PHP + SQL

Guía del curso

Guía del curso “PHP+SQL”, por Jorge Gil García (laertes). Grupo Universitario de Informática (GUI) de la Universidad de Valladolid. 10-20 de marzo de 2003.

1. Introducción.

En los últimos años hemos vivido una explosiva popularización de internet. Entre todos los servicios que La Red puede ofrecer los grandes triunfadores han sido los servicios web, los servicios de páginas html. Hasta tal punto son importantes estos servicios que muchos usuarios asocian internet únicamente a la consulta de páginas web.

Los proveedores de servicios, conscientes de esta realidad, han desarrollado sistemas basados en páginas web que sustituyen a muchos de los servicios clásicos: El correo web sustituye a los servicios POP, NNTP es sustituido por los foros de noticias web, los servidores web permiten la subida y bajada de archivos sustituyendo a los servicios FTP, etc.

Las páginas web han dejado de considerarse como lugares donde se consulta información estática para convertire en sitios dinámicos e interactivos. En la página de nuestra aerolínea ya no solo podemos consultar el horario de los vuelos, también podemos hacer reservas y compra de billetes. Cada vez que efectuamos una búsqueda en google o yahoo recibimos una página web única, creada por el servidor en el momento en el que pulsamos "buscar". Esto es interactividad.

El lenguaje HTML es un lenguaje de composición de texto, no de programación. HTML le dice al navegador como tiene que colocar los elementos de la página, con que colores, con que tipos de letra, etc.
Netscape y Microsoft han dotado a sus navegadores con la capacidad de interpretar lenguajes de programación, **JavaScript** en el caso de Netscape, **JScript** y **VBScript** en el caso de Microsoft. Es muy importante tener presente que **estos lenguajes se ejecutan en el navegador**, no en el servidor.

Para que un sitio web sea dinámico e interactivo es necesario que un programa, corriendo en el lado del servidor, recoja las peticiones del usuario y genere una página web personalizada. Estos programas se denominan **CGI**'s (**Common Gateway Interface**) y el desarrollo de los mismos "**Server Side Scripting**".

He aquí una lista de los lenguajes más comunes:

PHP Gran difusión, eficiente, buena curva de aprendizaje, GNU, multiplataforma.
ASP Gran difusión, eficiente, propietario de Microsoft, sólo en plataformas Microsoft.
JSP Basado en java, potente, estable y robusto. Alto consumo de recursos, especialmente bueno para sitios grandes, multiplataforma.
C++ Su única baza es su alta eficiencia. Hay que compilar cada vez que se hace un cambio. No está pensado para funcionar como CGI y su codificación para esta tarea es muy tediosa.
Perl Está siendo sustituido por lenguajes específicamente desarrollados para funcionar como CGI's.
.NET (C++, ASPX, C#, VBasic) Tecnología emergente de Microsoft. .NET es el interfaz entre el lenguaje y el código que se envía al navegador. Gran consumo de recursos. Sólo en plataformas Microsoft. El desarrollador puede elegir entre cuatro lenguajes. .NET compite con Java en el mismo "Nicho tecnológico".

En los sitios web con CGI's encontramos, casi invariablemente, bases de datos donde los programas depositan y recogen la información que necesitan. Las **bases de datos relacionales** están presentes de manera hegemónica. Estas permiten insertar y extraer datos de manera muy flexible y eficiente a requerimiento del programa. El lenguaje utilizado para realizar consultas se denomina **SQL** (Structured Query Language) y también es objeto de este curso.

El sitio web de la UVA (www.uva.es) funciona con PHP4, la base de datos MySQL y el servidor web de Netscape.

2. ¿Qué es PHP?

PHP (acrónimo recursivo: “PHP: Hypertext Preprocessor”) es un lenguaje de programación de código abierto, que se distribuye bajo licencia GNU, pensado para funcionar como CGI y que puede ser embebido en el código HTML.

Es un lenguaje sencillo de aprender, con una sintaxis basada en C, Java y Perl.

PHP es un lenguaje interpretado, no compilado, que permite hacer cambios rápidamente y con una sintaxis “suave”.

PHP puede ser programado “top-down” (para ser interpretado línea por línea, de arriba abajo), con funciones u orientado a objetos, según los requerimientos del sitio.

Una de las más importantes bazas del lenguaje es el amplio número de funciones implementadas (http://www.php.net/manual/en/funcref.php), entre las que podemos encontrar funciones para creación de archivos PDF, creación y modificación de imágenes, consultar bases de datos, crear sockets, acceso ftp, encriptación, etc.

3. Referencias

www.php.net El sitio oficial.
www.php.net/manual/en/ Manual online
www.php.net/manual/en/faq.php Cuestiones frecuentes
www.php.net/manual/en/funcref.php Referencia de funciones

El manual y la referencia de funciónes están disponibles en castellano (cambiar “en” por “es” en la url) La referencia en inglés es la recomendada, ya que siempre es la más actualizada.

4. Salida de texto.

4.1. “Hola, mundo”

```
<html>
<head>
        <title>Hola, mundo</title>
</head>
<body>

<?php
        print “Hola, mundo”;
?>

</body>
</html>
```

En este primer ejemplo vemos nuestro primer código PHP embebido dentro de HTML. Sólo el texto que se encuentra entre “<?php” y “?>” es interpretado por el servidor. Si analizamos el código fuente (ver->código fuente, en Explorer) observamos que sólo tenemos “Hola, mundo” en el “body”. Nuestro código PHP ha sido interpretado por el servidor y ha enviado al navegador únicamente el resultado del programa.

Ejercicio 1:

Comprueba que <? echo "Hola, mundo" ?> y <?= "Hola, mundo" ?> producen el mismo resultado.

4.2. Comentarios

Los comentarios son líneas de texto que se introducen en el código para referencia del programador. Estas líneas son ignoradas por PHP. Los comentarios de una sola línea deben empezar por "//". Pueden incluirse comentarios de varias líneas delimitados por "/**" y "**/".

```
<?
// Esto es un comentario de una línea

print "Hola mundo";

/*
Esta parte del código tampoco será interpretada.
print "Adios mundo";
*/
?>
```

4.3. El caracter de escape

El carácter "\" es un caracter de **escape**. Nos permitirá escribir comillas y caracteres especiales.

```
<?
print "Tenemos entrecomillada la palabra \"Comillas\" ";
print "<br>";
print " 'Comillas' está ahora entre comillas simples";
print "<br>";
print ' "Comillas" no está entre comillas \'simples\' ';
print "<br>";
print "Este \\ es el <b>carácter de escape</b>";
print "<br>";
print "Si miras el \"c&oacute;digo fuente\" de esta p&aacute;gina ver&aacute;s \n aquí
\n dos saltos de l&iacute;nea";
print "<br>";
?>
```

Ejercicio 2:

Pon en práctica este ejemplo y asegúrate de comprendes cómo funciona el caracter de escape.

