
Apellidos:López y nombre:María  
Apellidos:Prieto y nombre:Mónica

## b. Escribir datos

Para escribir datos, debe ejecutar una consulta de tipo INSERT.

El método que permite ejecutar una consulta SQL de tipo UPDATE, INSERT o DELETE es: exec( ).

Este método forma parte del objeto conexión que se ha devuelto por new PDO(). Toma como argumento la consulta SQL en forma de cadena de caracteres.

Por ejemplo, para insertar una persona llamada Oliver Durán, de 36 años de edad:

```
<?php

try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

    $sql = "INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
('Oliver','Durán',36)";
    // Añadir datos en la tabla Persona
    $base->exec($sql);
    echo "Persona añadida.";
}

catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}

?>
```

Entonces obtiene un nuevo registro en la base de datos:

<input type="checkbox"/>	 Editar	 Copiar	 Borrar	7   Oliver   Durán   36
--------------------------	--	--	--	-------------------------

Observe que el id\_person no se ha añadido en la consulta. Como es autoincremental, la base de datos asignará

un nuevo número al id\_person.

Por lo tanto, cuando inserta una nueva persona, no conoce su identificador. Para recuperar el último Id autoincremental que se ha añadido en la base de datos, debe utilizar el método `lastInsertId()`. Este método forma parte del objeto conexión que se ha devuelto por `new PDO()` y devuelve el último entero autoincremental que se ha añadido en la base de datos.

Por ejemplo, para insertar una persona llamada Gerardo Roldán, de 64 años de edad:

```
<?php
try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
('Gerardo','Roldán',64)";
    // Añadir de datos en la tabla Persona
    $base->exec($sql);
    echo "El identificador de la última persona añadida es:";
    echo $base->lastInsertId()."";
}
catch(Excepción $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}

?>
```

Da como resultado:

El identificador de la última persona añadida es:8.

<input type="checkbox"/>		Editar		Copiar		Borrar	8	Gerardo	Roldán	64
--------------------------	--	--------	--	--------	--	--------	---	---------	--------	----

### c. Eliminar datos

Para eliminar datos, debe ejecutar una consulta de tipo DELETE.

El método que permite ejecutar una consulta SQL de tipo UPDATE, INSERT o DELETE es: `exec()`.

Este método forma parte del objeto conexión que se ha devuelto por `new PDO()`. Toma como argumento la consulta SQL en forma de cadena de caracteres.

Por ejemplo, para eliminar una persona cuyo apellido es López:

```
<?php
try
```

```

{
$base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
$base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

$sql = "DELETE FROM Persona WHERE Apellidos = 'López'";
// Eliminación de datos en la tabla Persona
$base->exec($sql);
echo "Persona eliminada.";
}

catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}

?>

```

Da como resultado:

Persona eliminada.

#### **d. Actualizar datos**

Para modificar datos, ejecute una consulta de tipo UPDATE.

El método que permite ejecutar una consulta SQL de tipo UPDATE, INSERT o DELETE es: `exec()`.

Este método forma parte del objeto conexión que se ha devuelto por `new PDO()`. Toma como argumento la consulta SQL en forma de cadena de caracteres.

Para obtener el número de registros modificados en una consulta de tipo UPDATE, utilice el retorno del método `exec()`, que devuelve el número de registros afectados.

Por ejemplo, para modificar el apellido Prieto por Lucas y la edad de 36 a 33 años:

```

<?php

try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

    $sql = "UPDATE Persona SET Apellidos = 'Lucas', Edad = 33 WHERE
Apellidos = 'Prieto'";
    // Modificación de datos en la tabla Persona
    $numero = $base->exec($sql);
    echo "Número de personas modificadas:".$numero;
}
catch(Excepción $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}

```

```
}
```

```
?>
```

Da como resultado:

```
Número de personas modificadas:1
```

## 4. Consultas preparadas

### a. Leer datos

El método que permite preparar una consulta SQL de tipo SELECT, UPDATE, DELETE o INSERT es: `prepare()`.

Este método forma parte del objeto conexión que se ha devuelto por `new PDO()`. Toma como argumento una cadena de caracteres que contiene la consulta SQL con apellidos o marcadores que hay que unir a valores diferentes.

Esta función devuelve un objeto de tipo `PDOStatement` que contiene todo lo que vuelve a enviar la consulta SQL.

Por ejemplo, para mostrar los apellidos y los nombres incluidos en la tabla Persona con una edad > 5 y un apellido que comienza por 'Mo':

```
<?php

try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

    $sql = "Select Nombre, Apellidos FROM Persona WHERE Edad > ?
AND Apellidos LIKE ?";
    // Preparación de la consulta con los marcadores
    $resultado = $base->prepare($sql);
    $resultado->execute(array(5,'Mo%'));
    //execute toma como parámetro
    //una tabla que contiene los valores en el orden cuyo marcador es un ?
    while ($registro = $resultado->fetch())
    {
        echo 'Apellidos:'.$registro['Apellidos'].',
Nombre:'.$registro['Nombre'].'<br />';
    }

    $resultado->closeCursor();

}
catch(Excepción $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}
?>
```

Da como resultado:

Apellidos:Morales HonHon, Nombre:Nanie

Y el mismo ejemplo designando los marcadores:

```
<?php

try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "Select Nombre, Apellidos FROM Persona WHERE Edad > :Edad
AND Apellidos LIKE :apellido";
    // Preparación de la consulta con los marcadores
    $resultado = $base->prepare($sql);
    $resultado->execute(array(':Edad' => 5, ':apellido' => 'Mo%'));//execute
    //toma como parámetro una tabla que contiene como clave el número
    //de marcadores y sus valores correspondientes
    while ($registro = $resultado->fetch())
    {
        echo 'Apellidos:'.$registro['Apellidos'].',
Nombre:'.$registro['Nombre'].'<br />';
    }

    $resultado->closeCursor();
}

catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}
?>
```

Da como resultado:

Apellidos:Morales HonHon, Nombre:Nanie

Esta solución tiene la ventaja de ser un poco más legible en los argumentos que pasan a la consulta SQL.

## b. Escribir datos

Al igual que para leer datos, va a utilizar los métodos `prepare()` y `execute()`.

Por ejemplo, para insertar una persona llamada Clara Rincón López, de 42 años de edad:

```
<?php
```

```

try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
(:apellido, :nombre, :edad)";
    // Preparación de la consulta con los marcadores
    $resultado = $base->prepare($sql);
    $resultado->execute(array(':apellido' => 'Rincón López', ':nombre' =>
'Clara',
':edad' => 42));
    echo "El identificador de la última persona añadida es:" ;
    echo $base->lastInsertId()."";
    $resultado->closeCursor();

}
catch(Excepción $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}

?>

```

Da como resultado:

El identificador de la última persona añadida es:9.

Puede ver en PHPMyADmin lo siguiente:

<input type="checkbox"/>	 Editar	 Copiar	 Borrar	9	Clara	Rincón López	42
--------------------------	--	--	--	---	-------	--------------	----

Si quiere ejecutar una consulta de inserción varias veces seguidas, utilice el enlace de argumentos con el método bindParam().

Por ejemplo, para insertar una persona llamada Juan López Ruiz, de 57 años, y otra persona llamada Bob Martínez, de 45 años de edad:

```

<?php
try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "INSERT INTO Persona (Nombre, Apellidos, Edad) VALUES
(:apellido, :nombre, :edad)";
    // Preparación de la consulta con los marcadores
    $resultado = $base->prepare($sql);
    // Enlace de argumentos.

```

```

$resultado->bindParam(':apellido', $apellido);
$resultado->bindParam(':nombre', $nombre);
$resultado->bindParam(':edad', $edad);
//primera persona
$apellido = "López Ruiz ";
$nombre = " Juan";
$edad = 57;
$resultado->execute();
echo "El identificador de la última persona añadida es:" ;
echo $base->lastInsertId().".<br />";
//segunda persona
$nombre = "Bob";
$apellido = "Martínez";
$edad = 45;
$resultado->execute();
echo "El identificador de la última persona añadida es:" ;
echo $base->lastInsertId().".";

$resultado->closeCursor();

}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}

?>

```

Da como resultado:

El identificador de la última persona añadida es:10.

El identificador de la última persona añadida es:11.

<input type="checkbox"/>		Editar		Copiar		Borrar	10	Juán	López Ruiz	57
<input type="checkbox"/>		Editar		Copiar		Borrar	11	Bob	Martínez	45

### c. Eliminar datos

Al igual que para escribir y leer datos, va a utilizar los métodos `prepare()` y `execute()`.

Par ejemplo, para eliminar el registro Martínez:

```

<?php

try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}

```

```

$sql = "DELETE FROM Persona WHERE Apellidos =:apellido";
// Preparación de la consulta con los marcadores
$resultado = $base->prepare($sql);
$resultado->execute(array(':apellido' => 'Martínez'));
echo "Persona eliminada.";
$resultado->closeCursor();

}

catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}
?>

```

Da como resultado:

Persona eliminada.

#### d. Modificar datos

Al igual que para escribir y leer datos, va a utilizar los métodos `prepare()` y `execute()`.

Por ejemplo, para modificar la edad de Nanie Morales HonHon de 55 a 36 años:

```

<?php
try
{
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

    $sql = "UPDATE Persona SET Edad = :edad WHERE Apellidos = :apellido
AND Nombre = :nombre";
    // Preparación de la consulta con los marcadores
    $resultado = $base->prepare($sql);
    $resultado->execute(array(':edad' => 36, ':apellido' => 'Morales HonHon',
':nombre' => 'Nanie'));
    echo "Persona modificada.";
    $resultado->closeCursor();

}
catch(Excepción $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}

?>

```

Da como resultado:

Persona modificada.

## e. Llamar a un procedimiento almacenado

Un procedimiento almacenado tiene un nombre y puede tener argumentos de entrada y de salida.

Lo hemos explicado más detalladamente en este capítulo, sección SQL avanzado - Los procedimientos almacenados y funciones.

Para empezar, debe crear los procedimientos almacenados `creacion_persona` y `cubo` de la sección SQL avanzado - Los procedimientos almacenados y funciones. Compruebe que el campo `Id_idioma` no está en el procedimiento almacenado `creacion_persona`.

El primer procedimiento almacenado `creacion_persona` solo tiene argumentos de entrada. Este procedimiento utiliza la palabra clave `CALL` y solo debe unir sus argumentos a la función `bindParam()` y ejecutar la consulta.

Por ejemplo, para añadir la persona David MORALES, de 40 años de edad:

```
<?php
try
{
    // Conexión a la base de datos
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

    $stmt = $base->prepare("CALL creacion_persona(?, ?, ?)");
    $apellido = 'MORALES';
    $nombre = 'David';
    $edad = 40;
    $stmt->bindParam(1, $nombre, PDO::PARAM_STR, 20); //argumento de tipo
                                                       //STRING
    $stmt->bindParam(2, $apellido, PDO::PARAM_STR, 20);
    $stmt->bindParam(3, $edad, PDO::PARAM_INT); //argumento de tipo INTEGER

    // Ejecución del procedimiento almacenado
    $stmt->execute();
    echo "El procedimiento ha insertado una nueva persona.";
}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}
?>
```

Da como resultado:

El procedimiento ha insertado una nueva persona.

El segundo procedimiento almacenado `cubo` tiene un argumento de entrada y un argumento de salida.

Este procedimiento vuelve a enviar el valor en entrada al cubo. El problema es que hay un bug en PDO/MySQL que

impide recuperar el valor de retorno de manera estándar. Por lo tanto, debe ejecutar una segunda consulta para recuperar el valor de salida.

```
<?php
try
{
    // Conexión a la base de datos
    $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    //llamada al procedimiento almacenado y la consulta que recupera el
    //argumento de salida.
    $statement = $base->prepare('CALL cubo(:entrada, @salida); SELECT
@salida AS salida;');
    $entrada = 3;
    $statement->bindParam(':entrada', $entrada);
    //ejecución de la procedimiento almacenada
    $statement->execute();

    $statement->nextRowset();
    //lectura del argumento de salida
    $row = $statement->fetchObject();
    echo "El valor ".$entrada." Al cubo es:".$row->salida;

}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}

?>
```

Da como resultado:

El valor de 3 al cubo es:27

# Ejercicios

## 1. Enunciados

### **Ejercicio 1 (fácil): Página de conexión**

Cree una página login.php que contenga un área login y una zona password con un botón de tipo submit. Esta página llama a comprobar\_login.php, que comprueba si el login y la contraseña se encuentran en la base de datos. Retome el ejercicio 1 del capítulo Transmitir datos de una página a otra y modifique la página verif\_login.php para mostrar "login correcto" si la persona se halla en la base de datos o para que se redireccione a la página login.php en caso contrario. También debe crear una tabla inicio de sesion\_password que contenga el Id, el login y la contraseña.



### **Ejercicio 2 (medio): Formulario de inscripción**

Este ejercicio requiere buenos conocimientos en JavaScript o en HTML5.

Cree una página inicio.php que permita solicitar la inscripción a formación.

# Bienvenido al sitio de inscripción "Formación para todos"

Rellene todos los campos del formulario y  
haga clic en el botón Enviar para validar su inscripción

Nombre:

Apellido:

Título de la formación:

Inicio de la formación:

Fin de la formación:

Dirección email:

Acepto las condiciones, accesibles en [este enlace](#).

Todos los campos son obligatorios. El Enlace muestra un pdf que contiene las condiciones generales y es necesario haber pulsado sobre el enlace y después marcar la casilla de selección para poder enviar el formulario.

A continuación debe crear una página conexión.php encargada de realizar la conexión a la base de datos y después una página insertar.php que compruebe si la persona todavía no está registrada con su email y almacene los datos del formulario en la base de datos. Esta página también debe comprobar los errores y devolver un mensaje en la página inicio.php.

 Se propone otros ejercicios relacionados con las base de datos al final de este libro.

## 2. Soluciones

### Solución del ejercicio 1

En este ejemplo el login es Estefania y la contraseña es Morales\_HH.

- Cree la base de datos \_prueba.
- El script SQL de creación de la tabla inicio de sesión\_password es:

```
CREATE TABLE usuario_password (
Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
usuario VARCHAR( 20 ) NOT NULL ,
Password VARCHAR( 20 ) NOT NULL
) ENGINE = MYISAM ;
```

- login.php:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejercicio login</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  </head>
<body>
<h2>Rellene su login y su contraseña</h2>
<form action="comprobar_login.php" method="POST">
login:<input type="text" name="login" /><br />
contraseña:<input type="text" name="password" /><br />
<input type="submit" name="enviar" value="validar"/>
<br />
<?php
if (isset($_GET['mensaje']) && $_GET['mensaje'] == '1') {
  echo "<span style='color:#ff0000'>login incorrecto</span>";
}
?>
</form>
</body>
</html>

```

- comprobar\_login.php:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Ejercicio login</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
  </head>
<body>
<?php

try
{
  $base = new PDO('mysql:host=127.0.0.1;dbname=_prueba', 'root', '');
  $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $sql = "SELECT * FROM usuario_password WHERE usuario =
:login and password = :password";
  // Preparación de la consulta con los marcadores
  $resultado = $base->prepare($sql);
  $login = htmlentities(addslashes($_POST['login']));
  $password = htmlentities(addslashes($_POST['password']));
  $resultado->bindValue(':login', $login);
  $resultado->bindValue(':password', $password);
  $resultado->execute();
  $numero_registro = $resultado->rowCount(); //método que devuelve
                                              //el número de registros
  //Si hay un registro, es porque la persona existe en base de
  //datos
}

```

```

if ($numero_registro != 0) {
    echo "<h2>¡login correcto!</h2>";
}
else {
    header("location:login.php?mensaje=1");
}
$resultado->closeCursor();

}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->GetMessage());
}

?>
</body>
</html>

```

## **Solución del ejercicio 2**

- Cree la base de datos de formación.
- Script SQL de creación de la tabla inscripción:

```

CREATE TABLE IF NOT EXISTS `inscripcion` (
  `Id` int(11) NOT NULL AUTO_INCREMENT,
  `Nombre` varchar(25) NOT NULL,
  `Apellidos` varchar(25) NOT NULL,
  `Titulo` varchar(100) NOT NULL,
  `Inicio` date NOT NULL,
  `Fin` date NOT NULL,
  `Email` varchar(100) NOT NULL,
  PRIMARY KEY (`Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

```

- Página inicio.php

```

<!doctype html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Inscripció a Formació para todos</title>

    <!-- El estilo de los botones con bootstrap -->
    <link href="css/bootstrap.css" rel="stylesheet">
    <link href="css/bootstrap-responsive.css" rel="stylesheet">
    <style type="text/css">
        body {
            color:#C73333;
        }
        a {
            color:#C73333;
        }
    </style>

```

```

}

#div1 {
    background-color:#ffffff;
    margin:20px;
    width:80%;
}

#casel {
    margin-right:20px;
    margin-top:-4px;
}

</style>

<script>
// Función de comprobación de la aceptación y otros campos para
los navegadores no compatibles con HTML5
function verif() {
    if (document.form1.aceptacion.checked == false) {
        alert("Debe aceptar las condiciones.");
        return false;
    }
    else {
        //para IE8 y <
        if (document.form1.nom.value==""
            || document.form1.apellidos.value==""
            || document.form1.titulo.value==""
            || document.form1.inicio.value==""
            || document.form1.fin.value==""
            || document.form1.email.value=="" ) {
            alert("Todos los campos son obligatorios.");
            return false;
        }
        else {
            return true;
        }
    }
}

var check_ok=0; // inicialización a 0 -> el usuario no ha leído
las condiciones.
function verif_check() {
    if (check_ok == 0) {
        document.form1.aceptacion.checked = false;
        alert("Debe leer las condiciones.");
    }
}

function maj_check() {
    check_ok=1; // el usuario ha leído las condiciones
}

</script>

<!-- Jquery para gestionar los datepicker -->
<link rel="stylesheet"
href="http://code.jquery.com/ui/1.9.0/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.8.2.js">
</script>

```

```

<script src="http://code.jquery.com/ui/1.9.0/jquery-ui.js">
</script>
<script>
$(function() {
    $("#inicio").datepicker();
});

$(function() {
    $("#fin").datepicker();
});

jQuery(function($){
    $.datepicker.regional['es'] = {
        closeText: 'Cerrar',
        prevText: '&#x3c;Prec',
        nextText: 'Siguiente;',
        currentText: 'Actual',
        monthNames: ['Enero','Febrero','Marzo','Abril','Mayo',
        'Junio','Julio','Agosto','Septiembre','Octubre','Noviembre',
        'Diciembre'],
        monthNamesShort: ['Ene','Feb','Mar','Abr','May','Jun',
        'Jul','Ags','Sep','Oct','Nov','Dic'],
        dayNames: ['Domingo','Lunes','Martes','Miércoles','Jueves',
        'Viernes','Sábado'],
        dayNamesShort: ['Dom','Lun','Mar','Mir','Jue','Vie','Sáb'],
        dayNamesMin: ['Do','Lu','Ma','Mi','Ju','Vi','Sa'],
        weekHeader: 'Sm',
        dateFormat: 'dd/mm/yy',
        firstDay: 1,
        isRTL: false,
        showMonthAfterYear: false,
        yearSuffix: ''};
    $.datepicker.setDefaults($.datepicker.regional['es']);
});
</script>

</head>
<body>
<div id="div1">
    <h2>Bienvenido al sitio de inscripción "Formación para todos"</h2><br />
    <h4>Rellene todos los campos del formulario y</h4>
    <h4>haga clic en el botón Enviar para validar su inscripción</h4><br />
    <form action="insertar.php" name="form1" method="POST"
onsubmit="return verif()">
        <table>
            <tr><td>Nombre:</td><td><input type="text" name="nom"
placeholder="Escriba un nombre" required="required" maxlength="25"/>
</td></tr>
            <tr><td>Apellido:</td><td><input type="text" name=
"apellidos" placeholder="Escriba sus apellidos" required="required"
maxlength="25"/>
</td></tr>
            <tr><td>Título de la formación:</td><td><input

```

```

type="text" name="titulo" placeholder="Título" required="required" maxlength="100"/></td></tr>
<tr><td>Inicio de la formación:</td><td><input type="text" name="inicio" id="inicio" placeholder="Indique la fecha de inicio" required="required" /></td></tr>
<tr><td>Fin de la formación:</td><td><input type="text" name="fin" id="fin" placeholder="Indique la fecha de fin" required="required" /></td></tr>
<tr><td>Dirección email:</td><td><input type="email" name="email" placeholder="Dirección de mail validada" required="required" maxlength="100"/></td></tr>
<tr><td align="right"><input type="checkbox" name="aceptacion" id="casel" onclick="verif_check()"/></td><td>Acepto las condiciones, accesibles en <a href="conditions.pdf" style="color:blue" onclick="maj_check()" target="_blank">este enlace</a>.</td></tr>
<tr><td colspan="2"><br /></td></tr>
<tr><td colspan="2" align="center"><input type="submit" name="go" value="Enviar" /></td></tr>
</table>
</form>
<br />
<h4 style="color:red;">
<?php
if (isset($_GET['mensaje']) && $_GET['mensaje'] == 1) {
    echo "Su inscripción se ha registrado.";
}
if (isset($_GET['mensaje']) && $_GET['mensaje'] == 2) {
    echo "Uno de los campos está vacío.";
}
if (isset($_GET['mensaje']) && $_GET['mensaje'] == 3) {
    echo "Ya ha realizado la solicitud.";
}
?>
</h4>
<br />

</div>

</body>
</html>

```

- Página conexión.php:

```

<?php
//conexión a la base de datos
$base = mysqli_connect("127.0.0.1", "root", "", "formacion");
?>

```

- Página insertar.php

```

<?php
include('conexion.php');

```

```

if ($base)
{
    $message=0;
    $Nombre = htmlentities(addslashes($_POST['nombre']));
    //funciones para evitar la inyección de script malintencionado
    $Apellidos = htmlentities(addslashes($_POST['apellidos']));
    $titulo = htmlentities(addslashes($_POST['titulo']));
    $inicio = htmlentities(addslashes($_POST['inicio']));
    $Fin = htmlentities(addslashes($_POST['fin']));
    $Email = htmlentities(addslashes($_POST['email']));
    if ($Nombre == "" || $Apellidos == "" || $titulo == "" ||
    $inicio == "" || $Fin == "" || $Email == "") {
        // caso en el que el usuario esquiva el javascript
        $mensaje=2; //uno de los campos está vacío
    }
    else {
        $nbr=0; //nombre de persona que ya tiene este email
        // VERIFICACIÓN SI LA PERSONA ESTÁ INSCRITA
        =====
        $sql = "SELECT Nombre, Apellidos FROM inscripcion WHERE
Email LIKE ? OR (Nombre LIKE ? AND Apellidos LIKE ?)";
        // Preparación de la consulta
        $resultado = mysqli_prepare($base, $sql);
        $ok = mysqli_stmt_bind_param($resultado, 'sss', $Email,
$Nombre, $Apellidos);
        // Ejecución de la consulta.
        $ok = mysqli_stmt_execute($resultado);
        if ($ok == FALSE) {
            echo "Error de ejecución de la consulta.<br />";
            $mensaje=0;
        }
        else {
            // Asociación de las variables de resultado.
            $ok = mysqli_stmt_bind_result($resultado, $Nombre,
$Apellidos);
            // Almacena los valores.
            $ok = mysqli_stmt_store_result($resultado);
            $nbr=mysqli_stmt_num_rows($resultado);
            //Libera el resultado
            mysqli_stmt_free_result($resultado);
            mysqli_stmt_close($resultado);

        }
        if ($nbr > 0) {
            $mensaje=3; //Ya ha realizado la petición";
        }
        else {
            // INSERCIÓN DE NUEVOS DATOS
            =====
            =====
            $sql = "INSERT INTO inscripcion (Nombre, Apellidos,
titulo, inicio, Fin, Email) VALUES (?,?,?,?,?,?)";
            // Preparación de la consulta
            $resultado_insert = mysqli_prepare($base, $sql);
            date_default_timezone_set('Europe/Paris');

```

```

$dt_inicio = date_create_from_format('d/m/Y', $inicio);
//creación de un objeto Date a partir de la cadena de caracteres
//$inicio
$dt_fin = date_create_from_format('d/m/Y', $Fin);
//creación de un objeto Date a partir de la cadena de caracteres $Fin
$ok = mysqli_stmt_bind_param($resultado_insert,
'ssssss', $Nombre, $Apellidos, $titulo, $dt_inicio->format('Y/m/d'),
$dt_fin->format('Y/m/d'), $Email);
// Ejecución de la consulta.
$ok = mysqli_stmt_execute($resultado_insert);
if ($ok == FALSE)
{
    echo "Error de ejecución de la consulta.<br />";
    $mensaje=0;
}
else
{
    $mensaje=1; //echo "Datos añadidos.";
}
mysqli_stmt_close($resultado_insert);
}

if (mysqli_close($base) == false) {
    echo 'Error de desconexión.';
    $mensaje=0;
}
}
else
{
    printf('Error %d : %s.<br />', mysqli_connect_errno(),
    mysqli_connect_error());
    $mensaje=0;
}

if ($mensaje!=0) { //No redirigir automáticamente a la página
//de inicio en caso de error grave.
    header("Location:inicio.php?mensaje=". $mensaje);
}
?>

```

- Cree un archivo condiciones.pdf.
- Puede descargar bootstrap en la dirección <http://getbootstrap.com/2.3.2/> y meter los archivos bootstrap.css y bootstrap-responsive.css en el directorio css para mejorar el estilo de la página.