4.4. Salida de texto con formato: printf

printf es una función común a muchos lenguajes de programación. Con *printf* podemos elegir el formato de salida.

printf (*formato, argumento1, arg2,...*)

Donde los argumentos pueden ser los siguientes:

% - un caracter literal de porcentaje. No se precisa argumento.

b - el argumento es tratado como un entero y presentado como un número binario.

c - el argumento es tratado como un entero, y presentado como el caracter con dicho valor ASCII.

d - el argumento es tratado como un entero y presentado como un número decimal.

f - el argumento es tratado como un doble y presentado como un número de coma flotante.

o - el argumento es tratado como un entero, y presentado como un número octal.

s - el argumento es tratado como una cadena y es presentado como tal.

x - el argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas).

X - el argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas).

Veamos un ejemplo:

```
<?

printf ("El %s decimal %s es el  %b binario y el %o octal.", "n&uacute;mero", 26, 26,
26);

?>
```

Ejercicio 3:

Familiarizate con printf. Usando como referencia la sentencia anterior representa el valor ASCII del caracter "*" y el valor binario y hexadecimal de 2002.

5. Variables.

Con PHP no es necesario definir variables. Las variables se asignan por referencia.

Todas las variables han de comenzar por un signo "$", seguido de una letra o "_". El resto de caracteres deben ser letras a-z, A-Z o caracteres ASCII del 127 al 255.

PHP distingue entre mayúsculas y minúsculas en el nombre de sus variables.

Probemos el siguiente ejemplo:

```
<?
$texto = "Hola";
$TEXTO = "mundo";

print "$texto, $TEXTO";
print "<br>";

$texto = "El texto es \"$texto, $TEXTO\"";
print $texto;
?>
```

En este ejemplo hemos creado por referencia las variables "$texto" y "$TEXTO", luego les hemos dado salida formando la frase "Hola, mundo". Finalmente hemos reasignado el valor de la variable "$texto" y le hemos dado salida.

PHP asigna los tipos de variable automáticamente. Puedes ver una lista en http://www.php.net/manual/en/language.types.php

6. Arrays

Un array es una asociación entre una **clave** y un **valor**. En PHP pueden definirse mediante la función **array()** o por referencia. Veamoslo de manera práctica.

```
<?
$a = array(    'color' => 'rojo',
                       'sabor' => 'dulce',
                       'forma' => 'esférica',
                       'nombre' => 'manzana',
                       4       // Su clave será 0
       );

print $a['color'];
print "<br>";
print $a['sabor'];
print "<br>";
print $a['forma'];
print "<br>";
print $a['nombre'];
print "<br>";
print $a[0];

/*
Salida :

rojo
dulce
esférica
manzana
4
*/

// Esta forma de definir el array es equivalente a esta :

$a['color']='rojo';
$a['sabor']='dulce';
$a['forma']='esférica';
$a['nombre']='manzana';
$a[0]=4;
?>
```

Cuando se define un array sin especificar la clave se le asignan automáticamente valores enteros.

```
<?
$b = array ('Cero', 'Uno', 'Dos', 'Tres');

print $b[0];
print "<br>";
print $b[1];
print "<br>";
print $b[2];
print "<br>";
print $b[3];
print "<br>";

/*
La salida de nuestro código es:

Cero
Uno
```

```
Dos
Tres

Esta forma de definir el array es equivalente a esta otra:
*/

$a[]='Cero';
$a[]='Uno';
$a[]='Dos';
$a[]='Tres';

//O a esta:

$a[0]='Cero';
$a[1]='Uno';
$a[2]='Dos';
$a[3]='Tres';
?>
```

El valor asociado a una clave puede ser otro array. Esto permite construir arrays multidimensionales.

```
$fruits = array ( "fruits"  => array ( "a" => "orange"
                                     , "b" => "banana"
                                     , "c" => "apple"
                                     )
                , "numbers" => array ( 1
                                     , 2
                                     , 3
                                     , 4
                                     , 5
                                     , 6
                                     )
                , "holes"   => array (      "first"
                                     , 5 => "second"
                                     ,      "third"
                                     )
                );
```

```
print $fruits['fruits']['a'];
print "<br>";
print $fruits['numbers']['3'];
print "<br>";
print $fruits['holes']['6'];

/*
Tiene como salida :

orange
4
third

Los arrays multidimensionales también pueden definirse por referencia.
*/
```

Más información en:
http://www.php.net/manual/en/language.types.array.php

Existe un juego de variables predefinidas. Puedes consultarlas en **http://www.php.net/manual/en/language.variables.predefined.php** Nos ocuparemos de ellas más adelante.

6.1. Funciones útiles

unset Eliminará la clave 2 del array. Este no será reindexado, es decir, el resto de claves conservarán sus valores.
sizeof Devuelve en número de claves definidas.
foreach Un bucle que recorre todos los valores definidios del un array.

```
<?
$a[0]='Cero';
$a[1]='Uno';
$a[2]='Dos';
$a[3]='Tres';

unset ( $a[2] );

foreach ( $a as $salida ){
        print $salida;
        print "<br>";
}

/*
Salida:
Cero
Uno
Tres
*/
?>
```

sort y **rsort** Ordenan los valores del array alfabéticamente en orden ascendente y descentente, respectivamente. Se altera la asociación entre clave y valor.
asort y **arsort** Igual que sort y rsort, pero manteniendo la asociación clave-valor.

En **http://www.php.net/manual/en/ref.array.php** puedes ver la referencia de funciones relacionadas con arrays.

Ejercicio 4:
Crea un array con varios valores y preséntalo en pantalla ordenado alfabéticamente tanto en orden creciente como decreciente utilizando "foreach".

Ejercicio 5:
Presenta el resultado del ejercicio 4 en forma de columnas, dentro de una tabla. Utiliza también "foreach".

Creciente	Decreciente
Morpheus	Trinity
Neo	Neo
Trinity	Morpheus

7. Estructuras de control

El código PHP, al igual que otros muchos lenguajes, se desarrolla mediante **sentencias** (statements). Una sentecia es un fragmento de código comprendido entre dos caracteres "**;**" (a excepción de la primera).

```
print "Hola"; print ", mundo";
```

La línea anterior está formada por dos sentencias. Un conjunto de sentencias agrupadas entre los caracteres "**{**" y "**}**" forman una nueva sentencia.

```
{print "Hola"; print ", mundo";}
```

Puedes encontrar información detallada sobre las estructuras de control en esta página:
http://www.php.net/manual/en/control-structures.php

7.1. if, else y elseif

```
if ( expresión )
        sentencia
elseif ( expresión )
        sentencia
else
        sentencia
```

"If" permite decidir si se ejecuta o no una sentencia a partir de una expresión. Si la expresión es cierta se ejecuta la sentencia, si no lo es se compara con el siguiente "elseif". Si la expresión del último "elseif" tampoco se cumple se ejecuta la sentencia asociada a "else".

```
<?
$a=2;
$b=3;

if ( $a > $b )
        print "$a es mayor que $b";
elseif ( $a == $b )
        print "$a es igual a $b";
else
        {
        print "La &uacute;ltima posibilidad:";
        print "<br>";
        print "$a es menor que $b";
        }
/*
Cambia los valores de $a y de $b para probar todas las
posibilidades.
*/
?>
```

Un tipo muy corriente de expresiones son las expresiones de comparación. Estas expresiones se evalúan a 0 o 1, significando FALSO (FALSE) o CIERTO (TRUE), respectivamente. PHP soporta **>** (mayor que), **>=** (mayor o igual que), **==** (igual que), **!=** (distinto), **<** (menor que) y **<=** (menor o igual que).

Asimismo existe la posibilidad de escribir expresiones lógicas.

ejemplo	nombre	resultado
$a and $b	Y	Cierto si tanto $a como $b son ciertos.
$a or $b	O	Cierto si $a o $b son ciertos.
$a xor $b	O exclusiva	Cierto si $a es cierto o $b es cierto, pero no ambos a la vez.
! $a	Negación	Cierto si $a no es cierto.
$a && $b	Y	Cierto si tanto $a como $b son ciertos.
$a || $b	O	Cierto si $a o $b son ciertos.

```
<?
$a=1;
$b=2;
$c=3;

if ( $a<$b && $b<$c )
       print "$a es menor que $b y $b es menor que $c<br>";
if ( $a<$b && ! ( $b<$c ) )
```

```
        print "$a es menor que $b y $b no es menor que $c<br>";
if ( ($a<$b && $a<$c ) || ! ( $a < 9 ) )
        print "$a es menor que $b y que $c o $a no es menor que 9<br>";

/*
Copia este ejemplo en tu página y ensaya con distintas condiciones.
Cambia los valores de $a, $b y $c para que encajen con las distintas posibilidades.
*/
?>
```

Más información sobre expresiones:
http://www.php.net/manual/en/language.expressions.php

Más sobre operadores:
http://www.php.net/manual/en/language.operators.php

7.2. Bucles "while"

```
while ( expresión )
        sentencia
```

Ejecuta la sentencia mientras la expresión sea cierta.

```
<?
$iterator=0;

while ( $iterator < 100 )
  {
  print "El iterador vale $iterator<br>";
  $iterator=$iterator+1;
  }
print "Fin del bucle while";
?>
```

Ejercicio 6:
Sustituye "$iterator=$iterator+1" por "$iterator++". Es una forma más cómoda de escribir lo mismo.
Repite el ejemplo anterior con saltos de 3 unidades.

7.3. Bucles "for"

```
for ( expresión1, expresión2, expresión3 )
     sentencia
```

Se evalúa la expresión1 una única vez. Al comienzo de cada iteración se evalua expresión2, si expresión2 es cierta se ejecuta la sentencia. Al final de cada iteración se ejecuta expresión3.

Con un ejemplo quedará mucho más claro:

```
<?
for ( $contador=1; $contador < 100; $contador++ )
{
        print "El contador vale $contador";
        print "<br>";
}
```

```
?>
```

Al empezar $contador toma el valor 1. Como 1 < 100 se ejecuta la sentencia y posteriormente se evalúa $contador++, de manera que $contador pasa a valer 2. Como 2<100 se repite el bucle. Cuando $contador=100 se rompe el bucle.

Ejercicio 7:
Utiliza bucles "for" para crear una columna de caracteres "***" coloreados desde el color "884400" hasta el color "8888FF".

Ejercicio 8:
Si XXYYZZ es un color en HTML, XX es el nivel de rojo, YY de verde y ZZ de azul.
Construye una tabla cuyas celdas tengan un nivel de rojo fijo: 88. El nivel de verde debe crecer desde la columna 1 (10) hasta la última columna (FF) con saltos de 4. Lo mismo debe ocurrir con el nivel de azul en las filas.

7.4. La sentencia "switch".

En ocasiones es necesario ejecutar distintas partes del código dependiendo del valor de una sola variable. Aunque puede hacerse con "if", "switch" ha sido creado para esta tarea.

```
<?
if ($i == 0) {
    print "i es igual a 0";
}
if ($i == 1) {
    print "i es igual a 1";
}
if ($i == 2) {
    print "i es igual a 2";
}

switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
}
?>
```

En este ejemplo la sentencia "if" y la sentencia "switch" son equivalentes.
No debemos olvidar "break" al final de cada "case". Si la sentencia "switch" fuera esta:

```
<?
switch ($i) {
    case 0:
        print "i es igual a 0";
    case 1:
        print "i es igual a 1";
    case 2:
        print "i es igual a 2";
 }
?>
```

Entonces, cuando $i=1, se ejecutaría la sentencia del “case 1” y todas las que están por debajo, probablemente esto no es lo que queremos.

“break” sirve para “romper” la ejecución de “switch” y de cualquier otra sentencia de control de estructuras. Puedes comprobar que “break” rompe también bucles “for” y “while”.

Más información sobre estructuras de control:
http://www.php.net/manual/en/control-structures.switch.php

8. Entrada de argumentos.

Hasta ahora hemos usado variables con valores definidas dentro del propio programa. Ahora veremos como introducir valores dentro de nuestros scripts.

Si conoces otros lenguajes de programación seguramente estarás familiarizado con “argc” y “argv”. PHP puede ejecutarse desde la línea de comandos e introducir valores por este método. Sin embargo nosotros utilizamos PHP como CGI, donde la entrada de argumentos es manejada por el navegador y el servidor web.

8.1. Los métodos “GET” y “POST”.

“GET” y “POST” son métodos propios del protocolo HTTP. Su funcionamiento es prácticamente idéntico, diferenciándose en detalles técnicos que no nos interesan. Bastará saber que con el método “POST” se pueden enviar al servidor un volumen de datos mucho mayor que con “GET”.