## Introducción

El objetivo de este capítulo no es explicar todas las complejidades de la programación orientada a objetos (POO), sino ver las bases para que pueda programar un código simple o entender cómo funciona un código objeto que ya existe.

Por ahora, el código que se muestra es de carácter procedimental, es decir, puede crear las funciones a las que llama cuando las necesita, todo en orden cronológico.

En POO, casi todo son objetos y todos los objetos interactúan entre sí.

Un objeto tiene unas características, que son los atributos, y unas acciones, que son los métodos.

Por ejemplo, el objeto Animal tiene como atributos el color y el peso, y tiene como métodos moverse y comer.

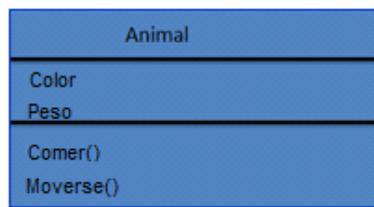
Para construir estos objetos, debe utilizar una clase.

# La clase

## 1. Introducción

Una clase sirve para fabricar objetos partiendo de un modelo. Estos objetos tienen sus propios atributos y ciertos métodos.

Por ejemplo, la clase Animal tiene los atributos color y peso y los métodos comer y moverse.



Cuando se crean ejemplares de animales en la clase Animal, se crea una instancia de esta clase. Crear una instancia de una clase significa que se crea un objeto de un tipo determinado (Animal) con ciertos atributos (color, peso).

Creación de una clase en PHP:

```
<?php
    class Animal // palabra clave class seguida del nombre de la clase.
    {
        // Declaración de atributos y métodos.
    }
?>
```

Se recomienda crear una clase por cada archivo PHP que tenga el mismo nombre que la clase.

## 2. La encapsulación

Todos los atributos en POO deben estar ocultos a otras personas que utilizan sus clases. Si trabaja en equipo y crea la clase Animal, los otros programadores no van a poder cambiar directamente los atributos de su clase. De esta forma, los atributos color y peso se ocultan a otras clases; se declaran privadas. La clase Animal tiene métodos para leer o escribir estos atributos. Este es el principio de encapsulación, que permite proteger el código cuando trabaja en equipo.

La clase Animal, que tiene como propiedades el color y el peso, dispone de un método para modificar su color, un método para leer el color, un método para modificar su peso, un método para leer su peso, así como otros métodos tales como comer o moverse (ver la sección Actualizar y leer los atributos de la instancia, más adelante en este capítulo).

## 3. Visibilidad de los atributos y de los métodos

Hay tres tipos de palabra clave para definir la visibilidad de un atributo o de un método:

- **private:** solo el código de su clase puede ver y acceder a este atributo o método.

- **public:** todas las demás clases pueden acceder a este atributo o método.
- **protected:** solo el código de su clase y de sus subclases pueden acceder a este atributo o método.

Las subclases se tratarán en la sección correspondiente a la herencia.

Creación de una clase en PHP con sus atributos:

```
<?php
    class Animal // palabra clave seguida del nombre de la clase.
    {
        // Declaración de atributos.
        private $color;
        private $peso;
    }
?>
```

Puede definir los valores por defecto de sus atributos:

```
<?php
    class Animal // palabra clave seguida del nombre de la clase.
    {
        // Declaración de atributos.
        private $color = "gris";
        private $peso = 10;
    }
?>
```

Para añadir los métodos a su clase, las normas de visibilidad son las mismas que en los atributos:

```
<?php
    class Animal // palabra clave seguida del nombre de la clase.
    {
        // Declaración de atributos y métodos.
        private $color = "gris";
        private $peso = 10;

        public function comer()
        {
            //Método posible para acceder a las propiedades
            //color y peso
        }

        public function moverse()
        {
            //Método posible para acceder a las propiedades
            //color y peso
        }
    }
}
```

## 4. Añadir un método a la clase

Añadir código a la clase significa aplicar la clase. Para acceder a los atributos de su clase, debe utilizar la pseudovariáble `$this`, que representa el objeto sobre el que va a escribir.

Para acceder al atributo o al método del objeto, utilice el operador `->`.

Por ejemplo, para aplicar el método `añadir_un_kilo()` en la clase Animal:

```
<?php
class Animal // palabra clave seguida del nombre de la clase.
{
    // Declaración de atributos y métodos.
    private $color = "gris";
    private $peso = 10;

    public function comer()
    {
        //Método para acceder a las propiedades
        //color y peso
    }

    public function moverse()
    {
        //Método para acceder a las propiedades
        //color y peso
    }

    public function añadir_un_kilo()
    {
        $this->peso = $this->peso + 1;
    }
}
```

Cuando llama al método `añadir_un_kilo()`, añadirá 1 al peso actual y por lo tanto el peso final será 11.

 Las propiedades se declaran con el símbolo `$`, pero se llaman con `$this` sin este símbolo.

## 5. Utilización de la clase

Como primer paso, tenemos que crear un archivo que contenga un código PHP `Animal.class.php`.

Para utilizar la clase `Animal`, debe incluirla en la página donde la quiere llamar.

Cree una página `uso.php` y escriba el siguiente código:

```
<?php  
  
include('Animal.class.php');  
  
?>
```

Ahora que ha cargado la clase, puede instanciarla, es decir, crear un objeto que tenga como modelo la clase Animal:

```
<?php  
  
//carga de la clase  
include('Animal.class.php');  
  
//instanciar la clase Animal  
$perro = new Animal();  
?>
```

La variable \$perro es una instancia de la clase Animal, con los atributos propios de color, peso, y como métodos comer, moverse, añadir\_un\_kilo.

## 6. Actualizar y leer los atributos de la instancia

El principio de encapsulación requiere que todos los atributos sean privados. Por lo tanto, debe crear métodos públicos que permitan leer o escribir sus atributos desde otra página PHP.

Estos métodos son accesos.

Generalmente sus nombres van precedidos del prefijo get para leer el valor del atributo y set para escribir el valor del atributo.

La clase Animal con los accesos es:

```
<?php  
  
class Animal // palabra clave seguida del nombre de la clase.  
{  
    // Declaración de atributos  
    private $color = "gris";  
    private $peso = 10;  
  
    //accesos  
    public function getColor()  
    {  
        return $this->color; //devuelve el color  
    }  
    public function setColor($color)  
    {  
        $this->color = $color; //escrito en el atributo color  
    }  
}
```

```

public function getPeso()
{
    return $this->peso; //devuelve el peso
}
public function setPeso($peso)
{
    $this->peso = $peso; //escrito en el atributo peso
}

//métodos
public function comer()
{
    //método para acceder a las propiedades
    //color y peso
}

public function moverse()
{
    //método para acceder a las propiedades
    //color y peso
}

public function añadir_un_kilo()
{
    $this->peso = $this->peso + 1;
}

}
?>

```

Los accesos son públicos y por lo tanto permiten leer o escribir en los atributos desde cualquier otra clase o página PHP.

Ejemplo con la página uso.php:

```

<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal
$perro = new Animal();

//leer el peso
echo "El peso del perro es:".$perro->getPeso()." kg<br />";
//añadir un kilo al perro
$perro->añadir_un_kilo();
//leer el peso
echo "El peso del perro es:".$perro->getPeso()." kg<br />";
//actualizar el peso del perro
$perro->setPeso(15);
//leer el peso
echo "El peso del perro es:".$perro->getPeso()." kg<br />";

```

```
?>
```

Da como resultado:

```
El peso del perro es:10 kg  
El peso del perro es:11 kg  
El peso del perro es:15 kg
```

En efecto, el peso del perro se inicializa a 10. A continuación el método `añadir_un_kilo()` añade 1; por lo tanto, su peso se convierte en 11. Para terminar, el método `setPeso(15)` ajusta el peso a 15.

Puede crear tantas instancias como quiera.

Por ejemplo, para crear un gato blanco de 5 kg y un perro negro de 18 kg:

```
<?php  
  
//carga de la clase  
include('Animal.class.php');  
  
//instanciar la clase Animal  
$perro = new Animal();  
//actualizar el peso del perro  
$perro->setPeso(18);  
//leer el peso  
echo "El peso del perro es:".$perro->getPeso()." kg<br />";  
//actualizar el color del perro  
$perro->setColor("negro");  
//leer el color  
echo "El color del perro es:".$perro->getColor()."<br />";  
  
//instanciar la clase Animal  
$gato = new Animal();  
//actualizar el peso del gato  
$gato->setPeso(5);  
//leer el peso  
echo "El peso del gato es:".$gato->getPeso()." kg<br />";  
//actualizar el color del gato  
$gato->setColor("blanco");  
//leer el color  
echo "El color del gato es:".$gato->getColor()."<br />";  
  
?>
```

Da como resultado:

```
El peso del perro es:18 kg  
El color del perro es:negro  
El peso del gato es:5 kg
```

## 7. Paso como argumento de tipo objeto

Los métodos son como las funciones, pueden tomar argumentos de tipos diferentes (Integer, String...) e incluso de tipo Objeto.

\$gato y \$perro son objetos de tipo Animal. Pueden pasar como argumento un método, siempre y cuando acepte este tipo de objeto.

Para probar este ejemplo, cambie el método comer() de la clase Animal. Se convierte en comer\_animal (Animal \$animal\_comido) y toma como argumento un objeto de tipo Animal.

La página Animal.class.php se convierte en:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;

    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escrito en el atributo color
    }

    public function getPeso()
    {
        return $this->peso; //devuelve el peso
    }
    public function setPeso($peso)
    {
        $this->peso = $peso; //escrito en el atributo peso
    }

    //Métodos
    public function comer_animal(Animal $animal_comido)
    {
        //el animal que come aumenta su peso tanto
        //como el del animal comido
        $this->peso = $this->peso + $animal_comido->peso;
        //el peso del animal comido y su color se restablecen a 0
        $animal_comido->peso = 0;
        $animal_comido->color = "";
    }
}
```

```

public function moverse()
{
    //método que pueda acceder a las propiedades
    //color y peso
}

public function añadir_un_kilo()
{
    $this->peso = $this->peso + 1;
}

}

?>

```

Para probar este método, la página uso.php se convierte en:

```

<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal
$gato = new Animal();
//actualizar el peso del gato
$gato->setPeso(8);
//leer el peso
echo "El peso del gato es:".$gato->getPeso()." kg<br />";
//actualizar el color del gato
$gato->setColor("negro");
//leer el color
echo "El color del gato es:".$gato->getColor()."<br />";

//instanciar la clase Animal
$pez = new Animal();
//actualizar el peso del pez
$pez->setPeso(1);
//leer el peso
echo "El peso del pez es:".$pez->getPeso()." kg<br />";
//actualizar el color del pez
$pez->setColor("blanco");
//leer el color
echo "El color del pez es:".$pez->getColor()."<br /><br />";

//el gato come al pez
$gato->comer_animal($pez);
//leer el peso
echo "El nuevo peso del gato es:".$gato->getPeso()." kg<br />";
//leer el peso
echo "El peso del pez es:".$pez->getPeso()." kg<br />";
//leer el color
echo "El color del pez es:".$pez->getColor()."<br /><br />";

```

```
?>
```

Da como resultado:

```
El peso del gato es:8 kg  
El color del gato es:negro  
El peso del pez es:1 kg  
El color del pez es:blanco  
El nuevo peso del gato es:9 kg  
El peso del pez es:0 kg  
El color del pez es:
```

El objeto \$gato llama al método comer\_animal (\$pez) y pasa como argumento el objeto de tipo Animal \$pez. Es decir, el objeto \$pez con sus atributos y sus métodos se pasan como argumento. Esto permite pasar como argumento varios valores con un único parámetro. El método comer\_animal(Animal \$animal\_comido) solo acepta un argumento de tipo Animal.

Por lo tanto, no puede llamar al método de la siguiente manera:

```
$gato->comer_animal("Rana");
```

O de esta manera:

```
$gato->comer_animal(4);
```

Ya que los tipos "Rana" (String) y 4 (Integer) no son de tipo Animal.

## 8. El constructor

El constructor, como su nombre indica, sirve para construir un objeto del tipo clase. Cuando escribe new Animal (), por defecto llama al constructor de la clase Animal.

Puede crear sus propios constructores y así pasar como argumento el valor de los atributos que desea asignar a su objeto.

El constructor se designa \_\_construct y no tiene return.

Para añadir un constructor que toma como argumentos el peso y el color, la página Animal.class.php se convierte en:

```
<?php  
class Animal  
{  
    // Declaración de atributos  
    private $color = "gris";  
    private $peso = 10;
```

```

    public function __construct ($color, $peso)
//Constructor que solicita 2 argumentos.
{
    echo 'Llamar al constructor.<br />';
    $this->color = $color; // Inicialización del
                           // color.
    $this->peso = $peso; // Inicialización del peso.
}
Etc.
.
.
.
?
?>

```

Llamar al constructor en la página uso.php:

```

<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal con su constructor
$perro = new Animal("beige",7);
//leer el peso
echo "El peso del perro es:".$perro->getPeso()." kg<br />";
//leer el color
echo "El color del perro es:".$perro->getColor()."<br />";

//actualizar el color del perro
$perro->setColor("negro");
//leer el color
echo "El color del perro es:".$perro->getColor()."<br />";

?>

```

Da como resultado:

```

Llamar al constructor.

El peso del perro es:7 kg

El color del perro es:beige

El color del perro es:negro

```

Se muestra en primer lugar "Llamar al constructor", ya que la instrucción echo que se ha escrito en el constructor \_\_construct de su clase Animal se llama cada vez que ejecuta new Animal().

El constructor toma como argumento los valores de sus atributos. Esto evita llamar los métodos setColor() y setPeso().



En PHP no se puede declarar dos constructores en la misma clase.

## 9. El destructor

El destructor sirve para destruir el objeto con el fin de liberarlo de la memoria. Se llama automáticamente al final del script PHP o cuando se destruye el objeto.

Para destruir un objeto, puede utilizar la función `unset()`. Esta función toma como argumento el objeto que hay que destruir.

Por ejemplo, para destruir el objeto `$perro`:

```
<?php  
  
//destrucción del objeto  
unset($perro);  
  
?>
```

Por defecto llama al destructor. Puede modificarlo si añade la función `__destruct()` en la clase.

```
<?php  
class Animal  
{  
    // Declaración de atributos  
    private $color = "gris";  
    private $peso = 10;  
  
    public function __construct ($color, $peso)  
    //Constructor que solicita 2 argumentos.  
    {  
        echo 'Llamar al constructor.<br />';  
        $this->color = $color; // Inicialización del  
                           // color.  
        $this->peso = $peso; // Inicialización del peso.  
    }  
  
    public function __destruct()  
    {  
        echo 'Llamar al destructor';  
    }  
  
Etc.  
. . .  
}  
?>
```

Generalmente no es necesario aplicar el destructor en la clase.

## 10. Ejercicio

### **Enunciado (fácil)**

Cree dos peces en la página uso.php:

- pez1, gris, 10 kg
- pez2, rojo, 7 kg

Muestre su peso y a continuación el pez1 se come al pez2.

Vuelva a mostrar su peso.

### **Solución**

```
<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal con su constructor
$pez1 = new Animal("gris",10);
$pez2 = new Animal("rojo",7);
//leer el peso
echo "El peso del pez1 es:".$pez1->getPeso()." kg<br />";
//leer el peso
echo "El peso del pez2 es:".$pez2->getPeso()." kg<br />";

//el pez1 se come al pez2
$pez1->comer_animal($pez2);
//leer el peso
echo "El nuevo peso del pez1 es:".$pez1->getPeso()." kg<br />";
//leer el nuevo peso
echo "El nuevo peso del pez2 es:".$pez2->getPeso()." kg<br />";

?>
```

Da como resultado:

Llamada al constructor.

Llamada al constructor.

El peso del pez1 es:10 kg

El peso del pez2 es:7 kg

El nuevo peso del pez1 es:17 kg

El nuevo peso del pez2 es:0 kg

Llamada al destructor

## 11. Las constantes de clase

Una constante de clase es similar a una constante normal, es decir, un valor asignado a un nombre y que no cambia nunca.

Ejemplo de declaración de una constante normal:

```
define('PI',3.1415926535);
```

Una constante de clase representa una constante pero que está unida a esta clase.

Para crear un animal con el constructor `__construct ($color, $peso)`, escriba:

```
$perro = new Animal("gris",10);
```

Si lee el código, no puede saber inmediatamente que el número 10 representa el peso del animal.

Utilice constantes que representen, cada una, un peso distinto:

```
const PESO_LIGERO = 5;
const PESO_MEDIO = 10;
const PESO_PESADO = 15;
```

Las constantes siempre están en mayúsculas, sin el símbolo \$ y precedidas de la palabra clave const.

La clase Animal.class.php se convierte en:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;

    //constantes de clase
    const PESO_LIGERO = 5;
    const PESO_MEDIO = 10;
    const PESO_PESADO = 15;
etc.
.
.
?>
```

Para llamar a esta constante desde la página uso.php, la sintaxis es algo peculiar. Debe escribir :: entre la clase y

su constante:

```
<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal con su constructor
$pez1 = new Animal("gris",Animal::PESO_MEDIO);
$pez2 = new Animal("rojo",Animal::PESO_LIGERO);
//leer el peso
echo "El peso del pez1 es:".$pez1->getPeso()." kg<br />";
//leer el peso
echo "El peso del pez2 es:".$pez2->getPeso()." kg<br />";

//el pez1 se come al pez2
$pez1->comer_animal($pez2);
//leer el peso
echo "El nuevo peso del pez1 es:".$pez1->getPeso()." kg<br />";
//leer el nuevo peso
echo "El nuevo peso del pez2 es:".$pez2->getPeso()." kg<br />";

?>
```

Da como resultado:

Llamada al constructor.

Llamada al constructor.

El peso del pez1 es:10 kg

El peso del pez2 es:5 kg

El nuevo peso del pez1 es:15 kg

El nuevo peso del pez2 es:0 kg

Animal::PESO\_MEDIO siempre es 10, sea cual sea la instancia. Por lo tanto, la constante no está unida a la instancia, sino a la clase. Por este motivo la sintaxis es peculiar.

## 12. Los atributos y métodos estáticos

### a. Método estático

El método estático está unido a la clase, pero no al objeto. En el ejemplo de la clase Animal, un método estático está unido al Animal, y no a los perros, los gatos o los peces.

Para convertir un método estático, debe añadir la palabra clave `static` delante de `function`.

Por ejemplo, modifique el método `moverse()` para convertirlo en estático y muestre "El animal se mueve."

La clase `Animal.class.php` se convierte en:

```

<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;

    //constantes de clase
    const PESO_LIGERO = 5;
    const PESO_MEDIO = 10;
    const PESO_PESADO = 15;

    public function __construct ($color, $peso)
// Constructor que solicita 2 argumentos.
    {
        echo 'Llamada al constructor.<br />';
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.
    }

    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escrito en el atributo color
    }

    public function getPeso()
    {
        return $this->peso; //devuelve el peso
    }
    public function setPeso($peso)
    {
        $this->peso = $peso; //escrito en el atributo peso
    }

    //métodos
    public function comer_animal(Animal $animal_comido)
    {
        //el animal que come aumenta su peso tanto como
        //el del animal comido
        $this->peso = $this->peso + $animal_comido->peso;
        //el peso del animal comido y su color se restablecen a 0
        $animal_comido->peso = 0;
        $animal_comido->color = "";
    }

    public static function moverse()
    {
        echo "El animal se mueve.";
    }
}

```

```

    }

    public function añadir_un_kilo()
    {
        $this->peso = $this->peso + 1;
    }

}

?>

```

Es imposible escribir en un método estático la palabra clave `$this`, ya que representa el objeto, y el método estático está unido a la clase.

Para llamar a este método desde la página uso.php, debe utilizar la misma sintaxis que en las constantes (también unidas a la clase), es decir, introducir `::` entre la clase y su método estático:

```

<?php

//carga de la clase
include('Animal.class.php');

//llamada al método estático
Animal::moverse()

?>

```

Da como resultado:

El animal se mueve.

Puede llamar al método estático desde un objeto, pero el método estático no puede cambiar nada de este objeto:

```

<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal con su constructor
$perro1 = new Animal("gris",Animal::PESO_MEDIO);

//llamada al método estático
$perro1->moverse();

?>

```

Da como resultado:

Llamada al constructor

## b. Atributo estático

Un atributo estático es un atributo propio de la clase y no del objeto, al igual que en los métodos estáticos. Es el mismo principio que en una constante, salvo que el atributo está en una variable y puede cambiar su valor.

Un atributo estático se escribe añadiendo la palabra clave `static` delante de su nombre.

Por ejemplo, para añadir un atributo estático que representa a un contador que indica el número de veces que se instancia la clase:

La clase `Animal.class.php` se convierte en:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;

    // constantes de clase
    const PESO_LIGERO = 5;
    const PESO_MEDIO = 10;
    const PESO_PESADO = 15;

    // Declaración de la variable estática $contador
    private static $contador = 0;

etc.
.
.
.
?>
```

Para cambiar el valor de este contador, no puede utilizar `$this`. De hecho, `$this` representa un objeto (perro, gato), y no la clase `Animal`. El contador es de tipo estático y por lo tanto está unido a la clase. Para llamar a este atributo en la clase, debe utilizar la palabra clave `self`, que representa la clase.

Para añadir 1 al contador cada vez que vaya a instanciar la clase `Animal`, debe modificar el constructor. A continuación debe añadir un método que permita leer este atributo privado con ayuda de un método de tipo `public static` y `getContador()`.

La clase `Animal.class.php` se convierte en:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;
```

```

// constantes de clase
const PESO_LIGERO = 5;
const PESO_MEDIO = 10;
const PESO_PESADO = 15;

// Declaración de la variable estática $contador
private static $contador = 0;

public function __construct ($color, $peso) // Constructor
//que solicita 2 argumentos.
{
    echo 'Llamada al constructor.<br />';
    $this->color = $color; // Inicialización del color.
    $this->peso = $peso; // Inicialización del peso.

    self::$contador = self::$contador + 1;
}

// método estático que devuelve el valor del contador
public static function getContador()
{
    return self::$contador;
}

...
?>
```

La página uso.php:

```

<?php

//carga de la clase
include('Animal.class.php');

//instanciar la clase Animal
$perro1 = new Animal("rojo",10);
//instanciar la clase Animal
$perro2 = new Animal("gris",5);
//instanciar la clase Animal
$perro3 = new Animal("negro",15);
//instanciar la clase Animal
$perro4 = new Animal("blanco",8);

//llamada al método estático
echo "Número de animales que se han instanciado:".Animal::
getContador();

?>
```

Da como resultado:

Llamada al constructor.

Llamada al constructor.

Llamada al constructor.

Llamada al constructor.

Número de animales que se han instanciado:4

# La herencia

## 1. Introducción

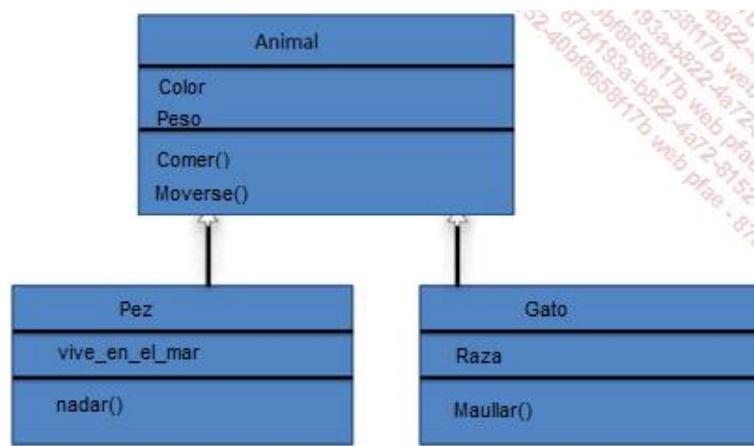
La herencia es un concepto muy importante en POO. Permite reutilizar el código de una clase sin necesidad de volver a escribirlo.

Una clase hija hereda de una clase madre, es decir, accede a todos los atributos y los métodos públicos de la clase madre.

Por ejemplo, la clase Mamífero hereda de la clase Animal, y la clase Coche hereda de la clase Vehículo.

Si la clase A es una subcategoría de la clase B, entonces puede hacer que la clase A (Mamífero o Coche) herede de la clase B (Animal o Vehículo).

En el siguiente esquema puede observar como las clases Pez y Gato son ejemplos de herencia de la clase Animal.



Para crear la clase Pez que hereda de la clase Animal, debe utilizar la palabra clave `extends` entre el nombre de la clase hija y el nombre de la clase madre.

Cree un archivo `Pez.class.php` y escriba el siguiente código:

```
<?php  
class Pez extends Animal  
{  
}  
?>
```

Ahora añada un atributo privado que corresponda a la variable `vive_en_el_mar` y los accesos `get` y `set`. Para terminar, el método público `nadar()`.

```
<?php  
class Pez extends Animal  
{  
    private $vive_en_el_mar; //tipo de pez
```

```

//accesos
public function getType()
{
    if ($this->vive_en_el_mar){
        return "vive_en_el_mar";
    }
    else if ($this->vive_en_el_mar==false){
        return "no_vive_en_el_mar";
    }else {return "";}
}
public function setType($vive_en_el_mar)
{
    $this->vive_en_el_mar = $vive_en_el_mar;
//escrito en el atributo vive_en_el_mar
}
//método
public function nadar()
{
    echo "Nado <br />";
}
}
?>

```

De la misma manera, cree un archivo Gato.class.php:

```

<?php
class Gato extends Animal
{
    private $raza; //raza del gato

    //accesos
    public function getRaza()
    {
        return $this->raza; //devuelve la raza
    }
    public function setRaza($raza)
    {
        $this->raza = $raza; //escrito en el atributo raza
    }
    //método
    public function maullar()
    {
        echo "Miau <br />";
    }

}
?>

```

Las clases Gato y Pez, que heredan de la clase Animal, tienen acceso a los atributos públicos de la clase Animal.

La página uso.php:

```

<?php

//carga de clases
include('Animal.class.php');
include('Pez.class.php');
include('Gato.class.php');

//instanciar la clase Pez que llama al constructor de
//la clase Animal
$pez = new Pez("gris",8);
//instanciar la clase Gato que llama al constructor de la
//clase Animal
$gato = new Gato("blanco",4);
//leer el peso con el acceso de la clase madre
echo "El peso del pez es:".$pez->getPeso()." kg<br />";
//leer el peso con el acceso de la clase madre
echo "El peso del gato es:".$gato->getPeso()." kg<br />";

$pez->setType(true);
//leer el tipo con el acceso de su propia clase
echo "El tipo de pez es:".$pez->getType()."<br />";
//llamada al método de la clase Pez
$pez->nadar();

$gato->setRaza("Angora");
//leer la raza con el acceso de su propia clase
echo "La raza del gato es:".$gato->getRaza()."<br />";
//llamada al método de la clase gato
$gato->maullar();

//llamada al método estático
echo "Número de animales que se han instanciado:"
.Animal::getContador();

?>

```

Da como resultado:

Llamada al constructor.

Llamada al constructor.

El peso del pez es:8 kg

El peso del gato es:4 kg

El tipo de pez es:vive en el mar

Nado

La raza del gato es:Angora

Miau

Número de animales que se han instanciado:2

La clase Pez no tiene acceso a los atributos de la clase Gato y viceversa, ya que una hereda de la otra. Las clases Pez y Gato no tienen directamente acceso a los atributos privados color y peso de la clase Animal. Deben pasar por

sus accesos públicos.

## 2. Protected

Este tipo de visibilidad equivale a `private`, salvo que las clases hijas puedan ver los atributos `protected` de la clase madre.

Por ejemplo, vamos a añadir el atributo `$edad` de visibilidad protegida (`protected`) en la clase madre `Animal`:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;
    protected $edad = 0;
etc.
```

Este atributo no tiene acceso público; por lo tanto, ninguna de las otras clases hijas que heredan de `Animal` y de la clase `Animal` pueden modificarlo o leerlo.

Añada la función `mostrarAtributos()` en la clase `Pez`:

```
public function mostrarAtributos()
{
    echo "Tipo:".$this->vive_en_el_mar; // correcto ya que es
                                            // privada de esta clase
    echo "<br />";
    echo "Edad:".$this->edad; // correcto, ya que el atributo
                            // está protegido en la clase madre.
    echo "<br />";
    echo "Peso:".$this->peso; // error, ya que el atributo es
//privado en la clase madre, el acceso está prohibido. Debe pasar
//por sus accesos públicos para modificar o leer su valor
    echo "<br />";
}
```

Para probar este método, la página `uso.php` se convierte en:

```
<?php

//carga de clases
include('Animal.class.php');
include('Pez.class.php');

//instanciar la clase Pez que llama al constructor de
//la clase Animal
$pez = new Pez("gris",8);
//leer el peso con el acceso de la clase madre
```

```

echo "El peso del pez es:".$pez->getPeso()." kg<br />";

//actualizar el tipo de pez
$pez->setType(true);
//llamada al método mostrando los atributos de la clase Pez
//y Animal
$pez->MostrarAtributos();

?>

```

Da como resultado:

```

Llamada al constructor
El peso del pez es $ kg
Tipo: 1
Edad: 0
Nota: Undefined property: Pez::$peso en C:\Prgrams Files\EasyPHP-DevServer-13.1VC11\data\localweb\Object\Pez.class.php en linea 35

```

La clase Pez no tiene acceso al atributo peso de la clase Animal, ya que es privado.

En conclusión, se recomienda poner los atributos en visibilidad `protected`, ya que la propia clase, las clases hijas y las que heredan tienen acceso a este atributo.

### 3. Sustitución

La Sustitución sirve para modificar un método que ya existe en una clase madre, con el objetivo de cambiar el comportamiento. El método existe en dos clases diferentes y según el contexto se ejecuta el de la clase hija o el de la clase madre.

Por ejemplo, para sustituir el método `comer_animal(Animal $animal_comido)` de la clase Animal con el objetivo de inicializar el tipo de pez comido, debe añadir este método en la clase Pez y aplicarlo de otra manera.

Añada en la clase Pez.class.php:

```

//método sustituido
public function comer_animal(Animal $animal_comido)
{
    if (isset($animal_comido->raza)){
        $animal_comido->raza="";
    }
    if (isset($animal_comido->vive_en_el_mar)){
        $animal_comido->vive_en_el_mar="";
    }
}

```

El problema es que este método inicializa correctamente el atributo `vive_en_el_mar` del pez comido, pero ya no inicializa su peso y su color. No puede cambiar aquí su peso y su color, ya que estos atributos son privados en la clase Animal.

La solución está en llamar al método `comer_animal(Animal $animal_comido)` de la clase Animal en el método `comer_animal(Animal $animal_comido)` de la clase Pez:

```

//Método sustituido
    public function comer_animal(Animal $animal_comido)
    {
        // al método comer_animal() de la clase padre,
        // es decir Animal
        parent::comer_animal($animal_comido);

        if (isset($animal_comido->raza)){
            $animal_comido->raza="";
        }
        if (isset($animal_comido->vive_en_el_mar)){
            $animal_comido->vive_en_el_mar=""
        }
    }
}

```

parent es una palabra clave que designa la clase madre, es decir, la clase Animal.

La página uso.php:

```

<?php

//carga de clases
include('Animal.class.php');
include('Pez.class.php');

//instanciar la clase Pez que llama al constructor de la
//clase Animal
$pez = new Pez("gris",8);
//actualizar el tipo de pez
$pez->setType(true);

//instanciar la clase Pez que llama al constructor de la
//clase Animal
$otro_pez = new Pez("negro",5);
// el tipo de pez
$otro_pez->setType(false);

//llamada al método que muestra los atributos de la clase Pez
//y Animal
$pez->comer_animal($otro_pez);
//leer el tipo por el acceso de su propia clase
echo "El tipo de pez comido es:".$otro_pez->getType()."<br />";
//leer el peso por el acceso de la clase madre
echo "El peso del pez comido es:".$otro_pez->getPeso()." kg<br />";

?>

```

Da como resultado:

Llamada al constructor.

Llamada al constructor.

El tipo de pez comido es:

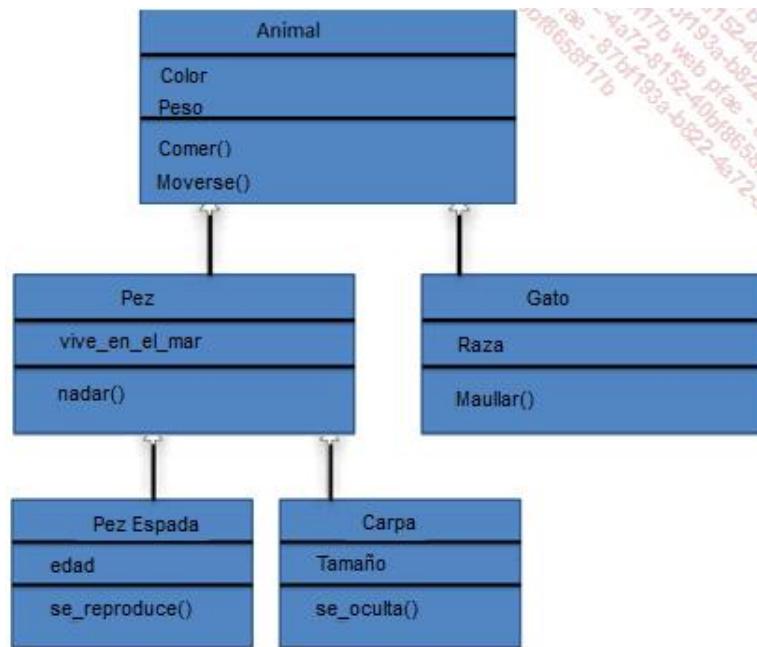
El peso del pez comido es:0 kg

El peso se ha inicializado a 0 por el método comer\_animal(Animal \$animal\_comido) de la clase Animal.

- 💡 En la clase Pez, el método comer\_animal(Animal \$animal\_comido) puede cambiar el atributo tipo en un objeto de tipo Animal, ya que Pez hereda de Animal y, en el archivo uso.php, es un Pez lo que pasa como argumento, y no un Animal. Se trata del polimorfismo de herencia.

## 4. Herencia en cascada

La herencia múltiple no existe en PHP. Una clase solo puede heredar de una única clase, que a su vez puede heredar de una clase, etc.



Este ejemplo muestra que las clases Pez Espada y Carpa heredan de la clase Pez, que a su vez hereda de la clase Animal.

La clase Pez Espada accede a:

- Todos los atributos y métodos privados, protegidos y públicos por sí misma.
- Todos los atributos y métodos protegidos y públicos de la clase Pez.
- Todos los atributos y métodos protegidos y públicos de la clase Animal.
- Todos los atributos y métodos públicos de la clase Gato.

## Las clases abstractas

Las clases abstractas se escriben con la palabra clave `abstract` delante de la palabra `class`. Una clase abstracta no se puede instanciar, es decir, no permite crear una instancia. Puede escribir métodos abstractos, que son métodos donde solo escribe la firma precedida por la palabra clave `abstract`: `abstract visibilidad function nombre_método (atributo tipo_atributo...)`. Estas clases solo sirven para obligar, a las clases que heredan de la clase abstracta, a reemplazar los métodos abstractos declarados en la clase abstracta.

En el siguiente ejemplo, la clase `Animal` es abstracta, ya que no se quiere crear (instanciar) animales, sino peces o gatos.

Añada también un método abstracto `respira()` en la clase `Animal`:

```
<?php
abstract class Animal
{
    // Declaración de atributos
    private $color = "gris";
    private $peso = 10;

    //constantes de clase
    const PESO_LIGERO = 5;
    const PESO_MEDIO = 10;
    const PESO_PESADO = 15;

    // Declaración de la variable estática $contador
    private static $contador = 0;

    public function __construct($color, $peso) // Constructor
    //que solicita 2 argumentos.
    {
        echo 'Llamada al constructor.<br />';
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.

        self::$contador = self::$contador + 1;
    }

    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escrito en el atributo color
    }
    public function getPeso()
    {
        return $this->peso; //devuelve el peso
    }
}
```

```

public function setPeso($peso)
{
    $this->peso = $peso; //escrito en el atributo peso
}

//métodos públicos

public function comer_animal(Animal $animal_comido)
{
    //el animal que come aumenta su peso tanto como
    //el del animal comido
    $this->peso = $this->peso + $animal_comido->peso;
    //el peso del animal comido y su color se restablecen a 0
    $animal_comido->peso = 0;
    $animal_comido->color = "";
}

public static function moverse()
{
    echo "El animal se mueve.";
}

public function añadir_un_kilo()
{
    $this->peso = $this->peso + 1;
}

// método estático que devuelve el valor del contador
public static function getContador()
{
    return self::$contador;
}

//código no aplicado por el método abstracto
abstract public function respira();

}
?>
```

Observe que el método abstracto `respira()` no tiene cuerpo, es decir, no hay llaves {} en la aplicación del método.

Como las clases Pez y Gato heredan de la clase Animal, está obligado a definir de nuevo el método `respira()` en las clases Pez y Gato.

Debe añadir en la clase Pez lo siguiente:

```

public function respira()
{
    echo "El pez respira.<br />";
}
```

Y en la clase Gato:

```
public function respira()
{
    echo "El gato respira.<br />";
}
```

La página uso.php se convierte en:

```
<?php

//carga de clases
include('Animal.class.php');
include('Pez.class.php');
include('Gato.class.php');

//instanciar la clase Pez, que llama al constructor de
//la clase Animal
$pez = new Pez("gris",8);
//instanciar la clase Gato, que llama al constructor de
//la clase Animal
$gato = new Gato("blanco",4);
//llamada al método respira()
$pez->respira();
$gato->respira();

?>
```

Da como resultado:

Llamada al constructor.  
Llamada al constructor.  
El pez respira.  
El gato respira.

## Las clases finales

Cuando una clase es final, no se puede crear la clase hija que hereda de esta clase. Esto tiene poco interés práctico.

Debe añadir la palabra clave `final` delante de la palabra clave `class`.

Por ejemplo, si no crea una clase que hereda de la clase Pez, puede escribir `final`:

```
<?php
final class Pez extends Animal
{
    private $vive_en_el_mar; //tipo de pez

    //accesos
    ...
    //método
    public function nadar()
    {
        echo "Nado <br />";
    }

    public function respira()
    {
        echo "El pez respira.<br />";
    }
}

?>
```

También puede declarar los métodos finales. Estos métodos no se podrán sustituir.

En el párrafo dedicado a la sustitución hemos visto un ejemplo donde el método `comer_animal(Animal $animal_comido)` se ha sustituido en la clase Pez. Si este método era final en la clase Animal:

```
final public function comer_animal(Animal $animal_comido)
{
    //el animal que come aumenta su peso tanto como
    //el del animal comido
    $this->peso = $this->peso + $animal_comido->peso;
    //el peso del animal comido y su color se restablecen a 0
    $animal_comido->peso = 0;
    $animal_comido->color = "";
}
```

Por lo tanto, es imposible sustituir este método en la clase Pez.

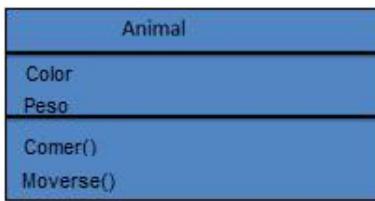
## Los métodos mágicos

Un método mágico es un método al que se llama automáticamente cuando se produce un acontecimiento.

Por ejemplo `__construct` es un método mágico. Se ejecuta automáticamente cuando instancia la clase que contiene `__construct`.

Los métodos mágicos `__get` y `__set` permiten leer o modificar los atributos que no existen y en los que el acceso está prohibido.

Retome el ejemplo al inicio del capítulo con la clase Animal:



Esta vez, el atributo color es privado y el atributo peso es público:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    public $peso = 10;

    public function __construct($color, $peso) // Constructor
    //que solicita 2 argumentos.
    {
        echo 'Llamada al constructor.<br />';
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.

    }

    //métodos públicos
    public function comer ()
    {

    }

    public function moverse ()
    {
    }
}
```

Cuando crea una instancia de la clase Animal, puede acceder al atributo peso porque es público, pero no al atributo

color porque es privado.

La página uso.php es:

```
<?php

//carga de clases
include('Animal.class.php');

//instanciar la clase Animal
$perro = new Animal("gris",8);

$perro->color = "negro";
echo $perro->color."<br />";
?>
```

Muestra un error porque intenta acceder directamente al atributo color, que es privado:

Llamada al constructor.

```
Fatal error: Cannot access private property Animal::$color in C:\Program Files\EasyPHP-DevServer-13.1VC11
\data\localweb\Objeto\uso.php on line 9
```

Ahora añada los métodos mágicos \_\_get y \_\_set en la clase Animal:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    public $peso = 10;
    private $tab_atributos = array();

    public function __construct($color, $peso) // Constructor
    //que solicita 2 argumentos.
    {
        echo 'Llamada al constructor.<br />';
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.

    }

    //métodos mágicos
    public function __get($nombre)
    {
        echo "__get <br />";
        if (isset ($this->tab_atributos[$nombre]))
            return $this->tab_atributos[$nombre];
    }

    public function __set($nombre, $valor)
```

```

{
    echo "__set <br />";
    $this->tab_atributos[$nombre] = $valor;
}

public function __isset($nombre)
{
    return isset ($this->tab_atributos[$nombre]);
}

//métodos públicos
public function comer ()
{
    ...
}

public function moverse ()
{
    ...
}

}

?>

```

Estos métodos se activan automáticamente si el atributo es privado o no existe. Almacenan el valor en la tabla \$tab\_atributos, y en el índice, el nombre del atributo. El método \_\_isset(\$atributo) permite saber si existe el atributo.

La página uso.php:

```

<?php

//carga de clases
include('Animal.class.php');

//instanciar la clase Animal
$perro = new Animal("gris",8);
if (isset($perro->color)) {
    echo "El atributo color existe.<br />";
}
else {
    echo "El atributo color no existe.<br />";
}

$perro->color = "negro";
echo $perro->color."<br />";

if (isset($perro->peso)) {
    echo "El atributo peso existe.<br />";
}
else {
    echo "El atributo peso no existe.<br />";
}

```

```

}

$perro->peso = 25;
echo $perro->peso."<br />";

?>

```

Da como resultado:

Llamada al constructor.

El atributo color no existe.

\_\_set

\_\_get

Negro

El atributo peso existe.

25

Explicación:

`$perro = new Animal("gris", 8);` da como resultado: Llamada al constructor.

`isset($perro->color);` devuelve falso, ya que no se puede acceder al atributo color, que es privado; por lo tanto, muestra: El atributo color no existe.

`$perro->color = "negro";` da como resultado: \_\_set. El atributo color es privado; por lo tanto, no se puede acceder a él y llama automáticamente a \_\_set y el valor se almacena en la tabla `$tab_atributos`.

`echo $perro->color."<br />";` da como resultado: \_\_get y negro. El atributo color siempre es privado; por lo tanto, llama automáticamente a \_\_get para mostrar el color.

`isset($perro->peso);` devuelve verdadero porque el atributo peso es público, se puede acceder a él y por lo tanto muestra: el atributo peso existe.

`$perro->peso = 25;` no muestra nada porque no llama a la función \_\_set. El atributo peso es público, puede acceder directamente a él.

`echo $perro->peso."<br />";` da como resultado 25. El atributo peso es público y por lo tanto puede acceder directamente a él.

Para eliminar un atributo que el método mágico \_\_set ha añadido, debe ejecutar el método mágico \_\_unset (`$atributo`), que eliminará el atributo de la tabla `$tab_atributos`.

Añada este método en la clase Animal:

```

public function __unset($nombre)
{
    if (isset($this->tab_atributos[$nombre]))
        unset($this->tab_atributos[$nombre]);
}

```

Para terminar, los métodos mágicos \_\_call y \_\_callStatic permiten llamar a los métodos privados o que no

existen. La función `method_exists()` comprueba si un método existe en un objeto. Toma como argumento el objeto y el nombre del método. Devuelve `true` si el método existe y `false` si no.

La clase `Animal` con un método público `comer()` y un método privado `moverse()` se convierte en:

```
<?php
class Animal
{
    // Declaración de atributos
    private $color = "gris";
    public $peso = 10;
    private $tab_atributos = array();

    public function __construct($color, $peso) // Constructor
    //que solicita 2 argumentos.
    {
        echo 'Llamada al constructor.<br />';
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.
    }

    //métodos mágicos
    public function __get($nombre)
    {
        echo "__get <br />";
        if (isset ($this->tab_atributos[$nombre]))
            return $this->tab_atributos[$nombre];
    }

    public function __set($nombre, $valor)
    {
        echo "__set <br />";
        $this->tab_atributos[$nombre] = $valor;
    }

    public function __isset($nombre)
    {
        return isset ($this->tab_atributos[$nombre]);
    }

    public function __call($nombre, $argumentos)
    {
        echo "El método ".$nombre." no es accesible. Sus argumentos
        eran los siguientes :".implode($argumentos, ', ')."<br />";
        if(method_exists($this, $nombre))
        {
            $this->$nombre(implode($argumentos, ', '));
        }
    }

    public static function __callStatic($nombre, $argumentos)
    {
```

```

        echo "El método estático ".$nombre." no es accesible.
Sus argumentos eran los siguientes :".implode($argumentos, ', ')
."  


```

La página uso.php:

```

<?php

//carga de clases
include('Animal.class.php');

//instanciar la clase Animal
$perro = new Animal("gris",8);
$perro->comer();
$perro->moverse("París");

?>

```

Da como resultado:

```

Llamada al constructor.