Existen dos formas de enviar datos por “GET”: Mediante URL y mediante formulario. Veamos dos ejemplos.

Creemos un script PHP llamado “escribe_datos.php”.

```
<?
print "dato1: $dato1";
print "<br>";
print "dato2: $dato2";
?>
```

Si ahora llamamos a nuestro archivo mediante la url de la siguiente manera:

http://3d.gui.uva.es/alu0/escribe_datos.php?dato1=primerdato&dato2=segundodato

El servidor web pasará a PHP el valor de los argumentos y PHP asignará a las variables $dato1 y $dato2 el valor que le hayamos asignado. Compruebalo.

También podemos utilizar un formulario.

```
<form action="escribe_datos.php" method=get>
Escribe dato1: <input type=text name=dato1 size=20>
<br>
Escribe dato2: <input type=text name=dato2 size=20>
<br>
<input type=submit value=Enviar>
```

```
</form>
```

El campo “action” indica a dónde hay que enviar los datos insertados en el formulario.

Llegado a este punto empezamos a ver interactividad. Los datos insertados por el usuario podrían ser almacenados en un archivo o en una base de datos. La página “escribe_datos.php” podría responder de manera distinta dependiendo de los datos insertados en el formulario.

El método “post” sólo se puede utilizar mediante formularios. Basta sustituir “method=get” por “method=post” en el encabezado del formulario. Es recomendable utilizar siempre este método.

```
Ejercicio 9:
Escribe una sencilla aplicación para transformar pesetas en euros y viceversa. ¿Serías
capaz de hacerla con botones?

Ejercicio 10:
Reescribe el ejercicio 8 de manera que el usuario pueda elegir, mediante un formulario,
los rangos de color  y el salto del bucle.
```

Por estos métodos sólo se pueden propagar variables, no arrays ni punteros ni “punteros a recursos”. Aunque existe un método, no muy conocido, para propagar arrays valor por valor con relativa comodidad.

```
<form action=”escribe_datos.php” method=post>
Nombre Alumno1: <input type=text name=”nombre[]” size=20>
<br>
Nombre Alumno2: <input type=text name=”nombre[]” size=20>
<br>
Nombre Alumno3: <input type=text name=”nombre[]” size=20>
<br>
Nombre Alumno4: <input type=text name=”nombre[ultimo]” size=20>
<br>
<input type=submit value=Enviar>
</form>
```

Al recibir este formulario PHP creará el array “nombre” y le asignará automáticamente los valores de cada campo con las claves 0,1,2 y “ultimo”. Compruebalo.

9. Manejo de cadenas.

Con PHP es muy frecuente tener que hacer manipulaciones con cadenas de texto. Veamos algunas de las funciones más importantes:

9.1. Concatenación

```
<?
$mundo = “mundo.”;
$texto = “Hola, “.$mundo;
print $texto;
?>
```

```
<?
$texto = “Hola, “;
$texto .= “mundo.”;
```

```
        print $texto;
        ?>
```

Los dos ejemplos anteriores producen el mismo resultado. En el primero anexamos la cadena contenida en la variable "$mundo", a la cadena "Hola, " y lo asignamos a la variable "$texto".
En el segundo usamos el operador ".=" para anexar la cadena "mundo." a la variable "$texto" que previamente habíamos definido con el valor "Hola, ".

9.2. explode (separador, cadena),

Genera un array resultado de dividir la cadena en fragmentos delimitados por el separador.

```
<?
$cadena = "uno, dos, tres, cuatro, cinco";
$array = explode ( ", ", $cadena );
foreach ( $array as $numero )
        print $numero."<br>";
?>
```

9.3. implode (pegamento, trozos)

Genera una cadena uniendo los elementos de un array con la cadena "pegamento".

9.4. str_replace (búsqueda, reemplazo, texto)

Genera una cadena con el texto "texto" donde todas las cadenas "búsqueda" han sido reemplazadas por las cadenas "reemplazo".
Si consultas la referencia de esta función verás que todos los elementos pueden ser arrays.

9.5. strip_tags (cadena)

Elimina todas las marcas html de la cadena. Esta función es especialmente util para evitar que un usuario que inserta un texto en un foro web, por ejemplo, altere la estructura de la página.

9.6. addslashes (cadena) y stripslashes (cadena)

Añade y elimina, respectivamente, caracteres "\" de la cadena. Sirve para controlar la inserción de caracteres de escape. Estas dos funciones son extremadamente importantes para la seguridad de las bases de datos.

9.7. strtolower (cadena) y strtoupper (cadena)

Devuelven la cadena con todos sus caracteres convertidos a minúsculas o mayúsculas, respectivamente.

9.8. strpos (cadena, búsqueda)

Devuelve la posición numérica de la cadena “búsqueda” en la cadena. La función **strrpos** efectúa la misma tarea empezando la búsqueda por el final de la cadena.

9.9. substr (cadena, inicio, fin)

Devuelve la parte de la cadena comprendida entre la posición “inicio” y “fin”.

```
<?
$rest = substr("abcdef", 1);    // devuelve "bcdef"
$rest = substr("abcdef", 1, 3); // devuelve "bcd"
$rest = substr("abcdef", 0, 4); // devuelve "abcd"
$rest = substr("abcdef", 0, 8); // devuelve "abcdef"

$rest = substr("abcdef", -1);    // devuelve "f"
$rest = substr("abcdef", -2);    // devuelve "ef"
$rest = substr("abcdef", -3, 1); // devuelve "d"

$rest = substr("abcdef", 0, -1);  // devuelve "abcde"
$rest = substr("abcdef", 2, -1);  // devuelve "cde"
$rest = substr("abcdef", 4, -4);  // devuelve ""
$rest = substr("abcdef", -3, -1); // devuelve "de"
?>
```

Ejercicio 11:
Escribe la lista de caracteres ASCII dentro de una tabla HTML. Los caracteres imprimibles son desde el 32 al 255.

Ejercicio 12:
Familiarizate con estas funciones. Crea un formulario donde el usuario pueda escribir un texto y efectuar sobre él cada una de las operaciones descritas en esta sección. El resultado debe aparecer en un campo “textarea”. No olvides consultar la referencia.

Más funciones de manejo de cadenas:
http://www.php.net/manual/en/ref.strings.php

10. Manejo de archivos

PHP ofrece funciones muy potentes para el manejo de archivos. Podemos manejar archivos locales tal y como permiten otros lenguajes de programación así como archivos remotos a los que se accede por http ó ftp prácticamente como si fueran locales.

10.1. fopen (archivo, modo)

Abre un archivo o una url. Si el archivo se abre con éxito devuelve un puntero, en caso contrario devuelve “FALSE”. El modo debe ser uno de los siguientes.