Método público comer()

El método moverse no es accesible.
Sus argumentos eran los siguientes:París

Método privado moverse()

```

El código \$perro->comer(); llama al método público comer(). Pero al método moverse(\$lugar), que es privado, no se puede acceder directamente. Llama al método mágico \_\_call, comprueba que existe el método moverse(\$lugar) y llama al método moverse (\$lugar).

Si el método moverse (\$lugar) es estático, la llamada a la página uso.php es:

```
Animal::moverse("París");
```

Entonces llama al método mágico `__callStatic`.

Hay otros métodos mágicos que no se explican en este libro. Puede obtener más información sobre estos métodos en el manual de PHP, en la siguiente dirección: <http://php.net/manual/es/language.oop5.magic.php>

## Namespaces

Cuando trabaja en proyectos grandes en equipo, es útil modularizar las clases y las funciones. De esta manera, cada desarrollador puede trabajar con su propio módulo. Desde PHP 5.3, los namespaces (espacio de nombres), permiten esta modularización. Un namespace es una especie de carpeta virtual en la que almacena sus objetos. De esta manera es posible usar clases o funciones con el mismo nombre, en namespaces diferentes.

Un namespace se declara con la palabra clave `namespace`, seguido de su nombre, al inicio del archivo.

Por ejemplo:

`espacio_nombres.php`

```
<?php
// Definición del espacio de nombres.
namespace Biblioteca;
// Definición de una constante.
const PI = 3.1416;
// Definición de una función.
function miFuncion() {
echo "Hola <br />";
}
// Definición de una clase.
class miClase {
/*
...
*/
}

?>
```

`Uso_espacio_nombres.php:`

```
<?php
include('espacio_nombres.php');
Biblioteca\miFuncion(); //Llamada al namespace Biblioteca raíz
?>
```

Muestra:

Hola

La constante `__NAMESPACE__` devuelve el nombre del espacio de nombres actual.

Es posible crear sub-espacios de nombres escribiendo:

```
Namespace Espacio1\subespacio1;
```

Las rutas para encontrar una función, clase o constante en un espacio de nombres son relativos si empieza por el namespace o absoluto si empieza con `/`.

Por ejemplo:

Espacio\_nombres.php:

```
<?php
// Definición del espacio de nombres.
namespace Biblioteca;

// Definición de una constante.
const PI = 3.1416;

// Definición de una función.
function miFuncion() {
    echo "Hola <br />";
}

// Definición de una clase.
class Animal
{
    // Declaración de los atributos
    private $color = "gris";
    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escribe en el atributo color
    }
}
?>
```

Uso\_espacio\_nombres.php

```
<?php
namespace Project;
include('espacio_nombres.php');
// Muestra el espacio de nombres actual.
echo 'Espacio de nombres actual = ', __NAMESPACE__, '<br />';
\Biblioteca\miFuncion(); //Llamada al namespace Biblioteca raíz
echo \Biblioteca\PI."<br />";
$gato = new \Biblioteca\Animal();
$gato->setColor("negro");
echo "El color del gato es:".$gato->getColor();

?>
```

Muestra:

Espacio de nombres actual:Project

Hola

3.1416

El color del gato es: negro

Para terminar, puede crear un alias en el espacio de nombres o en un objeto contenido en el espacio de nombres.

Para esto basta con usar el operador `use [namespace] as nombre_nuevo`

Por ejemplo:

```
Use\Biblioteca as biblio;
```

Con los alias, la página `Uso_espacio_nombres.php`, se convierte en:

```
<?php
namespace Project ;
include ('espacio_nombres.php') ;
//Muestra el espacio de nombres actual.
echo 'Espacio de nombres actual = ', _NAMESPACE_<br />;
\Biblioteca\miFuncion() ; llamada al namespace Biblioteca raíz
use\Biblioteca as biblio; // alias de un namespace
echo Biblioteca\PI., "<br />";
use \Biblioteca\Animal as ani; // alias de una clase
$gato = new ani (); // Llamada al alias de la clase
                //Animal$gato->setColor ("negro"),
echo "El color del gato es : ".$gato->getColor();

?>
```

Muestra:

Espacio de nombres actual=Project

Hola

3.1416

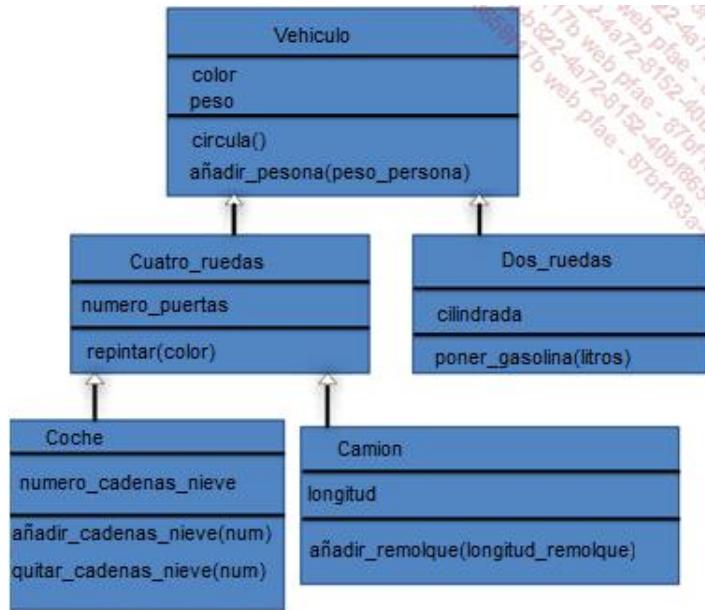
El color del gato es: negro

# Ejercicios

## 1. Enunciados

### **Ejercicio 1 (fácil)**

Cree las cinco clases del esquema teniendo en cuenta su herencia. Todos los métodos son públicos y los atributos son privados.



### **Ejercicio 2: continuación del ejercicio 1 (fácil)**

Cree los accesos de todos los atributos. Cree un constructor en la clase vehículo que tome como argumento el color y el peso. Modifique el método `circula()` para que muestre "El vehículo circula". Modifique el método `añadir_persona(peso_persona)` para que cambie el peso del vehículo en función del peso de la persona que pasa como argumento.

Cree la página mostrar.php y un vehículo negro de 1500 kg. Haga que circule.

Añada una persona de 70 kg y muestre el nuevo peso del vehículo.

### **Ejercicio 3: continuación del ejercicio 2 (dificultad media)**

Aplique el método `repintar(color)` para cambiar el color definido en la clase Vehículo. Ejecute el método `poner_gasolina(litros)` para cambiar el peso definido en la clase Vehículo. En este ejercicio, un litro equivale a un kilo.

Aplique los métodos `añadir_cadenas_nieve()` y `quitar_cadenas_nieve()` modificando el atributo `numero_cadenas_nieve`.

Aplique el método `añadir_remolque(longitud_remolque)` modificando el atributo `longitud`.

En la página mostrar.php, cree un coche verde de 1400 kg. Añada dos personas de 65 kg cada una. Muestre su

color y su nuevo peso.

Retome el coche en rojo y añada dos cadenas para la nieve.

Muestre su color y su número de cadenas para la nieve.

Cree un objeto Dos\_ruedas negro de 120 kg. Añada una persona de 80 kg. Ponga 20 litros de gasolina.

Muestre el color y el peso de dos ruedas.

Cree un camión azul de 10000 kg y de 10 metros de longitud con 2 puertas. Añada un remolque de 5 metros y una persona de 80 kg.

Muestre su color, su peso, su longitud y su número de puertas.

#### **Ejercicio 4: continuación del ejercicio 3 (dificultad media)**

Convierta en abstractos la clase Vehículo y su método añadir\_persona(peso\_persona).

Defina el método añadir\_persona(peso\_persona) en la clase Dos\_ruedas para que este método añada el peso de la persona, más 2 kg del casco del dos ruedas.

Defina el método añadir\_persona(peso\_persona) en la clase Cuatro\_ruedas para hacer lo mismo que en la clase vehículo.

Cree un método público estático en la clase Vehículo que se designe como ver\_atributo.

Este método toma como argumento un objeto y muestra el valor de todos sus atributos (si existen), es decir, el color, el peso, el número de puertas, la cilindrada, la longitud y el número de cadenas para la nieve.

En la página mostrar.php cree un dos ruedas rojo de 150 kg.

Añada una persona de 70 kg y muestre su peso total.

Cambie a verde el color de dos ruedas. Incluya una cilindrada de 1000.

Muestre todos los valores de los atributos del dos ruedas con la función ver\_atributo.

Cree un camión blanco de 6000 kg.

Añada una persona de 84 kg. Vuelva a pintarlo, en color azul. Incluya 2 puertas.

Muestre todos los valores de los atributos del camión con la función ver\_atributo.

#### **Ejercicio 5: continuación del ejercicio 4 (difícil)**

Añada un constructor en la clase Cuatro\_ruedas que tome como argumento el color, el peso y el número de puertas.

Sustituya el método público añadir\_persona(peso\_persona) en la clase Coche. Este método ejecuta el método añadir\_persona(peso\_persona) de la clase Cuatro\_ruedas y da como resultado "Atención, ponga 4 cadenas para la nieve." si el peso total del vehículo es mayor o igual a 1500 kg y si hay 2 cadenas para la nieve o menos.

Añada una constante SALTO\_DE\_LINEA = '<br />' en la clase Vehículo y modifique el método ver\_atributo

`($objeto)` para sustituir los '`<br />`'.

Añada un atributo estático protegido en esta clase que se llame `numero_cambio_color` e inicialice a 0. Este atributo representa el número de veces que va a cambiar el color, sea cual sea el objeto que cambia de color. Este cambio de color se realiza con el método `setColor()`.

Cambie el acceso `setPeso()` de la clase `Vehículo` para que el peso total del coche tenga como máximo 2100 kg.

Cree un coche verde de 2100 kg con 4 puertas en la página `mostrar.php`.

Añada 2 cadenas para la nieve y una persona de 80 kg.

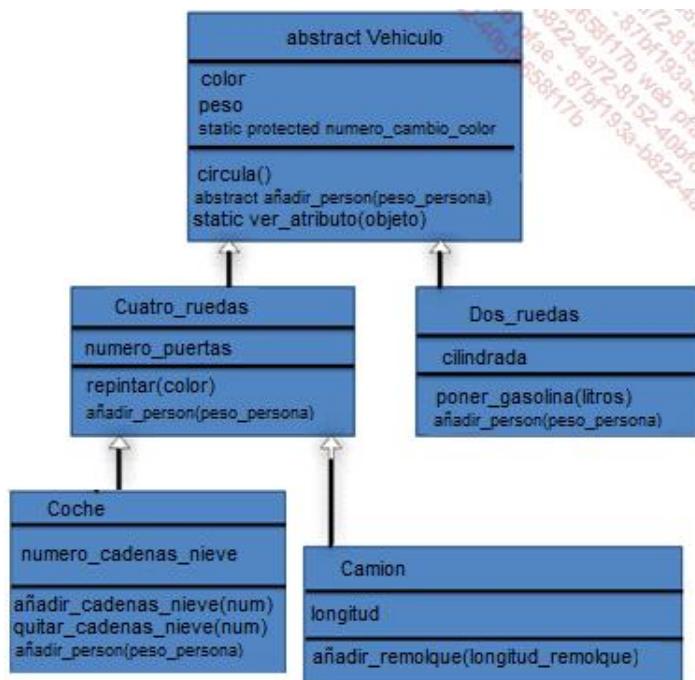
Cambie el color del coche a azul.

Quite 4 cadenas para la nieve.

Vuelva a pintar el coche en color negro.

Muestre todos los atributos del coche y el número de veces que se cambia el color con el método `ver_atributo($objeto)`.

El nuevo modelo es:



Los accesos no se representan.

### **Ejercicio 6 : Blog a la manera de objeto (difícil)**

Al final de este bloque encontrará un ejercicio que utiliza POO para hacer un Blog.

## **2. Soluciones**

## **Solución del ejercicio 1**

Vehículo.class.php:

```
<?php
class vehículo
{
    // Declaración de atributos
    private $color;
    private $peso;

    //métodos públicos
    public function circula()
    {
    }

    public function añadir_persona($peso_persona)
    {
    }

}
?>
```

Cuatro\_ruedas.class.php:

```
<?php
class Cuatro_ruedas extends vehículo
{
    // Declaración de atributos
    private $numero_puertas;

    //método público
    public function repintar($color)
    {

    }

}
?>
```

Dos\_ruedas.class.php:

```
<?php
class Dos_ruedas extends vehículo
{
    // Declaración de atributos
    private $cilindrada;
```

```

//método público
public function poner_gasolina($litros)
{
}

}

?>

```

Coche.class.php:

```

<?php
class Coche extends Cuatro_ruedas
{
    // Declaración de atributos
    private $numero_cadenas_nieve;

    //métodos públicos
    public function añadir_cadenas_nieve($num)
    {

    }

    public function quitar_cadenas_nieve($num)
    {

    }

}
?>

```

Camión.class.php:

```

<?php
class Camion extends Cuatro_ruedas
{
    // Declaración de atributos
    private $longitud;

    //métodos públicos
    public function añadir_remolque($longitud_remolque)
    {

    }

}
?>

```

## **Solución del ejercicio 2**

Vehiculo.class.php:

```
<?php
class vehiculo
{
    // Declaración de atributos
    private $color;
    private $peso;

    //constructor
    public function __construct($color, $peso) // Constructor
//que solicita 2 argumentos.
    {
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.
    }

    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escrito en el atributo color
    }

    public function getPeso()
    {
        return $this->peso; //devuelve el peso
    }
    public function setPeso($peso)
    {
        $this->peso = $peso; //escrito en el atributo peso
    }

    //métodos públicos
    public function circula()
    {
        echo "El vehículo circula.<br />";
    }

    public function añadir_persona($peso_persona)
    {
        $this->peso = $this->peso + $peso_persona;
    }
}
```

Cuatro\_ruedas.class.php:

```
<?php
class Cuatro_ruedas extends vehículo
{
    // Declaración de atributos
    private $numero_puertas;

    //accesos
    public function getNumeroPuertas()
    {
        return $this->numero_puertas; //devuelve el número
    }
    public function setNumeroPuertas($numero_puertas)
    {
        $this->numero_puertas = $numero_puertas; //escrito en
    }
    //el atributo número_puertas
}

//método público
public function repintar($color)
{
}

}

}

}

?>
```

Dos\_ruedas.class.php:

```
<?php
class Dos_ruedas extends vehículo
{
    // Declaración de atributos
    private $cilindrada;

    //accesos
    public function getCilindrada()
    {
        return $this->cilindrada; //devuelve la cilindrada
    }
    public function setCilindrada($cilindrada)
    {
        $this->cilindrada = $cilindrada; //escrito en el atributo
                                         //cilindrada
    }

    //método público
    public function poner_gasolina($numerolitros)
    {
```

```
    }

}

?>
```

#### Coche.class.php:

```
<?php
class Coche extends Cuatro_ruedas
{
    // Declaración de atributos
    private $numero_cadenas_nieve;

    //accesos
    public function getNumeroCadenasNieve()
    {
        return $this->numero_cadenas_nieve; //devuelve el
//numero_cadenas_nieve
    }
    public function setNumeroCadenasNieve($numero_cadenas_nieve)
    {
        $this->numero_cadenas_nieve = $numero_cadenas_nieve;
//escrito en el atributo numero_cadenas_nieve
    }

    //métodos públicos
    public function añadir_cadenas_nieve($numero)
    {

    }

    public function quitar_cadenas_nieve($numero)
    {

    }

}
```

#### Camion.class.php:

```
<?php
class Camion extends Cuatro_ruedas
{
    // Declaración de atributos
    private $longitud;

    //accesos
    public function getLongitud()
```

```

    {
        return $this->longitud; //devuelve la longitud
    }
    public function setLongitud($longitud)
    {
        $this->longitud = $longitud; //escrito en el atributo
                                    //longitud
    }

    //métodos públicos
    public function añadir_remolque($longitud_remolque)
    {

    }

}

?>

```

Mostrar.php:

```

<?php

//carga de clases
include('vehiculo.class.php');

//instanciar la clase vehículo
$vehiculol = new vehiculo("negro",1500);
$vehiculol->circula();
$vehiculol->añadir_persona(70);
echo "El nuevo peso del vehículo es:".$vehiculol->getPeso();

?>

```

Da como resultado:

El vehículo circula.

El nuevo peso del vehículo es:1570

### **Solución del ejercicio 3**

Cuatro\_ruedas.class.php:

```

<?php
class Cuatro_ruedas extends vehiculo
{
    // Declaración de atributos
    private $numero_puertas;

    //accesos

```

```

public function getNumeroPuertas()
{
    return $this->numero_puertas; //devuelve el número
                                   //de puertas
}
public function setNumeroPuertas($numero_puertas)
{
    $this->numero_puertas = $numero_puertas; //escrito en
                                              //el atributo número puertas
}

//Método público
public function repintar($color)
{
    $this->setColor($color);
}

}

?>

```

Dos\_ruedas.class.php:

```

<?php
class Dos_ruedas extends vehiculo
{
    // Declaración de atributos
    private $cilindrada;

    //accesos
    public function getCilindrada()
    {
        return $this->cilindrada; //devuelve la cilindrada
    }
    public function setCilindrada($cilindrada)
    {
        $this->cilindrada = $cilindrada; //escrito en el atributo
                                         //cilindrada
    }

    //método público
    public function poner_gasolina($numerolitros)
    {
        $this->setPeso($this->getPeso() + $numerolitros);
    }

}

?>

```

Coche.class.php:

```

<?php
class Coche extends Cuatro_ruedas
{
    // Declaración de atributos
    private $numero_cadenas_nieve=0;

    //accesos
    public function getNumeroCadenasNieve()
    {
        return $this->numero_cadenas_nieve;
    }
    // devuelve el numero_cadenas_nieve
    public function setNumeroCadenasNieve($numero_cadenas_nieve)
    {
        $this->numero_cadenas_nieve = $numero_cadenas_nieve;
    }
    // escrito en el atributo numero_cadenas_nieve

    //métodos públicos
    public function añadir_cadenas_nieve($numero)
    {
        $this->numero_cadenas_nieve =
        $this->numero_cadenas_nieve + $numero;
    }

    public function quitar_cadenas_nieve($numero)
    {
        if ($this->numero_cadenas_nieve - $numero < 0) {
            $this->numero_cadenas_nieve = 0;
        }
        else {
            $this->numero_cadenas_nieve =
            $this->numero_cadenas_nieve - $numero;
        }
    }
}

?>

```

Camion.class.php:

```

<?php
class Camion extends Cuatro_ruedas
{
    // Declaración de atributos
    private $longitud;

    //accesos
    public function getLongitud()
    {
        return $this->longitud; //devuelve la longitud
    }
}

```

```

public function setLongitud($longitud)
{
    $this->longitud = $longitud; //escrito en el atributo
                                //longitud
}

//métodos públicos
public function añadir_remolque($longitud_remolque)
{
    $this->longitud = $this->longitud + $longitud_remolque;
}

}

?>

```

Mostrar.php:

```

<?php

//carga de clases
include('vehiculo.class.php');
include('Cuatro_ruedas.class.php');
include('Dos_ruedas.class.php');
include('Coche.class.php');
include('Camion.class.php');

//instanciar la clase vehículo
$coche = new Coche("verde",1400);
$coche->añadir_persona(130);
echo "El color del coche es: ".$coche->getColor()."<br />";
echo "El nuevo peso del coche es: ".$coche->getPeso()."<br />";

//repintar en rojo
$coche->repintar("rojo");
//añadir 2 cadenas para la nieve
$coche->añadir_cadenas_nieve(2);
echo "El color del coche es: ".$coche->getColor()."<br />";
echo "El número de cadenas para la nieve del coche es: ".$coche->
getNumeroCadenasNieve()."<br /><br />";

//instanciar los dos ruedas
$dos_ruedas = new Dos_ruedas("negro",120);
$dos_ruedas->añadir_persona(80);
$dos_ruedas->poner_gasolina(20);
echo "El color del dos ruedas es: ".$dos_ruedas->getColor()."<br />";
echo "El peso del dos ruedas es: ".$dos_ruedas->getPeso()."<br />
<br />";

//instanciar el camión
$camion = new Camion("azul",10000);
$camion->setLongitud(10);
$camion->setNumeroPuertas(2);
$camion->añadir_remolque(5);

```

```

$camion->añadir_persona(80);
echo "El color del camión es: ".$camion->getColor()."<br />";
echo "El peso del camión es: ".$camion->getPeso()."<br />";
echo "La longitud del camión es: ".$camion->getLongitud()."<br />";
echo "El número de puertas del camión es: ".$camion->getNumeroPuertas()."<br />";
?>

```

Da como resultado:

```

El color del coche es: verde
El nuevo peso del coche es: 1530
El color del coche es: rojo
El número de cadenas para la nieve del coche es: 2
El color del dos ruedas es: negro
El peso del dos ruedas es: 220
El color del camión es: azul
El peso del camión es: 10080
La longitud del camión es: 15
El número de puertas del camión es: 2

```

#### **Solución del ejercicio 4**

Vehículo.class.php:

```

<?php
abstract class vehiculo
{
    // Declaración de atributos
    private $color;
    private $peso;

    //constructor
    public function __construct($color, $peso)
//Constructor que solicita 2 argumentos.
    {
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.
    }

    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escrito en el atributo color
    }
}

```

```

}

public function getPeso()
{
    return $this->peso; //devuelve el peso
}
public function setPeso($peso)
{
    $this->peso = $peso; //escrito en el atributo peso
}

//métodos públicos
public function circula()
{
    echo "El vehículo circula.<br />";
}

public static function ver_atributo($objeto) {
    if(method_exists($objeto,'getColor')) {
        echo "El color es:".$objeto->getColor()."<br />";
    }
    if(method_exists($objeto,'getPeso')) {
        echo "El peso es:".$objeto->getPeso()."<br />";
    }
    if(method_exists($objeto,'getCilindrada')) {
        echo "La cilindrada es:".$objeto->getCilindrada().
"<br />";
    }
    if(method_exists($objeto,'getNumeroPuertas')) {
        echo "El número de puertas es:";
        echo $objeto->getNumeroPuertas()."<br />";
    }
    if(method_exists($objeto,'getLongitud')) {
        echo "La longitud es:".$objeto->getLongitud();
        echo "<br />";
    }
    if(method_exists($objeto,'getNumeroNeumaticoNieve')) {
        echo "El número de cadenas para la nieve es:";
        echo $objeto->getNumeroNeumaticoNieve()."<br />";
    }
}
}

abstract public function añadir_persona($peso_persona);

}

?>

```

Cuatro\_ruedas.class.php:

```

<?php
class Cuatro_ruedas extends vehiculo
{

```

```

// Declaración de atributos
private $numero_puertas;

//accesos
public function getNumeroPuertas()
{
    return $this->numero_puertas; //devuelve el número
                                    //de puertas
}
public function setNumeroPuertas($numero_puertas)
{
    $this->numero_puertas = $numero_puertas; //escrito en
                                              //el atributo número_puertas
}

//métodos públicos
public function retomar($color)
{
    $this->setColor($color);
}

public function añadir_persona($peso_persona)
{
    $this->setPeso($this->getPeso() + $peso_persona);
}

}

?>

```

Dos\_ruedas.class.php:

```

<?php
class Dos_ruedas extends vehiculo
{
    // Declaración de atributos
    private $cilindrada;

    //accesos
    public function getCilindrada()
    {
        return $this->cilindrada; //devuelve la cilindrada
    }
    public function setCilindrada($cilindrada)
    {
        $this->cilindrada = $cilindrada; //escrito en el
                                         //atributo cilindrada
    }

    //método público
    public function poner_gasolina($numerolitros)
    {
        $this->setPeso($this->getPeso()+$numerolitros);
    }
}

```

```

        }

        public function añadir_persona($peso_persona)
        {
            $this->setPeso($this->getPeso() + $peso_persona + 2);
        }

    }

?>

```

Las otras clases no cambian.

Mostrar.php:

```

<?php

//carga de clases
include('vehiculo.class.php');
include('Cuatro_ruedas.class.php');
include('Dos_ruedas.class.php');
include('Camion.class.php');

//instanciar los dos ruedas
$dos_ruedas = new Dos_ruedas("rojo",150);
$dos_ruedas->añadir_persona(70);
echo "El peso de dos ruedas es: ".$dos_ruedas->getPeso()."<br />";
$dos_ruedas->setColor("verde");
$dos_ruedas->setCilindrada(1000);
vehiculo::ver_atributo($dos_ruedas);
echo "<br />";

//instanciar el camión
$camion = new Camion("blanco",6000);
$camion->añadir_persona(84);
$camion->setColor("azul");
$camion->setNumeroPuertas(2);
vehiculo::ver_atributo($camion );

?>

```

Da como resultado:

El peso de dos ruedas es:222

El color es:verde

El peso es:222

La cilindrada es:1000

El color es:azul

El peso es:6084

El número de puertas es:2

La longitud es:

## Solución del ejercicio 5

Vehiculo.class.php:

```
<?php
abstract class vehiculo
{
    // Declaración de atributos
    private $color;
    private $peso;
    const SALTO_DE_LINEA = '<br />';
    static protected $numero_cambio_color=0;

    //constructor
    public function __construct($color, $peso)
//Constructor que solicita 2 argumentos.
    {
        $this->color = $color; // Inicialización del color.
        $this->peso = $peso; // Inicialización del peso.
    }

    //accesos
    public function getColor()
    {
        return $this->color; //devuelve el color
    }
    public function setColor($color)
    {
        $this->color = $color; //escrito en el atributo color
        self::$numero_cambio_color =
self::$numero_cambio_color + 1;
    }

    public function getPeso()
    {
        return $this->peso; //devuelve el peso
    }
    public function setPeso($peso)
    {
        if ($peso > 2100) {
            $this->peso = 2100; //escrito en el atributo peso
        }
        else {
            $this->peso = $peso; //escrito en el atributo peso
        }
    }

    //métodos públicos
    public function circula()
    {
        echo "El vehículo circula.<br />";
    }
}
```

```

}

public static function ver_atributo($objeto) {
    if(method_exists($objeto,'getColor')) {
        echo "El color es:" ;
        echo $objeto->getColor().self::SALTO_DE_LINEA;
    }
    if(method_exists($objeto,'getPeso')) {
        echo "El peso es:" ;
        echo $objeto->getPeso().self::SALTO_DE_LINEA;
    }
    if(method_exists($objeto,'getCilindrada')) {
        echo "La cilindrada es:" ;
        echo $objeto->getCilindrada().self::SALTO_DE_LINEA;
    }
    if(method_exists($objeto,'getPuerta')) {
        echo "El número de puertas es:" ;
        echo $objeto->getNumeroPuertas().self::SALTO_DE_LINEA;
    }
    if(method_exists($objeto,'getLongitud')) {
        echo "La longitud es:" ;
        echo $objeto->getLongitud().self::SALTO_DE_LINEA;
    }
    if(method_exists($objeto,'getNumeroNeumaticoNieve')) {
        echo "El número de cadenas para la nieve es:" ;
        echo $objeto->getNumeroNeumaticoNieve().self::SALTO_DE_LINEA;
    }
    echo "El color se ha cambiado";
    echo self::$numero_cambio_color." vez.";
}

abstract public function añadir_persona($peso_persona);

}
?>
```

Cuatro\_ruedas.class.php:

```

<?php
class Cuatro_ruedas extends vehiculo
{
    // Declaración de atributos
    private $numero_puertas;

    //constructor
    public function __construct($color, $peso, $numero_puertas)
//Constructor que solicita 3 argumentos.
    {
        $this->setColor($color); // Inicialización del color.
        $this->setPeso($peso); // Inicialización del peso.
        $this->numero_puertas = $numero_puertas; // Inicialización
                                                //del número de puertas.
```

```

    }

    //accesos
    public function getNumeroPuertas()
    {
        return $this->numero_puertas; //devuelve el número de puertas
    }
    public function setNumeroPuertas($numero_puertas)
    {
        $this->numero_puertas = $numero_puertas; //escrito en
                                                //el atributo número_puertas
    }

    //métodos públicos
    public function retomar($color)
    {
        $this->setColor($color);
    }

    public function añadir_persona($peso_persona)
    {
        $this->setPeso($this->getPeso() + $peso_persona);
    }

}

?>

```

Coche.class.php:

```

<?php
class Coche extends Cuatro_ruedas
{
    // Declaración de atributos
    private $numero_cadenas_nieve=0;

    //accesos
    public function getNumeroCadenasNieve()
    {
        return $this->numero_cadenas_nieve; //devuelve
                                            //el numero_cadenas_nieve
    }
    public function setNumeroCadenasNieve
    ($numero_cadenas_nieve)
    {
        $this->numero_cadenas_nieve = $numero_cadenas_nieve;
//escrito en el atributo numero_cadenas_nieve
    }

    //métodos públicos
    public function añadir_cadenas_nieve($numero)
    {
        $this->numero_cadenas_nieve =

```

```

$this->numero_cadenas_nieve + $numero;
}

public function quitar_cadenas_nieve($numero)
{
    if ($this->numero_cadenas_nieve - $numero < 0) {
        $this->numero_cadenas_nieve = 0;
    }
    else {
        $this->numero_cadenas_nieve =
        $this->numero_cadenas_nieve - $numero;
    }
}

public function añadir_persona($peso_persona)
{
    parent::añadir_persona($peso_persona);
    if ($this->getPeso() >= 1500 &&
$this->numero_cadenas_nieve <= 2)
    {
        echo "Atención, ponga 4 cadenas para la nieve.";
        echo "<br />";
    }
}

}

?>

```

Mostrar.php:

```

<?php

//carga de clases
include('vehiculo.class.php');
include('Cuatro_ruedas.class.php');
include('Coche.class.php');

//instanciar el coche
$coche = new Coche("verde",2100,4);
$coche->añadir_cadenas_nieve(2);
$coche->añadir_persona(80);
$coche->setColor("azul");
$coche->quitar_cadenas_nieve(4);
$coche->setColor("negro");
vehiculo::ver_atributo($coche);

?>

```

Da como resultado:

Atención, ponga 4 cadenas para la nieve.

El color es:negro

El peso es:2100

El número de puertas es:4

El número de cadenas para la nieve es:0

El color se ha cambiado 3 veces.

## PHP.ini

Este archivo contiene una serie de directivas que se pueden activar y que influyen en el comportamiento de PHP. Por ejemplo, define las carpetas, añade librerías PHP, cambia los parámetros de MySQL, etc.

Se accede a través del menú **Configuración - PHP** de EasyPHP.

Se ubica por defecto en la carpeta C:\Program Files (x86)\EasyPHP-DevServer-13.1VC11\binarios\conf\_files, aunque se puede ubicar en otro lugar. También se puede encontrar en Windows, en la carpeta C:\Windows, en Linux, en la carpeta /usr/local/lib, o en la carpeta definida por la variable de entorno PHPRC.

En este libro no se explican todas las directivas, tan solo las más usadas. Atención, los valores que cambie en el archivo de configuración PHP.ini en local, no son forzosamente las mismas que las de vuestro host y su sitio Web corre el riesgo de no volver a funcionar cuando vuelva a subirlas a su proveedor. Por tanto debe asegurarse de que los valores del host no son incompatibles con los definidos en su PHP.ini en local.

Cuando hay un punto y coma delante de una línea en el archivo, quiere decir que esta línea está comentada. Para tener en cuenta esta directiva, debe quitar el punto y coma, cambiar el valor de la directiva, guardar la carpeta y reiniciar el servidor Web.

Las directivas más importantes son:

- **asp\_tags**: permite añadir código PHP a los tags <% %> (ver capítulo Las bases del lenguaje PHP). Esta directiva está por defecto en modo *off*.
- **display\_errors**: permite mostrar los errores de PHP. Esta directiva está por defecto en modo *on*.
- **error\_reporting**: permite mostrar los tipos de errores. Esta directiva por defecto tiene el valor E\_ALL|E\_STRICT, y muestra todos los errores, avisos y errores de sintaxis. Es recomendable cambiar este valor por E\_ALL&~E\_DEPRECATED, que ya no muestra los errores de sintaxis.
- **include\_path**: permite definir el directorio que contiene los archivos include (ver capítulo Funciones y estructuras de control - Los includes). Cuando utiliza la función `include()` o `require()`, PHP busca automáticamente en esta carpeta.
- **variables\_order**: permite definir las tablas superglobales que PHP tiene en cuenta. Esta directiva es GPCS (*Get Post Cookie and Session*). Para probar todos los capítulos y sobre todo las variables de entorno, debe poner EGPCS, E en Entorno.
- **display\_startup\_errors**: permite ver los errores cuando arranca el servidor web. Por defecto, esta directiva está en *on* pero es aconsejable ponerla en *off* en modo producción.

## Archivo upload

- **upload\_max\_filesize**: permite definir el tamaño máximo del archivo que se va a enviar. Este valor está en 2 GB por defecto (ver capítulo Transmitir datos de una página a otra).
- **file\_uploads**: permite autorizar el envío del archivo. Esta directiva está en modo *on* por defecto.
- **upload\_tmp\_dir**: permite definir el directorio temporal que va a almacenar los archivos que se van a transmitir.
- **post\_max\_size**: permite definir el tamaño máximo de los datos que envía el formulario (imagen + texto). Esta directiva debe ser siempre superior a `upload_max_filesize`.

## Activar las librerías

Para activar las librerías, como por ejemplo la librería GD o PDO, debe quitar el punto y coma que está delante de la

línea relativa a esta librería (por ejemplo, extensión=php\_gd2.dll). A continuación, reinicie el servidor Web (menú **Reiniciar**).

## Archivo de configuración MySQL: My.ini

Este archivo contiene las directivas que hay que activar sin que pueda afectar a MySQL. Proporciona los mejores parámetros de ajuste de MySQL para aumentar el rendimiento y la seguridad.

Se accede a través del menú **Configuración - MySQL**.

Se ubica en la carpeta C:\Program Files (x86)\EasyPHP-DevServer-13.1VC11\ binarios\conf\_files, pero también se puede llamar My.cnf.

También puede ubicar este archivo en la carpeta C:\Windows, o en la carpeta /etc en Unix.

Las líneas precedidas de almohadilla son comentarios. Para actualizar una directiva, debe quitar la #, modificar el valor, guardar el archivo y reiniciar el servidor MySQL.

El administrador de la base de datos puede cambiar las directivas, pero hay que tener en cuenta que, para poder conectarse a MySQL, el puerto y la contraseña están en el área Cliente.

Puede introducir la directiva skip-networking si nadie se conecta a distancia a la directiva MySQL, que será solo accesible en modo local.

## Archivo de configuración Apache: Httpd.conf

Es el archivo de configuración del servidor Web Apache. Contiene directivas que pueden influir en el funcionamiento del servidor Web. Se recomiendan estas directivas, aunque puede activarlas o desactivarlas añadiendo o eliminando la # delante de la línea.

Este archivo se ubica por defecto en la carpeta C:\Program Files (x86)\EasyPHP-DevServer-13.1VC11\binarios\conf\_files y se puede acceder a él a través del menú **Configuración - Apache**.

Debe reiniciar el servidor Web para que se puedan aplicar posibles modificaciones.

Estas directivas permiten cambiar los módulos o la seguridad de algunos directorios de su sitio Web.

Este archivo contiene una explicación exhaustiva de las directivas, y va dirigido a webmasters y administradores de sitios Web. Las directivas más importantes son:

- DocumentRoot: directorio de base del sitio Web ("/data/localweb" por defecto, ubicado en el directorio de instalación de EasyPHP).
- Listen: establece el puerto Apache o la dirección que atiende las peticiones de los clientes (por defecto, 80).
- ServerRoot: directorio de instalación de Apache.
- ErrorLog: ruta del archivo que define una lista de errores surgidos por el uso de Apache.
- LogLevel: define el nivel en el que los errores detectados se almacenan en el archivo "log".
- DirectoryIndex: si el cliente no indica un archivo concreto, Apache busca por orden en el directorio los archivos específicos (index.html index.php...).

También puede modificar la configuración de un sitio web o de algunas carpetas si añade un archivo .htaccess. Puede prohibir listar el contenido de una carpeta, administrar los errores 404, volver a escribir las direcciones (URL rewriting), etc.

Por ejemplo, si quiere prohibir el contenido de un directorio, escriba en el archivo .htaccess del directorio lo siguiente:

```
Options - Indices
```

# Fallos de seguridad XSS

Los fallos de seguridad XSS o Cross Site Scripting consisten en la ejecución del código HTML, JavaScript o VBScript sin conocimiento del webmaster.

Hay dos tipos de XSS.

## 1. XSS no permanente

Este tipo de fallos de seguridad se puede encontrar cuando recurre a una página PHP con información que se pasa en la URL.

Por ejemplo, cuando recurre a la página get\_recibe.php con el parámetro nombre=Juan:

`http://127.0.0.1:8888/get_recibe.php?nombre=Juan`

En la página get\_recibe.php, aparece el siguiente código:

```
<?php  
    echo $_GET['nombre'];  
?>
```

Da como resultado:

Juan

A continuación el usuario cambia Juan por:

```
<script>alert('Hola')</script>
```

La página get\_recibe abre un pop-up que muestra "Hola". El navegador ejecuta JavaScript que se ha generado con la instrucción `echo $_GET['nombre'];`

En este caso, el código intruso es una alerta, pero el usuario puede introducir un código para leer las cookies, redirigir a otra URL, etc.

Para impedir esto, los navegadores suelen activar la seguridad XSS por defecto, que procede de un script que se encuentra en la URL y que por tanto no se ejecuta.

Sin embargo, no puede saber con antelación la configuración del navegador del usuario, por lo que se recomienda añadir código PHP para impedir estos scripts.

Utilice la función `htmlentities()` cuando recupere la información con `$_GET` o `$_POST`. Esta función traduce los caracteres especiales y hace inoperativos los scripts maliciosos. Por defecto convierte las dobles comillas e ignora las simples. Para utilizar las comillas simples, añada la constante `ENT_QUOTES`.

El ejemplo anterior se convierte en:

```
<?php
```

```
echo htmlentities($_GET['nombre'], ENT_QUOTES);
?>
```

Y da como resultado:

```
<script>alert('Hola')</script>
```

Si muestra el código de origen de la página, aparecerá lo siguiente:

```
&lt;script&gt;alert(Hola)&lt;/script&gt;
```

Y el navegador muestra el script sin ejecutarlo.

## 2. XSS permanente

Supongamos que el script malicioso se ha insertado en la base de datos con un mensaje en el foro, el blog o en un e-mail. Este mensaje se ejecuta cada vez que una persona lee el mensaje en Internet.

La solución es la misma: si recupera el mensaje con `$_POST['mensaje']`, añada la función `htmlentities()`.

Por ejemplo, si una página recupera el nombre, el apellido y el mensaje con POST, dará el siguiente resultado:

```
<?php
echo htmlentities($_POST['nombre'], ENT_QUOTES);
echo htmlentities($_POST['apellido'], ENT_QUOTES);
echo htmlentities($_POST['mensaje'], ENT_QUOTES);
?>
```

Los caracteres especiales se insertan en la base de datos y el script no se ejecuta cuando se muestra el mensaje que lo contiene.

## 3. Error de página

Si recurre a su página PHP utilizando como parámetro el nombre de una página contenida en un include, aparecerá lo siguiente: `http://127.0.0.1:8888/get_recibe.php?pagina=pagina1`

El código pagina get\_recibe.php es:

```
<?php
include("files/".htmlentities($_GET['pagina']).".php");
?>
```

Tiene como efecto incluir pagina1.php

Si el usuario agrega en la URL un script malicioso, como: `http://127.0.0.1:8888/get_recibe.php?pagina=<script>`

```
alert('Hola')</script>
```

Aparecerá en su página un mensaje de error, con un e-mail que contiene la URL que provoca dicho error, siempre y cuando se haya activado el informe de error.

Cuando lea su e-mail, se ejecutará el script, ya que tendrá su código JavaScript en el e-mail.

Para evitar esto, añada el código `error_reporting(0)` en la página de inicio. El resultado será no mostrar el error y no enviará el e-mail de la URL que contiene el script malicioso.

# Derechos de la base de datos

Cuando se conecta a la base de datos en modo "root", tiene todos los derechos sobre ella. Y si se conecta a una página PHP en modo root, puede ser víctima de una inyección SQL que puede eliminar datos o incluso una tabla completa.

Para evitarlo, cree en su base de datos unos usuarios que tendrán derechos limitados y conéctese a través de estos en su código PHP. De esta manera, haga lo que haga el usuario, nunca podrá eliminar los datos.

Para crear un usuario, seleccione su base de datos en PHP-MyAdmin y haga clic en la pestaña **Privilegios**. Con esto se trata de agregar un usuario que tenga solamente derechos en modo de lectura.

Haga clic en el enlace **Agregar un nuevo usuario**. Introduzca el nombre "EstefaniaMorales" en el nombre y "123" en la contraseña. A continuación, compruebe los datos con la opción **SELECT**.

The screenshot shows the 'Agregar un nuevo usuario' (Add new user) form in PHP-MyAdmin. The 'Información de la cuenta' (Account information) section includes fields for 'Nombre de usuario' (username) set to 'EstefaniaMorales', 'Contraseña' (password) set to '123', and 'Base de datos para el usuario' (Database for user) set to 'Ninguna'. The 'Privilegios globales' (Global privileges) section has 'SELECT' checked under 'Datos' (Data). The 'Limites de recursos' (Resource limits) section has 'MAX QUERIES PER HOUR' set to 0. At the bottom right is a 'Continuar' (Continue) button.

Haga clic en **Continuar**.

The screenshot shows the 'Vista global de usuarios' (Global view of users) page in PHP-MyAdmin. It lists users: AdministradorUni, EstefaniaMorales, LunaLunera, and root. The root user has 'ALL PRIVILEGES' assigned. A link to 'Agregar un nuevo usuario' (Add new user) is at the bottom.

A continuación, cuando se conecte a su base de datos en una página PHP en modo de lectura, use la cuenta "EstefaniaMorales".

La cadena de conexión no es la siguiente:

```
<?php  
// Conexión a la base de datos  
$base = mysqli_connect("127.0.0.1", "root", "", "_prueba");  
?>
```

Sino:

```
<?php  
// Conexión a la base de datos solo en modo de lectura  
$base = mysqli_connect("127.0.0.1", "EstefaniaMorales", "123", "_prueba");  
?>
```

## Inyección SQL (addslashes)

La inyección SQL tiene como objetivo injectar un código SQL que ha escrito un usuario malicioso. Este código permite iniciar la sesión sin conocer necesariamente la contraseña, mostrar todas las contraseñas e incluso destruir la tabla por completo.

En el siguiente ejemplo, la directiva Magic quotes está en Off.

El usuario utiliza un formulario que le solicita un identificador y una contraseña. Cuando hace clic en validar, puede comprobar que la persona tiene un identificador y una contraseña correcta con una consulta de tipo:

```
<?php  
$login = $_POST['login'];  
$password = $_POST['contraseña'];  
$sql = "SELECT * FROM Persona WHERE login='".$login."'  
and password='".$password."'";  
?>
```

Si la consulta devuelve una fila, significa que la persona existe en la base de datos y que, por tanto, la puede conectar.

Si el usuario desconoce el identificador y la contraseña, pero introduce en el campo del identificador 'pepe' y en el campo de la contraseña:

```
' OR 1=1 OR '
```

La consulta SQL se convierte en:

```
$sql = "SELECT * FROM Persona WHERE login='pepe' and password=''  
OR 1=1 OR ''";
```

El código OR 1=1 es correcto; entonces la consulta devuelve todos los registros de la tabla Persona. Así puede pensar que la persona existe realmente en la base de datos.

Si quiere evitarlo, añada la función addslashes( ) cuando recupere los datos con POST, y no con GET. Esta función agrega el carácter \ (barra invertida) delante de ' (apóstrofos), " (comillas), \ (barra invertida) y NULL.

El código queda protegido y se convierte en:

```
<?php  
$login = addslashes($_POST['login']);  
$password = addslashes($_POST['contraseña']);  
$sql = "SELECT * FROM Persona WHERE login='".$login."'  
and password='".$password."'";  
?>
```

De este modo, la consulta SQL se convierte en:

```
SELECT * FROM Persona WHERE login=' pepe' and password = '\' OR 1=1 OR \"
```

El código OR 1=1 se interpreta como una cadena de caracteres; por lo tanto, ya no se ejecuta.

Este método no es seguro al cien por cien, porque en algunas consultas se puede eludir esta protección.

Para estar totalmente seguro, se recomienda utilizar las consultas que se han explicado en capítulos anteriores.

## Comprobación de la sesión

La sesión permite almacenar un objeto en la memoria mientras el usuario esté conectado a su sitio Web. Es el caso, por ejemplo, de un sitio Web que contiene una página que permite iniciar sesión antes de acceder a otras páginas de su sitio Web, que son: pagina1.php, pagina2.php...

Compruebe en su página de inicio que la persona existe en la base de datos antes de dirigirse a la pagina1.php. Nada impide que la persona escriba directamente <http://www.misitio.es/pagina1.php> para acceder de forma inmediata a esta página sin tener que identificarse. Esta persona debe conocer el nombre de la página PHP, pero algunas veces los nombres de estas páginas son muy concretos: login.php, mostrar.php, forum.php.

Para solucionar este problema, utilice las sesiones. Cuando una persona se identifique, almacene en la sesión su identificador y compruebe que la sesión se reconoce en cada página de su sitio Web.

De este modo, se mostrará en todas las páginas PHP de su sitio Web (salvo la página de conexión, que utiliza el identificador de la sesión) lo siguiente:

```
<?php
session_start();
if (!isset($_SESSION['login'])) {
    //redireccionar a la página de conexión
    header("Location:conexion.php");
}
?>
```

# Rendimiento

## 1. Optimizar el rendimiento en PHP

A continuación se muestran algunos puntos que se pueden aplicar, que harán ganar tiempo durante la ejecución del código. Cada truco aplicado de manera independiente no aumenta significativamente el rendimiento, pero todos juntos y ejecutados miles de veces, tendrán un impacto importante.

- Declare un método "static" si es posible: el rendimiento se multiplica por 4.
- No use funciones dentro de su bucle: `for ($i=0;$i<sizeof($array);$i++)->` el método `sizeof` se llama en cada interacción.
- Cuando recupere los datos con la función `mysql_fetch_assoc`, `$registro['id']`, es siete veces más rápido que `$registro[id]`.
- Evite los métodos mágicos.
- Evite las expresiones regulares y use funciones como `stripos()`, `strcasecmp()`, `str_replace()`...
- Use los módulos que permiten su cacheo como `cache_Lite` de PEAR o `memcached`.
- Use `require()` en lugar de `require_once()`.
- La función `echo` es más rápida usando apóstrofes en lugar de comillas.
- Evite las variables globales: estas variables permanecen en memoria todo el tiempo de ejecución de la página PHP.
- Evite el uso de variables inútiles:  
`echo htmlentities($_POST['nombre']);` es más rápido que  
`$nombre=htmlentities($_POST['nombre']);`  
`echo $nombre;`
- Use `==` en lugar de `=`. Esto puede ser hasta doce veces más rápido.
- Actualice su versión de PHP si alberga su sitio web.

## 2. Optimizar el rendimiento en MySQL

- Use un sistema de caché como QueryCache.
- Indique los campos que se usan en las Joins y las cláusulas where.
- Use consultas preparadas, sobretodo si su consulta está en su bucle. En este caso es mejor poner la consulta en un procedimiento almacenado que la ejecutará varias veces. Así tendrá un único acceso a la base de datos desde la página PHP.
- La optimización de la base de datos es una tarea más propia de un administrador de base de datos. Encontrará más información en el siguiente enlace:  
<http://dev.mysql.com/doc/refman/5.0/en/optimization.html>

## Crear un blog (procedimiento)

- Cree un formulario para añadir el contenido al blog:

The screenshot shows a web browser window with the following details:

- Title bar: Blog, 127.0.0.1, https://w...
- Address bar: 127.0.0.1:8888/prueba.php
- Content area:
  - Formulario para añadir contenido al Blog**
  - Título:
  - Comentario:
  - Selección de foto: Seleccione una foto con un tamaño inferior a 2 MB.  
[Selezionare file] No se ha seleccionado ningún archivo
  - Enviar
  - [Página de visualización del blog](#)

- Cree una página PHP para añadir el contenido a la base de datos y copie a la carpeta de imágenes la imagen que se ha transmitido. Esta página mostrará lo siguiente:

Ningún error en la transferencia del archivo.  
El archivo Desierto.jpg se ha copiado en el directorio de fotos.  
Se ha añadido el comentario correctamente

[Volver a la página de inserción](#)

Cree una página PHP que muestre los distintos contenidos que se han añadido al blog.

# Blog

## Bienvenido

El 14/04/2013 00:00:00

Bienvenido a mi Blog



[Volver a la página de inserción](#)

## Solución

- Cree una carpeta de imágenes en el mismo lugar que sus php.
- Cree la base de datos Blog.

- El script SQL para crear el contenido de la tabla es:

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
DROP TABLE IF EXISTS contenido;
CREATE TABLE contenido (
    Id int(11) NOT NULL AUTO_INCREMENT,
    Titulo varchar(25) NOT NULL,
    Fecha Datetime NOT NULL,
    Comentario text NOT NULL,
    Imagen varchar(25) NOT NULL,
    PRIMARY KEY (Id)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;
```