'r' - Abre para sólo lectura; sitúa el puntero del fichero al comienzo del mismo.
'r+' - Abre para lectura y escritura; situa el puntero del fichero al comienzo del fichero.
'w' - Abre para sólo escritura; sitúa el puntero del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
'w+' - Abre el fichero para lectura y escritura; sitúa el puntero del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
'a' - Abre sólo para escribir (añadir); sitúa el puntero del fichero al final del mismo. Si el fichero no existe, trata de crearlo.

'a+' - Abre para lectura y escritura (añadiendo); sitúa el puntero del fichero al final del mismo. Si el fichero no existe, trata de crearlo.

Si el archivo es local puede indicarse con una ruta tanto absoluta como relativa (al directorio del archivo PHP). La ruta puede escribirse tanto con el formato de Windows como de UNIX. Si el archivo es remoto debe indicarse el protocolo (http o ftp).

```
<?
$fp = fopen ( "/var/www/alu0/archivos/prueba.txt", "r");

/*
Hay que tener cuidado de usar barras dobles al usar el formato de windows
*/
$fp = fopen ( "\\var\\www\\alu0\\archivos\\prueba.txt", "r");
$fp = fopen ("archivos/prueba.txt", "r");
$fp = fopen ( "http://www.terra.es/", "r");
$fp = fopen ( "ftp://anonymous:test@ftp.gui.uva.es/pub/wireless/doc/IPSec/x509-
certificate-freeswan.html", "r");
?>
```

10.2. fclose (puntero)

Cierra el puntero a un fichero abierto.

10.3. feof (puntero)

Verifica si el puntero está al final de un archivo. Devuelve "FALSE" en caso afiramtivo y "TRUE" en caso contrario.

10.4. fread (puntero, longitud)

Lee "longitud" bytes del fichero apuntado. La lectura es con seguridad binaria.

10.5. Funciones de tipo "fget"

Extraen de distintas maneras el contenido de un archivo.

fgetc (puntero) Devuelve un caracter del fichero apuntado o FALSE si se ha alcanzado el fin del archivo.
fgets (puntero, longitud) Devuelve todos los caracteres hasta que encuentre un caracter nueva línea o hasta que se alcance la longitud, lo que ocurra primero.
fgetss (puntero, longitud) Igual que fgets, pero eliminando las etiquetas html.
fpassthru (puntero) Devuelve todos los datos desde la posición del puntero hasta el final del archivo.

Veamos unos ejemplos:

```
<?
if ( ! $fp = fopen ("http://www.google.com/index.html",  "r")){
        print "Error al abrir el archivo";
        exit;
}

while ( ! feof ($fp) ){
        $linea = fgets ($fp, 4096);
```

```
        $linea = str_replace ("Google", "www.uva.es", $linea);
        print $linea;
}

fclose ($fp);
?>
```

Abre el archivo "index.html" de www.google.com, lo recupera línea por línea y lo escribe reemplazando todas las ocurrencias de la cadena "Google" por "www.uva.es".

```
<?
if ( ! $fp = fopen ("http://www.google.com/index.html",  "r")){
        print "Error al abrir el archivo";
        exit;
}

while ( ! feof ($fp) ){
        $linea = fgetss ($fp, 4096);
        print $linea;
}

fclose ($fp);
?>
```

Recupera "index.html" eliminando todas las etiquetas html.

```
<?
if ( ! $fp = fopen ("http://www.google.com/index.html",  "r")){
        print "Error al abrir el archivo";
        exit;
}

fpassthru ($fp);
?>
```

No necesitamos cerrar el puntero ya que fpassthru lo cierra automáticamente. Consulta la referencia de "**readfile**". Con "readfile" no necesitaremos llamar a "fopen".

10.6. fwrite (puntero, cadena)

Escribe la cadena en el puntero. La escritura es con seguridad binaria.

Referencia de funciones de manejo de archivos:
http://www.php.net/manual/en/ref.filesystem.php

Consulta la referencia de funciones de manejo de archivos y hazte una idea de las las que existen.

Para alumnos avanzados: Con PHP se pueden establecer sockets de Internet o UNIX y se pueden tratar como si fueran archivos usando estas funciones. Consulta la referencia de "fsockopen".

Ejercicio 13.
Escribe un programa que recupere el contenido de www.google.com/index.html y lo escriba en un archivo local previa eliminación de las etiquetas html.

Ejercicio 14.
Crea un script que requiera los datos personales de los usuarios y los almacene en un archivo. Escribe otro programa que presente todos los datos almacenados dentro de una tabla.

11. Funciones definidas por el usuario.

En este capítulo veremos cómo crear nuestras propias funciones. Las funciones personalizadas nos facilitarán llevar a cabo tareas repetitivas y nos permitirán escribir un código mejor estructurado.

Una función se define de esta manera:

```
function mi_funcion ( $argumento_1, $argumento_2, … , $argumento_n){
      cuerpo de la función.
}
```

```
<?
/*
Definimos la función. Las funciones pueden definirse en cualquier lugar del programa.
*/

function mi_funcion ( $arg1, $arg2 ){

       // Cuando la función sea llamada escribirá esto.
       print "El primer argumento vale: $arg1<br>";
       print "El segundo argumento vale: $arg2<br>";

       // Y devolverá el valor de la suma.
       return $arg1+$arg2;
}

/*
Llamada a la función. El valor de retorno de la misma queda almacenado en la variable
"$salida".
*/

$salida = mi_funcion ( 4, 7);

print "La función ha retornado el valor: ".$salida;
?>
```

Es conveniente diseñar nuestras funciones para que retornen "false" en caso de error. Esto nos ayudará a prevenir errores en el cuerpo del programa.

```
<?
function mi_funcion ( $arg1, $arg2 ){

/*
Si los argumentos de la función no son numéricos devuelve
"false". return detiene la ejecución de la función.
*/
       if ( ! is_numeric($arg1) || ! is_numeric($arg2))
              return false;

       print "El primer argumento vale: $arg1<br>";
       print "El segundo argumento vale: $arg2<br>";
       return $arg1+$arg2;
}

/*
Si la función retorna "false" evitamos que se genere un mensaje erróneo.
*/
if ( $salida = mi_funcion ( 4, 7) )
       print "La función ha retornado el valor: ".$salida;
else print "Los argumentos no son válidos";
?>
```

11.1. Ámbito de las variables.