- La página del formulario\_anadir.php es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Blog</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <h2>Formulario para añadir el contenido al Blog</h2>
        <form action="insertar_contenido.php" method="POST"
enctype="multipart/form-data">
            <p>Título: <input type="text" name="titulo" /></p>
```

```

<p>Comentario: <br /><textarea name="comentario" rows="10"
cols="50"></textarea></p>
<input type="hidden" name="MAX_FILE_SIZE" value="2097152">
<p>Seleccione una foto con un tamaño inferior a 2 MB.</p>
<input type="file" name="foto">
<br /><br />
<input type="submit" name="ok" value="Enviar">
</form>
<br />
<a href="mostrar_blog.php" >Página de visualización del blog</a>
</body>
</html>

```

- La página insertar\_contenido.php es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<title>Blog</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
</head>
<body>
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "blog");

/* Comprobar la conexión*/
if (!$connect) {
    echo "fallo de conexión : ".mysqli_connect_error();
    exit();
}

if ($_FILES['imagen']['error']) {
    switch ($_FILES['imagen']['error']){
        case 1: // UPLOAD_ERR_INI_SIZE
            echo "El tamaño del archivo supera el límite
permitido por el servidor (parámetro upload_max_filesize del archivo
php.ini).";
            break;
        case 2: // UPLOAD_ERR_FORM_SIZE
            echo "El tamaño del archivo supera el límite permitido
por el formulario (parámetro post_max_size del archivo php.ini).";
            break;
        case 3: // UPLOAD_ERR_PARTIAL
            echo "El envío del archivo se ha interrumpido durante
la transmisión.";

            break;
        case 4: // UPLOAD_ERR_NO_FILE
            echo "El tamaño del archivo que ha enviado es nulo.";
            break;
    }
}

```

```

else {
    //si no hay ningún error, entonces $_FILES['nombre
    del_archivo']['error']
    //vale 0
    echo "Ningún error en la transferencia del archivo.<br />";
    if ((isset($_FILES['imagen']['name'])&&($_FILES['imagen']['error'] ==
    UPLOAD_ERR_OK)) {
        $destino_de_ruta = 'imágenes/';
        //desplazamiento del archivo del directorio temporal (almacenado
        //por defecto) al directorio de destino
        move_uploaded_file($_FILES['imagen']['tmp_name'],
$destino_de_ruta.$_FILES['imagen']['name']);
        echo "El archivo ".$_FILES['imagen']['name']."' Se ha copiado en el
directorio de fotos";
    }
    else {
        echo "El archivo no se ha copiado en el directorio de fotos.";
    }
}

$consulta = "INSERT INTO contenido (Titulo, Fecha, Comentario, Imagen)
VALUES ('".htmlentities(addslashes($_POST['titulo']),
ENT_QUOTES)."' ,'" .date("Y-m-d H:i:s")."' ,'" .htmlentities
(addslashes($_POST['comentario']), ENT_QUOTES).',
'" .$_FILES['imagen']['name']."' )";
$resultado = mysqli_query($connect,$consulta);
$iniciar_sesion = mysqli_insert_id($connect);
/* Cierre de la conexión */
mysqli_close($connect);

if ($iniciar_sesion != 0) {
    echo "<br />La adición del comentario se ha hecho con éxito.<br /><br />";
}
else {
    echo "<br />El comentario no se ha añadido.<br /><br />";
}
?>
<a href="formulario_anadir.php" >volver a la página de añadir</a>
</body>
</html>

```

- La página mostrar\_blog.php es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Blog</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <h2>Blog</h2>
        <hr />
        <?php

```

```

$connect = mysqli_connect("127.0.0.1", "root", "", "blog");

/* Comprobar la conexión */
if (!$connect) {
    echo "fallo de la conexión : ".mysqli_connect_error();
    exit();
}

$consulta = "SELECT * FROM contenido ORDER BY Date";
if ($resultado = mysqli_query($connect,$consulta)) {
    Date_default_timezone_set('Europe/Paris');
    /* busca la matriz asociativa */
    while ($registro = mysqli_fetch_assoc($resultado)) {
        $dt_inicio = Date_create_from_format('Y-m-d H:i:s',
$registro['Fecha']);
        echo "<h3>".$registro['Titulo']."</h3>";
        echo "<h4>El ".$dt_inicio->formato('d/m/Y H:i:s')."</h4>";
        echo "<div style='width:400px'>".$registro['Comentario']."</div>";
        if ($registro['Imagen'] != "") {
            echo "<img src='imagenes/".$registro['Imagen']."' width='200px' height='200px' />";
        }
        echo "<hr />";
    }
}
?>
<br />
<a href="formulario_anadir.php" >Volver a la página de inserción</a>
</body>
</html>

```

## Crear un blog (objeto)

Vuelva a repetir el ejercicio A, pero ahora utilice objetos.

Cree una clase Blog cuyos atributos coincidan con los campos de la tabla Contenido. Cree una clase Manager que tenga la conexión PDO como atributo, así como unos métodos que permitan leer y escribir en la tabla Contenido. Las páginas mostrar\_blog.php, insertar\_contenido.php y añadir\_contenido.php no deben contener consultas SQL.

### Solución

#### → Cree la base de datos Blog

- El script SQL que crea la tabla Contenido es:

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
-- 
-- Base de datos : `Blog`
-- 
-- -----
-- 
-- Estructura de la tabla `contenido`
-- 
DROP TABLE IF EXISTS `contenido`;
CREATE TABLE IF NOT EXISTS `contenido` (
  `Id` int(11) NOT NULL AUTO_INCREMENT,
  `Titulo` varchar(25) NOT NULL,
  `Fecha` Datetime NOT NULL,
  `Comentario` text NOT NULL,
  `Imagen` varchar(25) NOT NULL,
  PRIMARY KEY (`Id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;
```

- La página formulario\_añadir.php es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>Blog</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <h2>Formulario para añadir el contenido al Blog</h2>
    <form action="insertar_contenido.php" method="POST"
      enctype="multipart/form-data">
      <p>Título: <input type="text" name="titulo" /></p>
      <p>Comentario: <br /><textarea name="comentario" rows="10"
cols="50"></textarea></p>
```

```

<input type="hidden" name="MAX_FILE_SIZE" value="2097152">
<p>Seleccione una foto que tenga un tamaño inferior a 2 MB.
</p>
<input type="file" name="imagen">
<br /><br />
<input type="submit" name="ok" value="Enviar">
</form>
<br />
<a href="mostrar_blog.php" >Página de visualización del blog
</a>
</body>
</html>

```

- La página insertar\_contenido.php es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Blog</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
<?php
include('Blog.class.php');
include('Manager.class.php');
try
{
    // Conexión a la base de datos
    $base = new PDO('mysql:host=127.0.0.1;dbname=Blog', 'root', '');
    $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

    if ($_FILES['imagen']['error']) {
        switch ($_FILES['imagen']['error']){
            case 1: // UPLOAD_ERR_INI_SIZE
                echo "El tamaño del archivo supera el límite permitido por
el servidor (parámetro upload_max_filesize del archivo
php.ini).";
                break;
            case 2: // UPLOAD_ERR_FORM_SIZE
                echo "El tamaño del archivo supera el límite permitido por
el formulario (parámetro post_max_size del archivo php.ini).";
                break;
            case 3: // UPLOAD_ERR_PARTIAL
                echo "El envío del formulario se ha interrumpido durante
su transmisión.";
                break;
            case 4: // UPLOAD_ERR_NO_FILE
                echo "El tamaño del archivo que ha enviado es nulo.";
                break;
        }
    }
}

```

```

else {
//si no hay error entonces $_FILES['nombre del archivo']
//['error'] vale 0

    echo "Ningún error durante la transmisión del archivo.<br />";
    if ((isset($_FILES['imagen']['name']))&&($_FILES['imagen']
['error'] == UPLOAD_ERR_OK))) {
        $destino_de_ruta = 'imagenes/';
        //desplazamiento del archivo del directorio temporal
        //(almacenado por defecto) al directorio de destino
        move_uploaded_file($_FILES['imagen']['tmp_name'],
$destino_de_ruta.$_FILES['imagen']['name']);
        echo "El archivo ".$_FILES['imagen']['name']."' Se ha copiado
en el directorio imágenes";
    }
    else {
        echo "El archivo no se ha copiado en el directorio imágenes.";
    }
}

$manager = new Manager($base);
//crear un objeto Blog con los valores de sus atributos
//completados con aquellas recibidas por $_POST
$blog = new Blog();
$blog->setTitulo(htmlentities(addslashes($_POST['titulo']),
ENT_QUOTES));
$blog->setFecha(Date("Y-m-d H:i:s"));
$blog->setComentario
(htmlentities(addslashes($_POST['comentario']), ENT_QUOTES));
$blog->setImagen($_FILES['imagen']['name']);
$iniciar_sesion = $manager->insersionContenido($blog);

if ($iniciar_sesion != 0) {
    echo "<br />Añadir comentario de éxito.<br />";
}
else {
    echo "<br />El comentario no se ha podido añadir.<br />";
}

}
catch(Exception $e)
{
    // mensaje en caso de error
    die('Error : '.$e->getMessage());
}
?>
<a href="formulario_anadir.php" >Volver a la página de inserción</a>
</body>
</html>

```

- La página mostrar\_blog.php:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Blog</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>
    <body>
        <h2>Blog</h2>
        <hr />
        <?php
            include('Blog.class.php');
            include('Manager.class.php');
            try
            {
                // Conexión a la base de datos
                $base = new PDO('mysql:host=127.0.0.1;dbname=Blog', 'root', '');
                $base->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                $manager = new Manager($base);
                $tabla_volver = $manager->getContenidoPorFecha();
                if (empty($tabla_volver))
                {
                    echo 'Ningún mensaje.';
                }
                else {
                    date_default_timezone_set('Europe/Paris');
                    foreach ($tabla_volver as $valor) { // $valor contiene
                        // el objeto blog
                        $dt_inicio = date_create_from_format(
                            'Y-m-d H:i:s', $valor->getFecha());
                        echo "<h3>".$valor->getTitulo()."</h3>";
                        echo "<h4>El ". $dt_inicio->format(
                            'd/m/Y H:i:s')."</h4>";
                        echo "<div style='width:400px'>";
                        echo $valor->getComentario()."</div>";
                        if ($valor->getImagen() != "") {
                            echo "<img src='imagenes/" . $valor->getImagen() . "' width='200px' height='200px' />";
                        }
                        echo "<hr />";
                    }
                }
            }
            catch(Exception $e)
            {
                // mensaje en caso de error
                die('Error : '.$e->getMessage());
            }
        ?>
        <br />
        <a href="formulario_anadir.php" >Volver a la página de inserción</a>
    </body>
</html>

```

- La página Blog.class.php:

```

<?php
class Blog
{
    // Declaración de atributos
    private $id;
    private $Titulo;
    private $date;
    private $Comentario;
    private $Imagen;

    //de acceso
    public function getId()
    {
        return $this->id; //vuelve al inicio de sesión
    }
    public function setId($id)
    {
        $this->id = $id; //escrito en el atributo de id
    }

    public function getTitulo()
    {
        return $this->titulo; //devuelve el título
    }
    public function setTitulo($titulo)
    {
        $this->titulo = $titulo; //escrito en el atributo título
    }

    public function getFecha()
    {
        return $this->Fecha; //devuelve la fecha
    }
    public function setFecha($fecha)
    {
        $this->Fecha = $fecha; //escrita en el atributo de fecha
    }

    public function getComentario()
    {
        return $this->comentario; //devuelve el comentario
    }
    public function setComentario($Comentario)
    {
        $this->comentario = $comentario; //escribe en el atributo
        de comentario
    }

    public function getImagen()
    {

```

```

        return $this->Imagen; //devuelve el nombre de la imagen
    }
    public function setImagen($imagen)
    {
        $this->Imagen = $imagen; //escrita en el atributo de imagen
    }

}
?>

```

- La página Manager.class.php es:

```

<?php
class Manager
{
    private $base; // Instancia de PDO

    public function __construct($base)
    {
        $this->setDb($base);
    }

    public function setDb(PDO $base)
    {
        $this->base = $base;
    }

    public function getContenidoPorFecha() {
        $matriz = array();
        $contador = 0;
        $resultado = $this->base->query(
'SELECT * FROM Contenido ORDER BY Fecha');
        //busca en cada registro devuelto por la consulta
        while ($registro = $resultado->busca()) {
            $blog = new Blog();
            $blog->setId($registro['Id']);
            $blog->setTitulo($registro['Titulo']);
            $blog->setFecha($registro['Fecha']);
            $blog->setComentario($registro['Comentario']);
            $blog->setFoto($registro['Imagen']);
            $matriz[$contador] = $blog; //Almacena el objeto
            //en la matriz
            $contador++;
        }
        return $matriz;
    }

    public function insercionContenido(Blog $blog) {
        $sql = "INSERT INTO Contenido (Titulo, Fecha, Comentario, Imagen)
VALUES ('".$blog->getTitulo()."', '".$blog->getFecha()."',
$blog->getComentario()."', '".$blog->getImagen()."')";
        $this->base->exec($sql);
        //recuperar el último inicio de sesión
        $inicio_de_sesion = $this->base->lastInsertId();
    }
}

```

```
    return $inicio_de_sesion;
}
?>
```

## Crear una newsletter

Cree una página PHP que recupere la lista de nombres y de e-mails de personas. A continuación envíe un mensaje por e-mail. También actualice en la base de datos la fecha de envío a cada una de estas personas.

### Solución

- Cree la base de datos Newsletter y ejecute el script.

- El script SQL que crea la tabla Cliente es:

```
CREATE TABLE Newsletter.Cliente (
Id INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
Nombre VARCHAR( 20 ) NOT NULL ,
Email VARCHAR( 25 ) NOT NULL ,
Fecha_envio DATE NOT NULL

) ENGINE = MYISAM ;
```

- La página envío\_newsletter.php es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <title>NewsLetter</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
  <body>
    <?php date_default_timezone_set('Europe/Paris');?>
    <h2>Envío de la newsletter el <?php echo date(
"Y-m-d H:i:s");?></h2>
    <hr />
    <?php
$connect = mysqli_connect("127.0.0.1", "root", "", "NewsLetter");

/* Comprobar la conexión */
if (!$connect) {
  echo "Fallo de la conexión : ".mysqli_connect_error();
  exit();
}

$consulta = "SELECT * FROM Cliente ORDER BY Nombre";
if ($resultado = mysqli_query($connect,$consulta)) {
  /* busca la matriz asociativa */
  while ($registro = mysqli_fetch_assoc($resultado)) {
    echo "Nombre:".$registro['Nombre'].", ";
    echo "Email:".$registro['Email'];
    echo "<br />";
    //mensaje en formato HTML
    $mensaje = 'Hola Sr/Sra '.$registro['Nombre'].' !
<br />Email : '.$registro['Email'].'<br />';
```

```
$mensaje = $mensaje.'El mensaje de la newsletter es:';  
$mensaje = $mensaje.';Hasta pronto !';  
  
$from = "From: Estefanía Morales <newsletter@misitio.es>\n";  
$from = $from."Mime-Version: 1.0\nContent-Type: text/html; charset=ISO-8859-1\n";  
// envío del mail  
mail($registro['Email'],'Newsletter',$mensaje,$from);  
  
//actualizar la fecha de envío en la base de datos  
$consultaMAJ = "UPDATE Cliente set Fecha_envio=NOW() where  
Id=".$registro['Id'];  
$resultadoMAJ = mysqli_query($connect,$consultaMAJ);  
}  
}  
?  
  
</body>  
</html>
```

## Crear un flujo RSS

Cree una página PHP con el flujo RSS que muestre las últimas novedades del periódico "El País" a la siguiente dirección: <http://ep01.epimg.net/rss/elpais/portada.xml>

### Solución

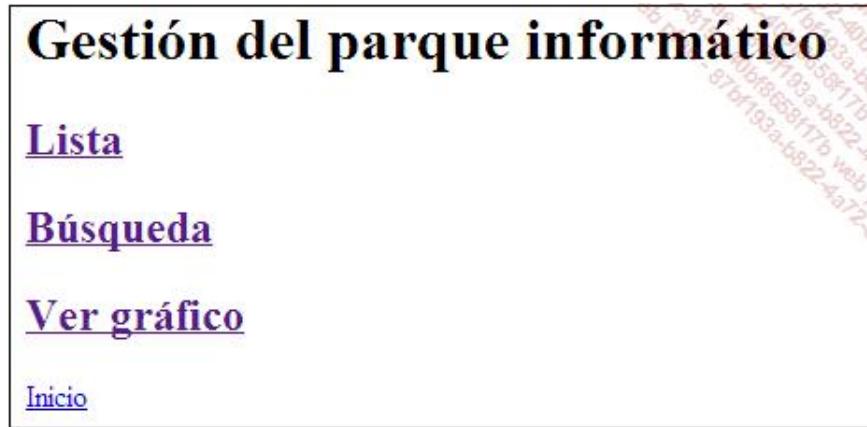
- La página mostrar\_rss.php es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
    <head>
        <title>Ejercicio con RSS</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
        <link rel="alternate" type="application/rss+xml"
href="http://ep01.epimg.net/rss/elpais/portada.xml" />
    </head>
    <body>
        <?php
$rss = simplexml_load_file('http://ep01.epimg.net/rss/elpais/
portada.xml');
foreach ($rss->channel->item as $item) {
    echo '<div>
        <h2>' . $item->title . '</h2>
        <h4>post; el: ' . date("d/m/Y", strtotime($item->
pubDate)) . '</h4>
        ' . $item->texto . ' <a href="' . $item->link . '">Leer todo el
artículo</a>
    </div><br />';
}
?>
    </body>
</html>
```

## Gestión de un parque informático en MVC

El objetivo es crear varios archivos PHP que permitan mostrar y buscar las máquinas en un parque informático. También será necesario visualizar un gráfico que represente el número de máquinas por sala. Estos archivos deben respetar la arquitectura Modelo Vista Controlador.

Inicio:



Lista:

# Gestión del parque informático

## Ver máquinas

id	ip	mac	nombre	sala
12	124.25.42.258	00:1454:5687_n20:31	pc-1	1
13	125.25.56.258	00:1564:5787_n21:35	pc-2	1
14	126.25.52.678	00:1674:58987_n22:31	pc-3	1
15	127.25.52.544	00:1784:5587_n23:35	pc-4	1
16	128.25.52.324	00:1744:5387_n24:31	pc-5	1
17	129.25.52.567	00:1784:4587_n25:35	pc-6	1
18	145.25.45.845	00:1234:5587_n26:31	pc-7	1
22	155.25.32.258	00:1344:6587_n27:35	pc-8	1
24	165.25.22.258	00:1454:7587_n28:31	pc-9	2
25	175.25.32.258	00:1564:8587_n29:35	pc-10	2
26	185.25.52.258	00:1784:9587_n34:31	pc-11	2
28	125.25.72.258	00:1254:125587_n34:35	pc-12	3
31	125.25.52.258	00:3654:3587_n44:32	pc-13	3
32	125.25.62.258	00:6754:4587_n54:35	pc-14	3
42	125.25.54.258	00:6754:4587_n44:32	pc-15	3
52	125.25.56.288	00:4554:56587_n64:35	pc-16	3
52	125.25.56.278	00:6754:7587_n74:32	pc-17	4
62	125.25.57.268	00:7854:8587_n84:35	pc-18	4
72	125.25.58.248	00:9854:9587_n94:32	pc-19	4
82	125.25.59.258	00:4554:25587_n55:35	pc-20	4

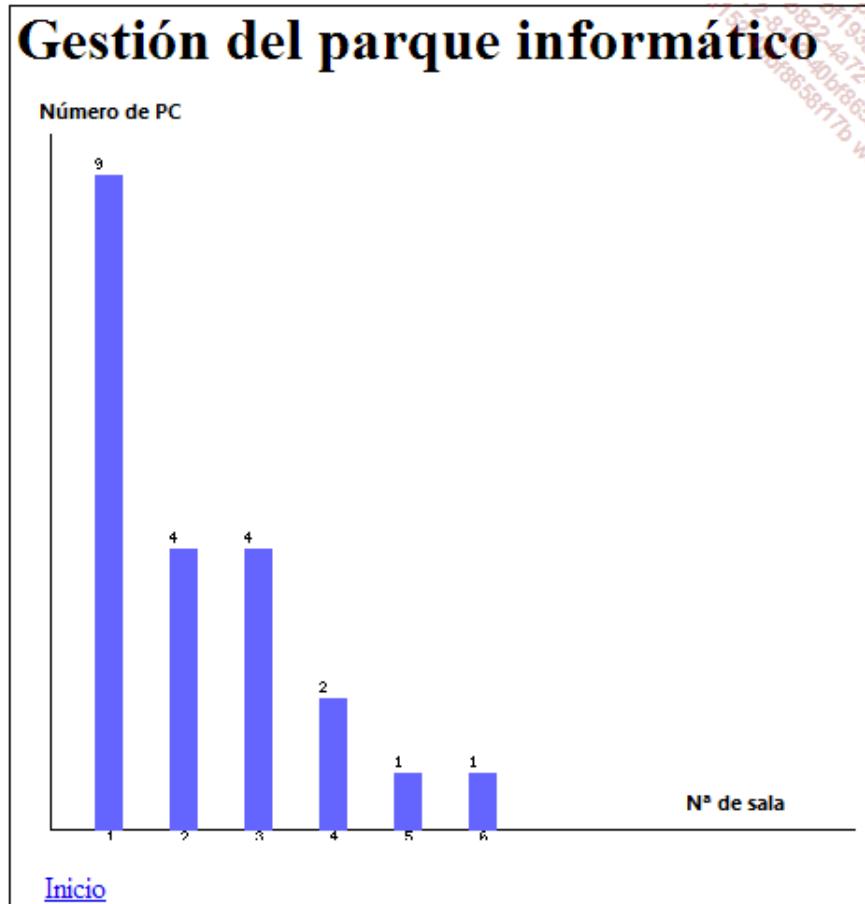
Búsqueda:

## Gestión del parque informático

### Selección de una máquina de dominio

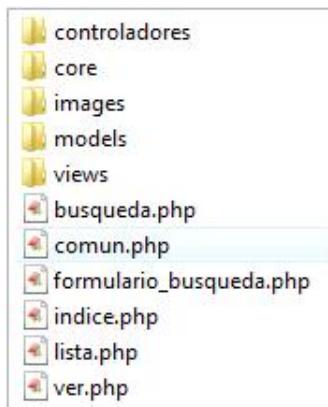
Dirección IP del parque:

[Inicio](#)



## Solución

A continuación se muestra el contenido de la carpeta principal:



- La página ver.php:

```
<?php // Dispatcher secundario  
// Archivo comun  
require('comun.php');  
$controlador->ver(); // acción del controlador  
?>
```

- La página común.php:

```
<?php // código común a todos los dispatchers
define('ROOT', '');
define('WEBROOT', ''); // Archivo de Core
require(ROOT.'core/model.php');
require(ROOT.'core/controlador.php'); //echo 'Conexión a la BDD';
require(ROOT.'controladores/parque.php'); // Inicialización del controlador
$controlador = new parque();
?>
```

- La página formulario\_busqueda.php:

```
<?php // Dispatcher secundario
require('comun.php');
$controlador->formulario_busqueda(); // acción del controlador
?>
```

- La página indice.php:

```
<?php // Dispatcher principal
require('comun.php');
$controlador->indice(); // acción del controlador
?>
```

- La página lista.php:

```
<?php // Dispatcher secundario
require('comun.php');
$controlador->busqueda(""); //acción de controlador
?>
```

- La página busqueda.php:

```
<?php // Dispatcher secundario
require('comun.php');
extract($_POST); // Importante las variables en la tabla
de símbolos. Permiten meter contenido el contenido
de $_POST['addip'] en la variable $addip
$controlador->busqueda($addip); // acción del controlador
?>
```

- La página controladores/parque.php:

```
<?php
class parque extends controlador {
    var $models = array('parque_model');
    //Action indice
    function indice(){
        $this->render('indice'); //llamada de la vista indice.php
    }
}
```

```

function busqueda($ip) {
    $d['record'] = $this->parque_model->getByIP($ip);
    $this->set($d);
    $this->render('busqueda'); //llamada de la vista busqueda.php
}

//ver el número de PC por sala (limitado a los 8 primeros)
function ver(){

    $d['record'] = $this->parque_model->getNumeroPCSala();
    $num_maximo = 0; //mayor número de PC
    $val1=0; // número de PC en una sala
    $val2=0; // número de la sala
    $tabla_PC = array(); //clave de la tabla = nº de sala,
valor de la tabla = número de PC en la sala
    foreach($d['record'] as $clave=>$val) {
        $val1 = $val["numero"];
        if ($val1 > $num_maximo) {
            $num_maximo = $val1;
        }
        $val2 = $val["sala"];
        $tabla_PC[$val2] = $val1;
    }

$numero_element = count($tabla_PC); //número de elementos en la tabla

    $largoImage = 450;
    $saltoImage = 400;
    $image = imagecreate($largoImage, $saltoImage);
    $blanco = imagecolorallocate($image, 255, 255, 255);
    $negro = imagecolorallocate($image, 0, 0, 0);
    $azul = imagecolorallocate($image, 100, 100, 255);

    // trazo horizontal para representar el eje de las salas
    imageline($image, 10, $saltoImage-10, $largoImage-10,
$saltoImage-10, $negro);
    // ver número de las salas
    for ($j=1; $j<=$numero_element; $j++) {
        imagestring($image, 0, $j*40, $saltoImage-10, $j, $negro);
    }

    // trazo vertical que representa el número de PC
    imageline($image, 10, 10, 10, $saltoImage-10, $negro);

    $num_maximo = $num_maximo+1; //para ver un gráfico un poco
más alto que el número máximo
    // traza de los rectángulos
    for ($j=1; $j<=$numero_element; $j++) {
        // el número máximo de PC proporcional al alto de la imagen
        $saltoRectangulo =
round((($tabla_PC[$j]*$saltoImage)/$num_maximo));
        imagefilledrectangle($image, $j*40-6,
$saltoImage-$saltoRectangulo, $j*40+8, $saltoImage-10, $azul);
        imagestring($image, 0, $j*40-6,

```

```

$altoImage-$altoRectangulo-10, $tabla_PC[$j], $negro);
}
imagestring($image, 3, 0, 0, "Número de PC.", $negro);
imagestring($image, 3, $largoImage-100, $altoImage-30,
"Nº de sala.", $negro);
imagepng($image, "./images/miImagen.png");
imagedestroy($image);

$this->render('ver'); //llamada de la vista ver.php
}

function formulario_busqueda(){
$this->render('formulario_busqueda');
//llamada de la vista formulario_busqueda.php
}
}

?>

```

- La página core/controlador.php:

```

<?php
class controlador{
var $vars = array();
var $layout = 'default';
function __construct(){
if(isset($_POST)){
$this->data = $_POST;
}
if(isset($this->models)){
foreach($this->models as $v){
$this->loadModel($v); //carga del modelo, aquí parque_model
}
}
function set($d){
$this->vars = array_merge($this->vars,$d);
//fusiona las tablas $this->vars et $d
}

// inclusión del archivo que se pasa como parámetro
Function render($filename){
extract($this->vars); //Importa las variables en la tabla
de símbolos.
ob_start();
require(ROOT.'views/'.get_class($this).'/'.$filename.'.php');
$content_for_layout = ob_get_clean(); //Lee el contenido de
memoria de salida y la borra
//Esto permite recuperar el contenido de la vista y almacenar
en la variable $content_for_layout
require(ROOT.'views/layout/'.$this->layout.'.php');
}

function loadModel($name){
require_once(ROOT.'models/'.strtolower($name).'.php');
$this->$name = new $name();
}

```

```
    }
}
```

- La página core/model.php:

```
<?php /** * Object Model * Permite las interactiones con la base  
de datos * */  
  
class Model{  
    public $table;  
    public $id;  
    private static $base;  
    private static $servidor='127.0.0.1';  
    private static $bdd='parque_info'; //nombre de la base de datos  
    private static $user='root' ;  
    private static $mdp='';  
  
    public function __construct()  
    {  
        //Conexión a la base de datos  
        Model::$base = mysqli_connect(Model::$servidor,  
Model::$user,Model::$mdp,Model::$bdd);  
    }  
  
    public function __destruct()  
    {  
        Model::$base = null;  
    }  
  
    /**      * Permite recuperar varias líneas en la BDD  
    * @param $data condiciones de recuperación * */  
    public function find($data=array()){  
        $conditions = "1=1";  
        $fields = "*";  
        $limit = " ";  
        $order = "id DESC";  
        extract($data); // Importa las variables en la tabla de símbolos.  
        if(isset($data["limit"])){  
            $limit = "LIMIT ".$data["limit"];  
        }  
        $sql = "SELECT ".$fields." FROM ".$this->table." WHERE ".  
$conditions." ORDER BY ".$order." ".$limit;  
        $req = mysqli_query(Model::$base, $sql) or die(mysqli_error  
(Model::$base)."  
=> ".$sql);  
        $d = array();  
        while($data = mysqli_fetch_assoc($req)){  
            $d[] = $data;  
        }  
        return $d;  
    }  
  
    /**      * Permite hacer una consulta compleja * @param $sql  
    Consulta a realizar * */  
    public function query($sql){  
        $req = mysqli_query(Model::$base, $sql)
```

```

or die(mysqli_error(Model::$base)."<br/> => ".$sql);
$d = array();
while($data = mysqli_fetch_assoc($req)){
    $d[] = $data;
}
return $d;
}

/**      * Permite cargar un modelo      * @param $name Nombre del
modelo a cargar * */
static function load($name){
    require("$name.php");
    return new $name();
}
} ?>

```

- La página models/parque\_model.php:

```

<?php
class parque_model extends Model{
    var $table = 'ordenadores'; //nombre de la tabla

    //devuelve los registros en función de sus dirección IP
    function getByIP($ip){
        return $this->query("SELECT id, ip, mac, nombre, sala FROM
". $this->table . " WHERE ip LIKE '" . $ip . "%' ORDER BY ip ASC");
    }

    //devuelve el número de PC por sala
    function getNumeroPCSalla(){
        return $this->query("SELECT count(id) as numero, sala FROM
". $this->table . " GROUP BY Sala order by sala LIMIT 8");
    }

}
?>

```

- La página views/layout/default.php:

```

<HTML>
    <HEAD> <TITLE> Parque inform&aacute;tico </TITLE> </HEAD>
    <BODY> <h1>Gesti&oacute;n del parque inform&aacute;tico</h1>
        <?php echo $content_for_layout;?> <p>      <a href="<?php echo
WEBROOT; ?>indice.php">Inicio</a> </p>
    </BODY>
</HTML>

```

- La página views/parque/ver.php:

```



```

- La página views/parque/formulario\_busqueda.php.

```

<body>
<h2> Selección de una máquina de dominio</h2>
<form name="formip" method="post" action="<?php echo
WEBROOT; ?>busqueda.php"> <p>Dirección IP del parque:</p>
<p><input name="addip" size="15" maxlength="15" type="text"></p>
<p><input value="Buscar" type="submit"></p>
</form>
</body>

```

- La página views/parque/indice.php:

```

<h2> <a href="<?php echo WEBROOT; ?>Lista.php">Lista</a> </h2>
<h2> <a href="<?php echo WEBROOT; ?>formulario_busqueda.php">
Búsqueda</a> </h2>
<h2> <a href="<?php echo WEBROOT; ?>ver.php">
Ver gráfico</a> </h2>

```

- La página views/parque/busqueda.php:

```

<h2><a>Ver máquinas</a></h2>
<?php
if (isset($record[0])) { //si hay máquinas
    echo "<table border=1 width=\"100%\">";
    $Lista_indices=array_keys((array)$record[0]);
    echo "<tr>";
    while($indice=each($Lista_indices)) {
        echo "<th>";
        echo $indice['value'];
        echo "</th>";
    }
    echo "</tr>";

    foreach((array)$record as $clave_tabla=>$linea) {
        echo "<tr>";
        foreach($linea as $clave=>$valor) {
            echo "<td align=center>";
            echo $valor;
            echo "</td>";
        }
        echo "</tr>";
    }
    echo "</table>";
}
else {
    echo "No existen máquinas que cumplan sus criterios.";
}
?>

```

Para terminar, el script de creación de la tabla y de inserción de datos.

--

```

-- Estructura de la tabla `ordenadores`
--

DROP TABLE IF EXISTS `ordenadores`;
CREATE TABLE IF NOT EXISTS `ordenadores` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ip` char(15) NOT NULL,
  `mac` char(17) NOT NULL,
  `nombre` char(15) NOT NULL,
  `sala` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=82 ;

-- 
-- Contenido de la tabla `ordenadores`
-- 

INSERT INTO `ordenadores`(`id`, `ip`, `mac`, `nombre`, `sala`) VALUES
(47, '178.17.5.4', '00:1a:73:7d:e0:eb', 'pc-x', 2),
(45, '172.17.5.2', '00:14:6c:65:86:ab', 'pc-w', 2),
(16, '172.17.4.21', '00:0d:56:c2:f2:5a', 'pc-01', 2),
(17, '192.17.4.22', '00:0d:56:c2:f3:a7', 'pc-02', 1),
(18, '172.17.4.23', '00:0d:56:c2:e8:4d', 'pc-03', 1),
(22, '172.17.4.27', '00:0d:56:c2:f3:ad', 'pc-07', 1),
(23, '178.17.4.28', '00:11:85:10:f4:bf', 'pc-08', 2),
(24, '172.17.4.29', '00:11:85:11:01:2b', 'pc-09', 1),
(25, '172.17.4.30', '00:11:85:14:4e:37', 'pc-10', 1),
(26, '192.17.4.31', '00:11:85:14:4d:99', 'pc-11', 1),
(27, '172.17.4.32', '00:11:85:62:71:05', 'pc-12', 1),
(28, '172.17.4.33', '00:11:85:14:4e:22', 'pc-13', 1),
(29, '176.17.4.34', '00:11:85:14:4d:ab', 'pc-14', 1),
(30, '172.17.4.35', '00:0f:b5:85:0c:21', 'pc-15', 3),
(31, '168.17.4.36', '00:14:85:7d:d2:35', 'pc-16', 3),
(32, '172.17.4.37', '00:14:85:7a:a9:8c', 'pc-17', 3),
(79, '178.17.4.38', '00:14:85:7D:CF:AA', 'pc-18', 4),
(34, '178.17.4.39', '00:14:85:79:78:fa', 'pc-19', 4),
(39, '172.17.4.43', '00:14:85:7a:78:dc', 'pc-23', 3),
(40, '172.17.4.44', '00:14:85:7a:78:a5', 'pc-24', 3),
(41, '172.17.4.45', '00:14:85:7a:78:de', 'pc-25', 5);

```

## Crear un sitio Web para gestionar becarios

Se trata de insertar en la tabla aquellos becarios que están en proceso de formación. Estos becarios tienen nombre, apellidos, una nacionalidad, un tipo de formación y un formador en una sala entre dos fechas determinadas.

Esta página tiene un código JavaScript que activa el área de formadores y las fechas, según el tipo de formación que seleccione.

### Insertar un becario en formación

Nombre:

Apellidos:

Nacionalidad: Española

Tipo de formación: Web designer

Formadores por fecha:

Estefanía Morales HonHon en sala 101, Inicio:  , fin:

Pablo García Arripe en sala 102, Inicio:  , fin:

Emilio Martín Cruz en sala 201, Inicio:  , fin:

María González Sánchez en sala 202, Inicio:  , fin:

[Eliminar un becario](#) [Modificar un becario](#)

También puede eliminar los datos de un becario:

### Eliminar datos de un becario

Nombre	Apellidos	Nacionalidad	Tipo de formación	Formador-Sala-Fecha Inicio-Fin	Eliminar
Becker	Josephine	Alemana	Desarrollador	Emilio - 201 - 01/02/2013 02/02/2014	<input type="checkbox"/>
Dupont	Robert	Francesa	Desarrollador	Emilio - 201 - 01/02/2013 02/02/2014	<input type="checkbox"/>
Monfils	Boby	Francesa	Desarrollador	Emilio - 201 - 01/02/2013 02/02/2014	<input type="checkbox"/>
Estefanía	Morales Honhon	Española	Web Designer	Pablo - 102 - 01/02/2013 02/02/2014	<input type="checkbox"/>
Sharapova	Nadia	Rusa	Web Designer	Pablo - 102 - 01/02/2013 02/02/2014 Emilio - 201 - 01/02/2013 02/02/2014	<input type="checkbox"/>

[Añadir un becario](#) [Modificar un becario](#)

Puede modificar todos los becarios al mismo tiempo:

## Modificar un becario

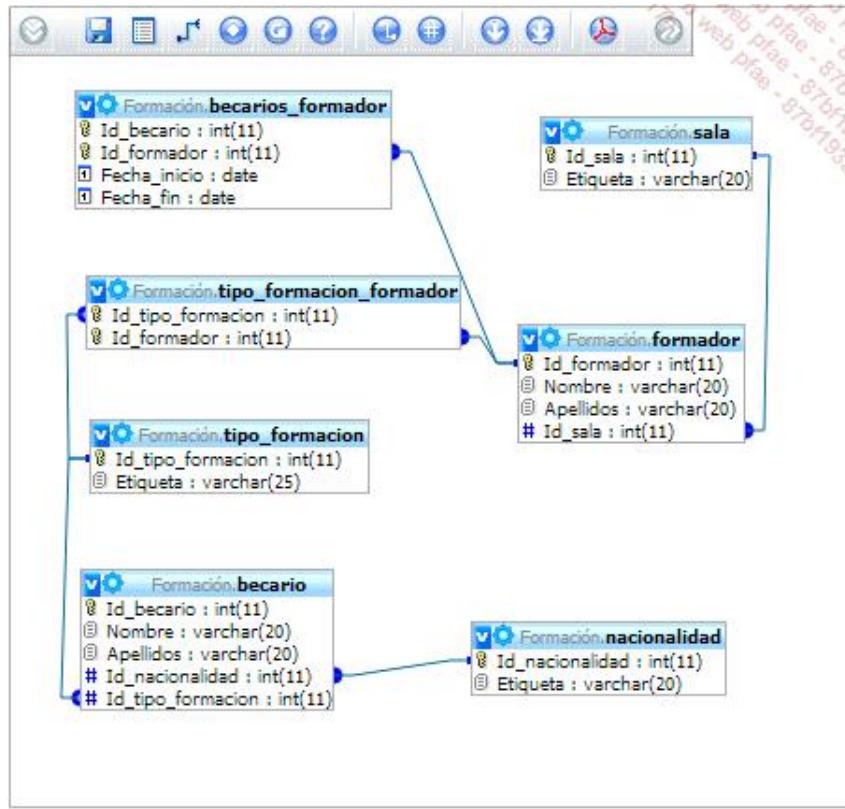
Nombre	Apellidos	Nacionalidad	Tipo de formación	Formador-Sala-Fecha Inicio-Fin	Modificar
Becker	Josephine	Alemana	Desarrollador	<input type="checkbox"/> Morales-101- 29/07/2013   04/14/2014 <input type="checkbox"/> García-102- 29/07/2013   04/14/2014 <input checked="" type="checkbox"/> Martín-201- 29/07/2013   04/14/2013 <input type="checkbox"/> González-202- 29/07/2013   04/14/2013	<input type="checkbox"/>
Dupont	Robert	Francesa	Desarrollador	<input checked="" type="checkbox"/> Morales-101- 29/07/2013   04/14/2014 <input type="checkbox"/> García-102- 29/07/2013   04/14/2014 <input type="checkbox"/> Martín-201- 29/07/2013   04/14/2014 <input type="checkbox"/> González-202- 29/07/2013   04/14/2014	<input type="checkbox"/>
Monfils	Boby	Francesa	Desarrollador	<input type="checkbox"/> Morales-101- 29/07/2013   04/14/2014 <input type="checkbox"/> García-102- 29/07/2013   04/14/2014 <input type="checkbox"/> Martín-201- 29/07/2013   04/14/2014 <input checked="" type="checkbox"/> González-202- 29/07/2013   04/14/2014	<input type="checkbox"/>
Estefanía	Morales Honhon	Española	Web Designer	<input type="checkbox"/> Morales-101- 29/07/2013   04/14/2014 <input type="checkbox"/> García-102- 29/07/2013   04/14/2014 <input type="checkbox"/> Martín-201- 29/07/2013   04/14/2014 <input checked="" type="checkbox"/> González-202- 29/07/2013   04/14/2014	<input type="checkbox"/>
Sharapova	Nadia	Rusa	Web Designer	<input type="checkbox"/> Morales-101- 29/07/2013   04/14/2014 <input type="checkbox"/> García-102- 29/07/2013   04/14/2014 <input checked="" type="checkbox"/> Martín-201- 29/07/2013   04/14/2014 <input type="checkbox"/> González-202- 29/07/2013   04/14/2014	<input type="checkbox"/>

[Modificar](#)

[Añadir un becario](#) [Eliminar un becario](#)

Esta página contiene un JavaScript que activa las áreas de formadores y las fechas, según el tipo de formación que seleccione.

El esquema de la base de datos es:



## Solución

- Cree la base de datos formación.
- Ejecute el script para crear unas tablas con sus correspondientes datos:

```

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

-- Estructura de la tabla `formador` 

DROP TABLE IF EXISTS formador;
CREATE TABLE formador (
    Id_formador int(11) NOT NULL AUTO_INCREMENT,
    Apellido varchar(20) NOT NULL,
    Nombre varchar(20) NOT NULL,
    Id_sala int(11) NOT NULL,
    PRIMARY KEY (Id_formador)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

-- Contenido de la tabla `formador` 

INSERT INTO formador (Id_formador, Apellido, Nombre, Id_sala)
VALUES
(1, 'Morales HonHon', 'Estefanía', 1),
(2, 'García Arripe', 'Pablo', 2),
(3, 'Mártin Cruz', 'Emilio', 3),
(4, 'Gonzalez Sánchez', 'María', 4);

-- Estructura de la tabla `nacionalidad` 
    
```

```

DROP TABLE IF EXISTS nacionalidad;
CREATE TABLE nacionalidad (
    Id_nacionalidad int(11) NOT NULL AUTO_INCREMENT,
    Etiqueta varchar(25) NOT NULL,
    PRIMARY KEY (Id_nacionalidad)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

-- Contenido de la tabla `nacionalidad`

INSERT INTO nacionalidad (Id_nacionalidad, Etiqueta) VALUES
(1, 'Francés'),
(2, 'Inglés'),
(3, 'Alemán'),
(4, 'Ruso');

-- Estructura de la tabla `sala`

DROP TABLE IF EXISTS sala;
CREATE TABLE sala (
    Id_sala int(11) NOT NULL AUTO_INCREMENT,
    Etiqueta varchar(20) NOT NULL,
    PRIMARY KEY (Id_sala)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

-- Contenido de la tabla `sala`

INSERT INTO sala (Id_sala, Etiqueta) VALUES
(1, '101'),
(2, '102'),
(3, '201'),
(4, '202');

-- Estructura de la tabla `becario`


DROP TABLE IF EXISTS becario;
CREATE TABLE becario (
    Id int(11) NOT NULL AUTO_INCREMENT,
    Nombre varchar(20) NOT NULL,
    Apellido varchar(20) NOT NULL,
    Id_nacionalidad int(11) NOT NULL,
    Id_tipo_formacion int(11) NOT NULL,
    PRIMARY KEY (Id)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=12 ;

-- Contenido de la tabla `becario`

INSERT INTO becario (Id, Apellido, Nombre, Id_nacionalidad,
Id_tipo_formacion) VALUES
(1, 'López', 'Nadia', 4, 1),
(2, 'Del Pozo', 'Carolina', 1, 2),
(8, 'Gómez', 'Alex', 2, 1),
(4, 'Rodríguez', 'María', 3, 2),
(6, 'Morales', 'Estefanía', 1, 2);

-- Estructura de la tabla `becario_formador`
```

```

DROP TABLE IF EXISTS becario_formador;
CREATE TABLE becario_formador (
    Id_becario int(11) NOT NULL,
    Id_formador int(11) NOT NULL,
    Fecha_inicio Date NOT NULL,
    Fecha_fin Date NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- Contenido de la tabla `becario_formador`


INSERT INTO becario_formador (Id_becario, Id_formador,
Fecha_inicio, Fecha_fin) VALUES
(1, 1, '2013-07-25', '2013-10-28'),
(1, 2, '2013-10-31', '2013-12-30'),
(2, 4, '2013-08-26', '2013-10-18'),
(8, 2, '2013-08-15', '2014-02-15'),
(6, 4, '2013-08-21', '2013-10-21'),
(4, 3, '2013-08-17', '2014-02-21');


-- Estructura de la tabla `tipo_formacion`


DROP TABLE IF EXISTS tipo_formacion;
CREATE TABLE tipo_formacion (
    Id_tipo_formacion int(11) NOT NULL AUTO_INCREMENT,
    Etiqueta varchar(25) NOT NULL,
    PRIMARY KEY (Id_tipo_formacion)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

-- Contenido de la tabla `tipo_formacion`


INSERT INTO tipo_formacion (Id_tipo_formacion, Etiqueta) VALUES
(1, 'Web designer'),
(2, 'Desarrollador');


-- Estructura de la tabla `tipo_formacion_formador`


DROP TABLE IF EXISTS tipo_formacion_formador;
CREATE TABLE tipo_formacion_formador (
    Id_tipo_formacion int(11) NOT NULL,
    Id_formador int(11) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- Contenido de la tabla `tipo_formacion_formador`