El ámbito de una variable definida dentro de una función es la propia función. La variable se destruye cuando la función termina. Además, las variables externas no son accesibles desde la función a menos que se indique explícitamente.

```
<?
$var = "Primera";

function funcion(){
        $var = "Segunda";
        print $var;
}

funcion();
print "-";
print $var;
/*
Devuelve "Segunda-Primera".
La definición de $var en la función no afectó al $var definido externamente.
*/
?>
```

Una variable definida como "**static**" no se destruye al finalizar la función, sino que conserva su valor mientras dura la ejecución del programa.

```
<?
function sumar(){
        static $a=0;
        print $a." ";
        $a++;
}

for ( $iter=0; $iter<10; $iter++)
        sumar();
/*
0 1 2 3 4 5 6 7 8 9
*/
?>
```

Se puede acceder a las variables externas desde la función de dos formas, mediante **global** o mediante el array asociativo **$GLOBALS[""]**.

```
<?
$uno = "uno";
$dos = "dos";
$tres= "tres";

function globales(){
        global $uno;
        print $uno."-";
        print $GLOBALS["dos"];
        print $tres;
}

globales();
/*
Devuelve: "uno-dos-".
*/
?>
```

Ejercicio 15:

Diseña un programa que presente una lista de números primos usando una función "es_primo" que tome un entero como entrada y que devuelva el mismo número si es primo o "false" en caso contrario. Utiliza la referencia de funciones matemáticas:
http://www.php.net/manual/en/ref.math.php

12. El lenguaje SQL.

Una base de datos relacional es un programa capaz de almacenar grandes cantidades de información y de ofrecerla de manera rápida con capacidad para combinar y ordenar los datos, ajustándose a los requerimientos del usuario.

SQL es el acrónimo de "Structured Query Language" (Lenguaje estructurado de consultas). Este es un estándar que utilizan las bases de datos relacionales para insertar, actualizar, eliminar y recuperar información.

Existen muchas bases de datos relacionales, las más difundidas son MySQL, Oracle, SQL Server y PostgresSQL. Cada una de ellas es compatible con el estándar SQL, aunque cada una de ellas tiene un juego de instrucciones propias.

12.1. Estructura de una BDD relacional.

Dentro de una BDD relacional la información se estructura en tablas, es decir, filas y columnas. Cada columna representa un "**campo**" y cada fila un "**registro**".

nombre	apellidos	puntuacion	sexo	fecha_de_nacimiento	Comunidad_id
José	Gutiérrez	57	H	1977-04-03	2
María	García	88	M	1980-12-09	3
David	Martínez	78	H	1976-11-02	8
Olga	Fernández	78	M	1977-12-03	1
Fernando	Ruiz	78	H	1979-01-04	2
Marta	García	25	M	1981-12-03	4

En esta tabla "Alumnos" tenemos los campos "Nombre", "Apellidos", "Edad", "Sexo", "comunidad_id" y los registros formados por los datos de los alumnos.

12.2. Los tipos de datos de MySQL

A la hora de crear las tablas es necesario indicar el tipo de datos que van a contener. Estos son los tipos más importantes que se pueden definir para MySQL. Los corchetes indican que el argumento es opcional.

- TINYINT[(M)] [UNSIGNED] [ZEROFILL]

Número entero. El rango con signo es de -128 a 127. El rango sin signo es de 0 a 255.

- SMALLINT[(M)] [UNSIGNED] [ZEROFILL]

Número entero. El rango con signo es de -32768 a 32767. El rango sin signo es de 0 a 65535.

- MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]

Número entero. El rango con signo es de -8388608 a 8388607. El rango sin signo es de 0 a 8388607.

- INT[(M)] [UNSIGNED] [ZEROFILL]

Número entero. El rango con signo es de -2147483648 a 2147483647. El rango sin signo es de 0 a 4294967295.

- BIGINT[(M)] [UNSIGNED] [ZEROFILL]

Número entero. El rango con signo es de -9223372036854775808 a 9223372036854775807. El rango sin signo es de 0 a 18446744073709551615.

- FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]

Número de punto flotante (decimal). Los valores válidos comprenden de -3.402823466E+38 a 1.175494351E-38, 0, y 1.175494351E-38 a 3.402823466E+38. Si se especifica "UNSIGNED" los valores negativos son descartados. M es el número de caracteres que se imprimirán cuando se haga la consulta y D el número de decimales que se representarán.

- DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]

Igual que "FLOAT", pero con sus valores comprendidos desde 1.7976931348623157E+308 a -2.2250738585072014E-308, 0, y 2.2250738585072014E-308 a 1.7976931348623157E+308

- DATE

Un campo de fecha con formato "AAAA-MM-DD". El rango válido es de "1000-01-01' a '9999-12-31"

- DATETIME

Campo de fecha y hora. Su formato es "AAAA-MM-DD HH:MM:SS". El rango válido es de "1000-01-01 00:00:00" a "9999-12-31 23:59:59".

.

- CHAR [(M)]

Campo alfanumérico. La longitud está fijada por "M".

- VARCHAR [(M)]

Campo alfanumérico de longitud variable. La longitud máxima está determinada por "M". El límite es 255 caracteres.

- BLOB y TEXT

Campos alfanuméricos largos. Las búsquedas efectuadas en un campo BLOB son sensibles a mayúsculas mientras que las búsquedas en un campo TEXT no lo son. La longitud máxima es de 65535. El campo BLOB suele utilizarse para almacenar archivos.

- MEDIUMBLOB y MEDIUMTEXT

Análogo a los campos BLOB y TEXT. Longitud máxima 16777215 caracteres.

- LONGBLOB y LONGTEXT

Análogo a los campos BLOB y TEXT. Longitud máxima 4294967295 caracteres.

Más información sobre los tipos de datos de MySQL:
http://www.mysql.com/doc/en/Column_types.html

Ejercicio 16.
Utiliza MySQLFront para crear una tabla "alumnos" capaz de albergar los datos de nuestra tabla de ejemplo. Los campos "nombre" y "apellidos" deben ser "VARCHAR(100)", "puntuación" "comunidad_id" deben ser "TINYINT UNSIGNED", "sexo" debe ser "CHAR(1)" y "fecha de nacimiento" del tipo "DATE".

12.3. Consultas.

12.3.1. Insert.

Inserta datos en una tabla. La sintaxis es:

```
INSERT [INTO] tabla [(columna1,columna2,…,columnaN)] VALUES (valor1,valor2,…,valorN)

INSERT INTO alumnos (nombre, apellidos) VALUES ('José, 'Gutiérrez')
```

Inserta un registro con "nombre" y "apellidos" el resto de campos queda con su valor por defecto.

```
INSERT INTO alumnos VALUES ('María', 'García', 88, 'M', '1980-12-9')
```

Inserta un registro con todos los campos. Nótese que el campo numérico no necesita comillas.

```
Ejercicio 17.
Inserta los datos de nuestra tabla de ejemplo en tu tabla "alumnos". Crea otra tabla con
dos campos, un campo numérico autoincrementable llamado "id" y otro de texto llamado
"nombre". Inserta los nombres de algunas comunidades autónomas.
```

Referencia:
http://www.mysql.com/doc/en/INSERT.html

12.3.2. Select

Extrae datos de una o varias tablas, combinando u ordenando entre otras tareas, si es necesario. Su sintaxis es:

```
SELECT expresión
       [FROM referencia_a_tablas]
              [[LEFT] [RIGHT] [INNER] JOIN referencia_a_tablas ON expresión_where]
              [WHERE expresión_where]
              [ORDER BY columna] [ASC o DESC]
              [LIMIT [límite,] registros]]
```

La expresión puede ser de distintos tipos. Las más sencillas son de carácter aritmético o lógico.

```
mysql> select 7+3
-> 10
```

Para extraer datos de tablas debemos escribir la expresión con la siguiente sintaxis:

```
SELECT talias1.campo1 [AS alias], talias2.campo2 [AS alias2],... from tabla1 [AS
talias1], tabla2 [AS talias1],...
```

Veamos algunos ejemplos:

```
SELECT apellidos, nombre FROM alumnos
```

Apellidos	Nombre
Gutiérrez	José
García	María
Martínez	David
Fernández	Olga
Ruiz	Fernando
García	Marta

En este caso no hace especificar "talias" porque sólo hay referencia a una tabla.
Si utilizamos "AS" cambia el nombre de las columnas devueltas.

```
SELECT apellidos AS ap, nombre AS nom FROM alumnos
```

ap	nom
Gutiérrez	José
García	María
Martínez	David
Fernández	Olga
Ruiz	Fernando
García	Marta

Mediante **CONCAT** podemos pedir que se nos devuelvan los datos en una sola columna.

```
SELECT CONCAT('Apellidos: ',apellidos,', 'Nombre: ',nombre') AS nombre_completo FROM
alumnos
```

Las funciones **SUM, AVG, MAX y MIN** devuelven la suma, el valor medio, el máximo y el mínimo de la columna seleccionada.

```
SELECT SUM(puntuacion) FROM alumnos
```

COUNT nos dirá el número de registros seleccionados.

```
SELECT COUNT(nombre) FROM alumnos
SELECT COUNT(*) FROM alumnos
```

LIMIT nos permitirá controlar el número de registros devueltos (limit es propio de MySQL).

```
SELECT * FROM alumnos LIMIT 2
```

Devuelve los dos primeros registros

```
SELECT * FROM alumnos LIMIT 3,2
```

Devuelve dos registros empezando por el cuarto (el primer registro es el 0).

Podemos utilizar **ORDER BY** para ordenar nuestros registros. El tipo de ordenamiento se elige automáticamente dependiendo del tipo de columna, será alfabético en un campo de texto, numérico en un campo numérico, etc. MySQL ordena de manera ascendente por defecto.

Con **ASC** y **DESC** forzamos el ordenamiento ascendente o descendente, respectivamente.

```
SELECT * FROM alumnos ORDER BY nombre
SELECT * FROM alumnos ORDER BY nombre DESC
SELECT * FROM alumnos ORDER BY fecha_de_nacimiento
SELECT * FROM alumnos ORDER BY puntuacion ASC, nombre DESC
```

La última consulta ordena los resultados por puntos de manera ascendente. Si dos alumnos tienen la misma puntuación los ordena de manera alfabética descendente.

Podemos hacer referencia a varias tablas para combinar los datos.

```
SELECT alumnos.apellidos, alumnos.nombre, meses.nombre FROM alumnos, meses
```

Extrae una tabla combinada.

Es posible usar alias para los nombres de las tablas:

```
SELECT a.apellidos, a.nombre, m.nombre FROM alumnos a, comunidades m
```

Esta consulta es equivalente a la anterior.

WHERE nos permite establecer condiciones de tipo aritmético o lógico en la consulta:

```
SELECT nombre, apellidos, puntuacion FROM alumnos WHERE puntuacion=88
```

```
SELECT nombre, apellidos, puntuacion FROM alumnos WHERE puntuacion>70
SELECT nombre, apellidos, puntuacion FROM alumnos WHERE puntuacion>70 AND sexo='M'
SELECT nombre, apellidos, puntuacion FROM alumnos WHERE puntuacion>70 AND ( sexo!='H' OR
fecha_de_nacimiento<'1977-1-1')
```

"**!=**" y "**<>**" son equivalentes y representan "distinto de".
"**%**" Es un caracter comodín que representa cualquier cadena. Se utiliza con **LIKE**.

```
SELECT apellidos FROM alumnos WHERE apellidos LIKE '%ez'
```

El elemento **JOIN** nos permitirá ajustar la manera en la que se combinan las tablas. Vamos a ver como funciona.

```
SELECT a.nombre, a.apellidos, a.comunidad_id, c.id, c.nombre FROM alumnos a LEFT JOIN
comunidades  c ON a.comunidad_id=c.id
```

"Alumnos" es la tabla de la izquierda. Esta consulta presenta completa la tabla "alumnos" asociando a cada registro de la misma otro registro de la tabla "comunidades" que cumpla la condición expresada en "on". Si en la tabla "comunidades" hubiera varios "id" idénticos "select" multiplicaría los registros coincidentes en "alumnos" el número de veces que fuera necesario.

"**RIGHT JOIN**" es análogo a "LEFT JOIN" intercambiando los papeles de la tabla izquierda y derecha.

"ON" admite las mismas condiciónes que "WHERE". Este tipo de consultas también admite "WHERE", "ORDER BY", "LIMIT", etc.

"**INNER JOIN**" presenta únicamente los registros cruzados coincidentes. En nuestro caso estas dos consultas son equivalentes.

```
SELECT a.nombre, a.apellidos, a.comunidad_id, c.id, c.nombre FROM alumnos a INNER JOIN
comunidades  c ON a.comunidad_id=c.id
```

```
SELECT a.nombre, a.apellidos, a.comunidad_id, c.id, c.nombre FROM alumnos a, comunidades
c WHERE a.comunidad_id=c.id
```

Referencia:
http://www.mysql.com/doc/en/SELECT.html

Más información sobre funciones propias de MySQL
http://www.mysql.com/doc/en/Functions.html

12.3.3. Update

Con esta instrucción podemos modficar el contenido de los registros. Su sintaxis es:

```
UPDATE tabla SET campo1=valor1, [campo2=valor2…] [WHERE sentencia_where]
```

La sentencia WHERE es la misma descrita para SELECT.
UPDATE también admite "ORDER BY" y "LIMIT".