INSERT INTO tipo_formacion_formador (Id_tipo_formacion,
Id_formador) VALUES
(1, 1),
(1, 2),
(2, 3),
(2, 4);

```

- La página formacion.php es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "formacion");

/* Comprobar la conexión */
if (!$connect) {
    echo "Fallo de la conexión : ".mysqli_connect_error();
    exit();
}
setlocale (LC_TIME, 'es-ES.utf-8','esp');
?>
<head>
    <title>Ejercicio formación</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-15" />
    <script language="JavaScript">
        function comprueba_fecha(fecha_recibida) {
            if (fecha_recibida == '') {
                alert('fecha no válida');
                document.formulario.enviar.disabled=true;
            }
            else {
                document.formulario.enviar.disabled=false;
            }
        }
    </script>
    <style type="text/css">
        body {
            font-family: sans-serif;
            font-size: 1em;
        }
        .error {
            color: red;
        }
    </style>
</head>
<body>
    <h1>Formulario de Ejercicio de Formación</h1>
    <form name="formulario" method="post" action="ejercicio_formacion.php">
        <table border="1">
            <tr>
                <td>Nombre:</td>
                <td><input type="text" name="nombre" value="" /></td>
            </tr>
            <tr>
                <td>Apellido:</td>
                <td><input type="text" name="apellido" value="" /></td>
            </tr>
            <tr>
                <td>Fecha de nacimiento:</td>
                <td><input type="text" name="fecha_nacimiento" value="" /></td>
            </tr>
            <tr>
                <td>Sexo:</td>
                <td><input type="radio" name="sexo" value="M" checked="checked"/> Hombre <input type="radio" name="sexo" value="F"/> Mujer</td>
            </tr>
            <tr>
                <td>Edad:</td>
                <td><input type="text" name="edad" value="" /></td>
            </tr>
            <tr>
                <td>Tipo de formación:</td>
                <td><input type="text" name="tipo_formacion" value="" /></td>
            </tr>
            <tr>
                <td>Formador:</td>
                <td><input type="text" name="formador" value="" /></td>
            </tr>
            <tr>
                <td colspan="2" style="text-align: center; padding-top: 10px;">
                    <input type="submit" value="Enviar" /> <input type="button" value="Cancelar" onclick="history.back()"/>
                </td>
            </tr>
        </table>
    </form>
</body>
<?php

```

```

        }
        $incremento = $incremento + 1;
    }
    /*libera el objeto resultado */
    mysqli_free_result($resultado);
}
?>
//reajustar los checkbox
var i;
var obj_input;
for (i = 0; i < document.formulario.elementos.length; i++)
{
    obj_input = document.formulario.elementos[i];
    if(obj_input.type=="checkbox") // comprueba si CheckBox
    {
        obj_input.disabled=true;
        obj_input.checked=false;
    }

    if(obj_input.name.substring(0,5)=="inicio" ||
obj_input.name.substring(0,3)=="fin") // comprueba si fecha inicio o
                                         // si fecha de fin
    {
        obj_input.disabled=true;
    }
}

// recuperar el Id_formacion seleccionado en
//la lista tipo_formacion
id_formacion=document.formulario.tipo_formacion.options[
document.formulario.tipo_formacion.selectedIndex].value;
var matriz_formador = new Array();
matriz_formador = matriz_formacion[id_formacion];
for(var i= 0; i < matriz_formador.length; i++)
{
    document.getElementById(
"formador"+matriz_formador[i]).disabled=false;

    document.getElementById(
"inicio"+matriz_formador[i]).disabled=false;

    document.getElementById(
"fin"+matriz_formador[i]).disabled=false;
}

}
</script>
</head>

<body onload="enable_formador()">

<h2>Insertar un becario en formación</h2><br />
<form name="formulario" id="formulario"
action="anadir_becario.php" method="POST">
apellido: <input type="text" name="apellido"><br />

```

```

nombre: <input type="text" name="nombre"><br />
nacionalidad: <select name="nacionalidad">
<?php
$consulta = "SELECT * FROM nacionalidad";

if ($resultado = mysqli_query($connect,$consulta)) {

    /* busca la matriz asociativa */
    while ($registro = mysqli_fetch_assoc($resultado)) {
        echo "<option value='".$registro['Id_nacionalidad']."'>";
        echo $registro['Etiqueta']."</option>";
    }

    /*libera el objeto resultado */
    mysqli_free_result($resultado);
}

?>
</select><br /><br />
tipo de formación: <select name="tipo_formacion"
onchange="enable_formador()">
<?php
$consulta = "SELECT * FROM tipo_formacion";

if ($resultado = mysqli_query($connect,$consulta)) {

    /* busca la matriz asociativa */
    while ($registro = mysqli_fetch_assoc($resultado)) {
        echo "<option value='".$registro['Id_tipo_formacion']."'>";
        echo $registro['Etiqueta']."</option>";
    }

    /*libera el objeto resultado */
    mysqli_free_result($resultado);
}

?>
</select><br /><br />
formadores por fecha:<br />
<?php
$consulta = "SELECT * FROM formador left join sala on
formador.id_sala = sala.id_sala";

if ($resultado = mysqli_query($connect,$consulta)) {

    /* busca la matriz asociativa */
    while ($registro = mysqli_fetch_assoc($resultado)) {
        echo "<input type='checkbox'";
        echo "value='".$registro['Id_formador']."'";
        echo "' id='formador".$registro['Id_formador']."' ";
        echo "' name='formador[]'>".$registro['Nombre']." ".$registro['Apellido'];
        echo " en la sala ".$registro['Etiqueta'].",";
        echo " inicio : <input type='text'";
        echo "' name='inicio".$registro['Id_formador']."' ";
        echo "' id='inicio".$registro['Id_formador']."' ";
        echo "' value='".$strftime("%d/%m/%Y")."' ";
        echo " onchange='comprueba_fecha(this.value)' />,";
    }
}

```

```

        echo " fin: <input tipo='text'>;
echo "'name='fin".$registro['Id_formador'];
echo "'id='fin".$registro['Id_formador']."' value='".date('d/m/Y',
strtotime('+6 month'))."' onchange='comprueba_fecha(this.value)' /><br />";
    }

/* libera el objeto resultado */
mysqli_free_result($resultado);
}

?>
<br />
<input type="submit" name="enviar" value="Enviar" />
</form>
<br />
<a href="lista_becario_a_eliminar.php">Eliminar un
becario</a> <a href="lista_becario_a_modificar.php">Modificar
un becario</a>
<?php

/* Cierre de la conexión */
mysqli_close($connect);

?>
</body>
</html>

```

- La página anadir\_becario.php inserta el becario en la base de datos:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "formacion");

/* Comprobar la conexión */
if (!$connect) {
    echo "Fallo de la conexión : ".mysqli_connect_error();
    exit();
}

?>
<head>
    <title>Ejercicio formación</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15" />

</head>

<body>

<h2>Recepción de datos del becario que se va a insertar</h2>

apellido: <?php echo $_POST['apellido'];?><br />
nombre: <?php echo $_POST['nombre'];?><br />
nacionalidad: <?php echo $_POST['nacionalidad'];?><br />

```

```

tipo de formación: <?php echo $_POST['tipo_formacion'];?><br />
<?php
$consulta = "INSERT INTO becario (Apellido, Nombre, Id_nacionalidad,
Id_tipo_formacion) VALUES ('".htmlentities(addslashes(
$_POST['apellido']), ENT_QUOTES).','".htmlentities(addslashes
($_POST['nombre']),
ENT_QUOTES).',".". $_POST['nacionalidad'].",
", ".$_POST['tipo_formacion']."')";
$resultado = mysqli_query($connect,$consulta);
$inicio_de_sesion=mysqli_insert_id($connect);

if (isset($_POST['formador'])) {?>
<br />
<?php //define la zona de España para la fecha
date_default_timezone_set('Europe/Paris');

foreach ($_POST['formador'] as $valor) {
    $dt_inicio = date_create_from_format('d/m/Y', $_POST['inicio'].$valor);
    $dt_fin = date_create_from_format('d/m/Y', $_POST['fin'].$valor);
    $consulta = "INSERT INTO becario_formador (Id_becario, Id_formador,
Fecha_inicio, Fecha_fin) VALUES (".$inicio_de_sesion.",".
$valor.",'".$dt_inicio->format('Y/m/d')."', '".$dt_fin->format(
'Y/m/d')."')";
    $resultado = mysqli_query($connect,$consulta);
}
}

/* Cierre de la conexión */
mysqli_close($connect);

?>
<br />
;El becario se ha agregado con éxito!
<br />
<a href="formacion.php">Regreso a agregar becario</a>
</body>
</html>

```

- La página lista\_becario\_a\_eliminar.php es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "formacion");

/* Comprobar la conexión */
if (!$connect) {
    echo "Fallo de la conexión : ".mysqli_connect_error();
    exit();
}

?>
<head>
    <title>Ejercicio formación</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15" />

```

```

</head>

<body>

<h2>Eliminar los datos del becario</h2><br />
<form action="eliminar_becario.php" method="POST">
<table border="1">
<tr>
    <th width="100px">Apellido</th> <th width="100px">Nombre</th> <th
width="100px">Nacionalidad</th> <th width="100px">Tipo de
formación</th> <th width="300px">Formador - Sala - Fecha inicio -
Fecha fin</th><th width="50px">Eliminación</th>
</tr>
<?php
$consulta = "SELECT *,nacionalidad.Etiqueta
Nacionalidad,tipo_formacion.Etiqueta as Tipo_formacion,
becario.Id as Id_becario FROM (becario".
        " INNER JOIN nacionalidad ON becario.Id_nacionalidad =
nacionalidad.Id_nacionalidad").
        " INNER JOIN tipo_formacion ON becario.Id_tipo_formacion =
tipo_formacion.Id_tipo_formacion".
        " ORDER BY Nombre";

if ($resultado = mysqli_query($connect,$consulta)) {
    Fecha_default_timezone_set('Europe/Paris');
    /* busca la matriz asociativa */
    while ($registro = mysqli_fetch_assoc($resultado)) {
        echo "<tr>";
        echo "<td>".$registro['Apellido']."</td>";
        echo "<td>".$registro['Nombre']."</td>";
        echo "<td>".$registro['Nacionalidad']."</td>";
        echo "<td>".$registro['Tipo_formacion']."</td>";
        echo "<td>";
        $consulta_formador = "SELECT * FROM becario_formador INNER JOIN
formador ON becario_formador.Id_formador = formador.Id_
formador INNER JOIN sala ON formador.Id_sala = sala.Id_sala WHERE
becario_formador.Id_becario = ".$registro['Id'];
        if ($resultado_formador =
mysqli_query($connect,$consulta_formador)) {
            /* busca la matriz asociativa */
            while ($registro_formador =
mysqli_fetch_assoc($resultado_formador)) {
                $dt_inicio = date_create_from_format('Y-m-d',
$registro_formador['Fecha_inicio']);
                echo $registro_formador['Nombre']." - ".$registro_formador[
'Etiqueta']." - ".$dt_inicio->format('d/m/Y')."-".
".$registro_formador['Fecha_fin']."<br />";
            }
        }
        echo "&nbsp;</td>";
        echo "<td><input type='checkbox' name='supresion[]'
value='".$registro['Id_becario']."' /></td>";
        echo "</tr>";
    }
}

```

```

        }
    }

    /* Cierre de la conexión */
    mysqli_close($connect);

    ?>
</table>
<br />
<input type="submit" name="eliminar" value="eliminar" />
</form>
<br />
<a href="formacion.php">Añadir un becario</a> <a
href="lista_becario_a_modificar.php">Modificar un becario</a>

</body>
</html>

```

- La página eliminar\_becario.php elimina en la base de datos:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "formacion");

/* Comprobar la conexión */
if (!$connect) {
    echo "Fallo de la conexión : ".mysqli_connect_error();
    exit();
}

?>
<head>
    <title>Ejercicio formación</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15" />

</head>

<body>

<?php
if (isset($_POST['supresion'])) {

    foreach ($_POST['supresion'] as $valor) {
        $consulta = "DELETE FROM becario_formador WHERE Id_becario =
        ".$valor;
        $resultado = mysqli_query($connect,$consulta);
        $consulta = "DELETE FROM becario WHERE Id = ".$valor;
        $resultado = mysqli_query($connect,$consulta);
    }
}
/* Cierre de la conexión */
mysqli_close($connect);

```

```

header('location:lista_becario_a_eliminar.php');
?>
</body>
</html>

```

- La página lista\_becario\_a\_modificar.php es:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "formacion");

/* Comprobar la conexión */
if (!$connect) {
    echo "Fallo de la conexión : ".mysqli_connect_error();
    exit();
}

?>
<head>
<title>Ejercicio formación</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15" />
<script language="JavaScript">
function comprueba_Fecha(Fecha_recibida) {
    if (Fecha_recibida == '') {
        alert('Fecha no válida');
        document.formulario.enviar.disabled=true;
    }
    else {
        document.formulario.enviar.disabled=false;
    }
}

function enable_formador(param) { //Id_becario en parámetro
    //reajustar a cero los checkbox
    var i;
    var obj_input;
    for (i = 0; i < document.formulario.elements.length; i++)
    {
        obj_input = document.formulario.elements[i];
        if(obj_input.type=="checkbox" &&
obj_input.name.substring(0,12)!="modificacion") // prueba si
//el tipo = CheckBox y el nombre <> modificacion
        {
            obj_input.disabled=true;
            if (obj_input.id.substring(obj_input.id.indexOf("_")+1) ==
param) { //elimina la selección de los checkbox si modifica la del
//tipo de formación
                obj_input.checked=false;
            }
        }
    }

    if(obj_input.name.substring(0,5)=="inicio" ||

```

```

obj_input.name.substring(0,3)=="fin") // se prueba si fecha inicio
                                         // o fecha fin
{
    obj_input.disabled=true;
}
}

var matriz_formacion = new Array();

<?php
$consulta_stag = "SELECT *, becario.Id AS Id_becario FROM
becario".
" ORDER BY Nombre";

if ($resultado_stag = mysqli_query($connect,$consulta_stag)) {

    while ($registro_stag = mysqli_fetch_assoc($resultado_stag)) {

        $consulta = "SELECT * FROM tipo_formacion_formador ORDER
BY Id_tipo_formacion";

        if ($resultado = mysqli_query($connect,$consulta)) {
            $incremento = 0;
            $Id_tipo_formacion = 0;
            /* busca la matriz asociativa */
            while ($registro = mysqli_fetch_assoc($resultado)) {
                if ($Id_tipo_formacion == $registro['Id_tipo_formacion']) {
                    echo "matriz_formacion[".

$registro['Id_tipo_formacion']."][".$incremento."]=".
$registro['Id_formador'].";\n";
                }
                else {
                    $Id_tipo_formacion = $registro['Id_tipo_formacion'];
                    $incremento = 0;
                    echo "matriz_formacion[".

$registro['Id_tipo_formacion']."] = new Array();\n";
                    echo "matriz_formacion[".

$registro['Id_tipo_formacion']."][".$incremento."]=".
$registro['Id_formador'].";\n";
                }
                $incremento = $incremento + 1;
            }
            /*libera el objeto resultado */
            mysqli_free_result($resultado);
        }
    ?>

    id_formacion=document.formulario.tipo_formacion<?php echo
$registro_stag['Id_becario'];?>.options[document.formulario.tipo_
formacion<?php echo
$registro_stag['Id_becario'];?>.selectedIndex].value;
    var matriz_formador = new Array();
    matriz_formador = matriz_formacion[id_formacion];
    for(var i= 0; i < matriz_formador.length; i++)
    {

```

```

document.getElementById("formador"+matriz_formador[i]+"_<?php
echo $registro_stag['Id_becario'];?>").disabled=false;

document.getElementById("inicio"+matriz_formador[i]+"_<?php
echo $registro_stag['Id_becario'];?>").disabled=false;

document.getElementById("fin"+matriz_formador[i]+"_<?php
echo $registro_stag['Id_becario'];?>").disabled=false;
}

<?php }//fin while
}//fin si ?>
}

</script>
</head>

<body onload="enable_formador(0)">

<h2>Modificación de los datos del becario</h2><br />
<form name="formulario" id="formulario"
action="modificacion_becario.php" method="POST">
<table border="1">
<tr>
<th width="100px">Apellido</th> <th width="100px">Nombre</th>
<th width="100px">Nacionalidad</th> <th width="100px">Tipo de
formación</th> <th width="300px">Formador - Sala - Fecha inicio -
Fecha fin</th><th width="50px">Modificación</th>
</tr>
<?php
$consulta = "SELECT *, becario.Id AS Id_becario FROM becario".
" ORDER BY Nombre";

if ($resultado = mysqli_query($connect,$consulta)) {
    date_default_timezone_set('Europe/Paris');
    /* busca la matriz asociativa */
    while ($registro = mysqli_fetch_assoc($resultado)) {
        echo "<tr>";
        echo "<td><input type='text' name='apellido'.
$registro['Id_becario']."' value='".$registro['Apellido']."' />
</td>";
        echo "<td><input type='text' name='nombre'.
$registro['Id_becario']."' value='".$registro['Nombre']."' /></td>";
        echo "<td>";
        echo '<select name="nacionalidad'.
$registro['Id_becario'].'>';
        $consulta_nac = "SELECT * FROM nacionalidad";
        if ($resultado_nac = mysqli_query($connect,$consulta_nac)) {
            /* busca la matriz asociativa */
            while ($registro_nac = mysqli_fetch_assoc($resultado_nac)) {
                if ($registro_nac['Id_nacionalidad'] == $registro['Id_nacionalidad']) {
                    echo "<option value='".$registro_nac['Id_nacionalidad']."' ".
'selected='selected' >".$registro_nac['Etiqueta']."</option>";
                }
            else {
                echo "<option value='".$registro_nac['Id_nacionalidad'].'

```

```

"">'>".$registro_nac['Etiqueta']."</option>";
        }
    }
    /* libera el objeto resultado */
    mysqli_free_result($resultado_nac);
}
echo "</select>";
echo "</td>";
echo "<td>";
echo '<select name="tipo_formacion'.$registro['Id_becario'].'"'
onchange="enable_formador('.$registro['Id_becario'].')">';
$consulta_form = "SELECT * FROM tipo_formacion";
if ($resultado_form = mysqli_query($connect,$consulta_form)) {
    /* busca la matriz asociativa */
    while ($registro_form = mysqli_fetch_assoc($resultado_form)) {
        if ($registro_form['Id_tipo_formacion'] ==
$registro['Id_tipo_formacion']) {
            echo "<option value='".$registro_form
['Id_tipo_formacion']."' selected='selected'>".$registro_form
['Etiqueta']."</option>";
        }
        else {
            echo "<option value='".$registro_form
['Id_tipo_formacion']."'>".$registro_form['Etiqueta']."</option>";
        }
    }
    /*libera el objeto resultado */
    mysqli_free_result($resultado_form);
}
echo "</select>";
echo "</td>";
echo "<td>";
$consulta_todo_formador = "SELECT * FROM formador left join
sala on formador.id_sala = sala.id_sala";
if ($resultado_todo_formador =
mysqli_query($connect,$consulta_todo_formador)) {

    /* busca la matriz asociativa */
    while ($registro_todo_formador = mysqli_fetch_assoc($resultado_todo_formador)) {
        $checked="";
        $dt_inicio = date_create_from_format('Y-m-d',
strftime("%Y-%m-%d"));
        $dt_fin = date_create_from_format('Y-m-d',strftime("%Y-
%m-%d",strtotime('+6 month')));
        $consulta_formador = "SELECT * FROM becario_formador
INNER JOIN formador ON becario_formador.Id_formador =
formador.Id_formador INNER JOIN sala ON formador.Id_sala =
sala.Id_sala WHERE becario_formador.Id_becario =
".$registro['Id'];

        if ($resultado_formador =
mysqli_query($connect,$consulta_formador)) {
            /* busca la matriz asociativa */
            while ($registro_formador =
mysqli_fetch_assoc($resultado_formador)) {

```

```

        if ($registro_formador['Id_formador'] ==
$registro_todo_formador['Id_formador']) {
            $dt_inicio = date_create_from_format('Y-m-d',
$registro_formador['Fecha_inicio']);
            $dt_fin = date_create_from_format('Y-m-d',
$registro_formador['Fecha_fin']);
            $checked = "checked='checked'";
        }
    }
}

echo "<input type='checkbox' ".$checked." id='
formador".$registro_todo_formador['Id_formador']."_".
$registro['Id_becario']."' name='formador".$registro['Id_becario']."' .
"[]' value='".$registro_todo_formador['Id_formador']."' />";
echo $registro_todo_formador['Nombre']."' - ".
$registro_todo_formador['Etiqueta']."' - <input type='text' size='8'
name='inicio".$registro_todo_formador['Id_formador']."' _".
$registro['Id_becario']."' id='inicio".$registro_todo_formador
['Id_formador']."' _".$registro['Id_becario']."' value='"
.$dt_inicio->format('d/m/Y')."' onchange='comprueba_date(this.value)' />";
echo "<input type='text' size='8' name='fin".
$registro_todo_formador['Id_formador']."' _".$registro['Id_becario']."' .
$id='fin".$registro_todo_formador['Id_formador']."' _"
.$registro['Id_becario']."' value='".$dt_fin->format('d/m/Y').
"' onchange='comprueba_date(this.value)' /><br />";
}
/*libera el objeto resultado */
mysqli_free_result($resultado_todo_formador);
}
echo " </td>";
echo "<td><input type='checkbox' name='modificacion[]'
value='".$registro['Id_becario']."' /></td>";
echo "</tr>";
}
}

/* Cierre de la conexión */
mysqli_close($connect);

?>
</table>
<br />
<input type="submit" name="modificar" value="modificar" />
</form>
<br />
<a href="formacion.php">Añadir un becario</a> <a
href="lista_becario_a_eliminar.php">Eliminar un becario</a>

</body>
</html>

```

- La página modificacion\_becario.php cambia el becario en la base de datos:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<?php
$connect = mysqli_connect("127.0.0.1", "root", "", "formacion");

/* Comprobar la conexión */
if (!$connect) {
    echo "Fallo de la conexión : ".mysqli_connect_error();
    exit();
}

?>
<head>
    <title>Ejercicio formación</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-15" />

</head>
<body>

<?php
if (isset($_POST['modificacion'])) {
    date_default_timezone_set('Europe/Paris');
    foreach ($_POST['modificacion'] as $valor) {
        //Para modificar la tabla becario_formador, hay que
        //eliminar y crear los que están controlados
        $consulta = "DELETE FROM becario_formador WHERE Id_becario
= ".$valor;
        $resultado = mysqli_query($connect,$consulta);
        if (isset($_POST['formador'].$valor)) {
            foreach ($_POST['formador'].$valor] as $valor_formador) {
                $dt_inicio = date_create_from_format('d/m/Y',
$_POST['inicio'].$valor_formador."_".$valor]);
                $dt_fin = date_create_from_format('d/m/Y',
$_POST['fin'].$valor_formador."_".$valor]);
                $consulta_formador = "INSERT INTO becario_formador
(Id_becario, Id_formador, Fecha_inicio, Fecha_fin) VALUES
('".$valor."','".$valor_formador."','".$dt_inicio->format('Y/m/d')."',
'".$dt_fin->format('Y/m/d')."')";
                $resultado_formador =
mysqli_query($connect,$consulta_formador);
            }
        }
        $consulta_becario = "UPDATE becario SET
Apellido='".htmlentities(addslashes($_POST['apellido'].$valor)),
ENT_QUOTES)."'"
        ,Nombre=".htmlentities(addslashes($_POST['nombre'].$valor)),
ENT_QUOTES)."',Id_nacionalidad=".$_POST['nacionalidad'].$valor].",
Id_tipo_formacion=".$_POST['tipo_formacion'].$valor]." WHERE Id =
".$valor;
        $resultado_becario =
mysqli_query($connect,$consulta_becario);
    }
}

```

```
/* Cierre de la conexión */  
mysqli_close($connect);  
header('location:lista_becario_a_modificar.php');  
?>  
</body>  
</html>
```