Veamos un ejemplo.

```
UPDATE alumnos SET comunidad_id=1, puntuacion=80 WHERE fecha_de_nacimiento>'1978-1-1'
```

Asigna los valores "comunidad_id" y "puntuación" indicados a todos los alumnos nacidos después del 1-1-78.
Si no se especifica WHERE se actualizan todos los registros de la tabla.

12.3.4. Delete

```
DELETE FROM tabla [WHERE expresión_where]
```

Elimina todos los registros que encajen en la “expresión_where”. Si no se especifica WHERE se eliminan todos los registros de la tabla.

DELETE admite “ORDER BY” y “LIMIT”.

13. PHP+SQL

Veremos en este capítulo como utilizar PHP para trabajar con los registros de bases de datos relacionales. Casi en cualquier sitio construido con PHP encontraremos que trabaja contra una base de datos relacional. La posibilidad de utilizar PHP como interfaz con el usuario de una base de datos relacional dota a los sitios web de una gran interactividad.

Si consultamos la referencia de PHP veremos que posee librerías para trabajar con gran número de bases de datos, entre las que se encuentran Oracle, MySQL, SQLServer, mSQL, PostgreSQL, etc.

Nosotros trabajaremos contra MySQL. En la referencia encontraremos las funciones que necesitamos: **http://www.php.net/manual/en/ref.mysql.php**

13.1. Funciones

13.1.1. mysql_pconnect

```
mysql_pconnect([servidor [,login [,password ]]])
```

Abre una conexión persistente a la base de datos. Hay que especificar el servidor, nombre de usuario y contraseña. Si la base de datos es local la comunicación entre PHP y MySQL se realiza mediante un socket UNIX, en caso contrario se realiza sobre IP. Es válido especificar la ruta hasta el socket UNIX en “servidor”.

```
<?
$conn = mysql_pconnect('localhost','alu0','contraseña');
$conn = mysql_pconnect('127.0.0.1','alu0','contraseña');
$conn = mysql_pconnect('/var/run/mysqld/mysqld.sock','alu0','contraseña');
?>
```

Las tres funciones anteriores son equivalentes.

En caso de éxito en la conexión la función devuelve un “**RECURSO**”, un tipo especial de datos. Siempre que tengamos que hacer referencia a esta conexión deberemos utilizar la referencia al recurso almacenado en la variable “$conn”. Los recursos de PHP no son punteros.

En caso de fallo en la conexión la función devuelve **FALSE**. Esto lo podemos encontrar en casi todas las funciones.

13.1.2. mysql_select_db

```
mysql_select_db (base_de_datos [, identificador_de_recurso])
```

Establece el nombre de la base de datos con la que vamos a trabajar. Si no se especifica identificador de recurso se utiliza el devuelto por la última operación mysql_pconnect.

```
<?
mysql_select_db ('alu', $conn);
?>
```

13.1.3. mysql_query

```
mysql_query (consulta [, identificador_de_recurso])
```

Envía una consulta a la base de datos. Si no se especifica el identificador de recurso se utiliza el último abierto. mysql_query devuelve un vínculo a un recurso "consulta" o **FALSE**.

```
<?
$recurso_consulta = mysql_query ("select * from alumnos", $conn);
?>
```

13.1.4. mysql_fetch_row

```
mysql_fetch_row (recurso_consulta)
```

Recupera un registro de la base de datos y lo devuelve en un array. Las siguientes llamadas a mysql_fetch_row recuperarán los registros posteriores. La función devolverá **FALSE** cuando se hayan terminado los registros.

```
<?
$array = mysql_fetch_row ( $recurso_consulta );
?>
```

13.1.5. mysql_escape_string

```
mysql_escape_string (cadena)
```

Esta función es fundamental para la seguridad. Evita que un usuario malicioso inserte caracteres de escape dentro de una variable alterando así el comportamiento de la consulta. Cuando tengamos que hacer consultas con variables adquiridas por POST o GET debemos aplicar primero esta función antes de integrar las variables en la cadena de consulta.

```
$nombre = mysql_escape_string ($nombre);
```

13.2. Trabajando con la base de datos.

Es conveniente crear una o varias funciones que se encarguen del manejo de la base de datos, de esta manera el código quedará más limpio y cualquier error será subsanado rápidamente. El punto en el que PHP consulta con la base de datos es, con frecuencia, el lugar más vulnerable de un site, así que hay que ser cuidadosos a la hora de construir estas funciones.
Vamos a ver un ejemplo:

```
<?
define (DEBUG, 1); // Modo de depuración activado-desactivado
define (SERVER, 'localhost');
define (USER, 'alu0');
define (PASS, 'pass');
define (DB,   'alu0');

/*
Con esta función evitamos que puedan aparecer mensajes de error personalizados en modo
de depuración desactivado, así prevenimos que un posible atacante pueda recabar
información sobre el site. PHP posee funciones propias para controlar los mensajes de
error.
*/
function stderr($text){
        if (DEBUG==1){
                print "<br><font style='color: red'>";
                print $text;
                print "</font></br>";
        }
        return true;
}

/*
Esta función realiza todos los pasos necesarios para efectuar la consulta.
*/
function db_query($sql){
        if ( ! $conn = mysql_pconnect(SERVER, USER, PASS)){
                stderr("Error en la conexión");
                return false;
        }
        if ( ! mysql_select_db(DB,$conn) ){
                stderr("Error al seleccionar la base de datos");
                return false;
        }
        if ( ! $query = mysql_query($sql,$conn) ){
                stderr ("Error al efectuar la consulta: $sql");
                return false;
        }
        return $query;
}


// La variable "puntos" es introducida por el usuario.

print "<table border=1>\n";
$puntos=mysql_escape_string($puntos); // NUNCA debemos olvidar usar esta función
if ($query = db_query ("select * from alumnos where puntuacion>'$puntos'")){
        while ($row = mysql_fetch_row($query)){
                print "<tr>";
                foreach ( $row as $out )
                        print "<td>$out</td>";
                print "</tr>\n";
        }
        print "</table>\n";
}
/*
Los caracteres especiales \n nos ayudarán a la hora de examinar el código fuente de la
página generada.
*/
?>
```

Ejercicio 18
Desarrolla un iterfaz web para la base de datos. Crea una página donde los usuarios puedan insertar, modificar y eliminar los registros de la tabla "alumnos".