Programación Hipermedia I

Práctica 11: PHP 4 (acceso a una base de datos)

1. Objetivos

- Aprender a acceder a una base de datos desde PHP.
- Aprender a realizar una consulta INSERT, UPDATE y DELETE.
- Aprender a validar los datos recibidos desde un formulario web en PHP.

2. Recursos

¿Qué funciones se emplean en PHP para acceder a MySQL?

- **PHP Mysql**[1]: ext/mysql, la API original para usar MySQL desde PHP.
- **PHP Mysqli**[2]: ext/mysqli, la extensión mejorada para usar MySQL desde PHP.
- **PHP MySQL Introduction**[3]: tutorial de W3Schools sobre el uso de una base de datos de MySQL desde PHP.
- **ext/mysqli: Part I - Overview and Prepared Statements**[4] y **Using ext/mysqli: Part II - Extending mysqli**[5]: explica de forma detallada y clara el empleo de mysqli.

¿Existen funciones en PHP para filtrar los datos recibidos?

- **Data Filtering**[6]: documentación oficial del uso de las funciones de filtrado de PHP.
- **PHP Filter**[7]: tutorial de W3Schools sobre el uso de las funciones de filtrado de PHP.
- **PHP Filter Functions**[8]: referencia en W3Schools sobre el uso de las funciones de filtrado de PHP.

¿Cómo se emplean las expresiones regulares con PHP?

- **Regular Expressions (Perl-Compatible)**[9]: documentación oficial sobre el uso de la expresiones regulares con la sintaxis compatible con Perl.
- **PCRE - Perl Compatible Regular Expressions**[10]: documentación oficial sobre la biblioteca de funciones PCRE.
- **Regular Expression (POSIX Extended)**[11]: documentación oficial sobre el uso de expresiones regulares con la sintaxis POSIX.

[1] http://www.php.net/manual/es/book.mysql.php
[2] http://www.php.net/manual/es/book.mysqli.php
[3] http://www.w3schools.com/php/php_mysql_intro.asp
[4] http://devzone.zend.com/239/ext-mysqli-part-i_overview-and-prepared-statements/
[5] http://devzone.zend.com/255/using-ext-mysqli-part-ii_extending-mysqli/
[6] http://es.php.net/manual/es/book.filter.php
[7] http://www.w3schools.com/PHP/php_filter.asp
[8] http://www.w3schools.com/PHP/php_ref_filter.asp
[9] http://es.php.net/manual/es/book.pcre.php
[10] http://www.pcre.org/
[11] http://es.php.net/manual/es/book.regex.php

3. ¿Qué tengo que hacer?

En está práctica debes programar el registro de un nuevo usuario, es decir, la página de respuesta a la página "registro nuevo usuario". El registro se realiza mediante la inserción de los datos del nuevo usuario en la tabla que corresponda. En el lado del servidor con PHP tienes que volver a validar los datos introducidos por el usuario (aunque ya se está realizando la validación en el cliente mediante JavaScript, hay que volver a repetirla en el servidor como medida de seguridad); en concreto, el formulario para registrarse como un nuevo usuario contiene los siguientes campos y restricciones:

Página con el formulario de registro como nuevo usuario Contiene un formulario con los datos necesarios para registrarse (nombre de usuario, contraseña, repetir contraseña, dirección de email, sexo, fecha de nacimiento, ciudad y país de residencia, foto). Una vez enviado el formulario, debes realizar las siguientes comprobaciones en el servidor:

- **nombre de usuario**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas) y números; longitud mínima 3 caracteres y máxima 15.
- **contraseña**: sólo puede contener letras del alfabeto inglés (en mayúsculas y minúsculas), números y el subrayado; al menos debe contener una letra en mayúsculas, una letra en minúsculas y un número; longitud mínima 6 caracteres y máxima 15.
- **repetir contraseña**: su valor debe coincidir con el escrito en el campo **contraseña**.
- **dirección de email**: no puede estar vacío, hay que comprobar que cumple el patrón de una dirección de email (no permitir dominios principales de menos de 2 caracteres y más de 4 caracteres).
- **sexo**: se debe elegir un valor.
- **fecha de nacimiento**: comprobar que es una fecha válida.
- El resto de campos no indicados se pueden quedar vacíos.

Además, tienes que añadir dos opciones nuevas al menú de usuario registrado:

Página "Mis datos" Permite al usuario visualizar y modificar sus datos de registro. En el formulario de modificación se tienen que mostrar los datos actuales del usuario. Tanto en el formulario de modificación como en la página de respuesta se tienen que volver a realizar los filtrados definidos en la página con el formulario de registro como nuevo usuario. Por tanto, los más conveniente es aislar el código de filtrado (tanto el de cliente en JavaScript como el de servidor en PHP) en un fichero a parte e incluirlo donde sea necesario.

Página "Darme de baja" Elimina al usuario de la base datos. Antes de realizar, la operación debe solicitar la confirmación del usuario para evitar que se produzca la operación de forma accidental.

Por último, tienes que desarrollar las siguientes dos páginas para que funcionen las opciones de crear álbum y añadir foto a álbum:

Respuesta Página "Crear álbum" : realiza la inserción de un álbum nuevo en la base de datos.

Respuesta Página "Añadir foto a álbum" : realiza la inserción de una foto nueva en la base de datos. Por ahora, no se tiene que gestionar el almacenamiento de la foto subida.

En concreto, tienes que modificar o crear las páginas que se indican con un color de relleno oscuro en la Figura 1.

4. ¿Cómo lo hago?

4.1. Funciones de filtrado

Los datos que se reciben a través de Internet se tienen que validar y filtrar en el servidor, ya que pueden provenir de una fuente no segura. Aunque los datos provengan de un formulario donde se han verificado previamente con JavaScript, no hay que confiar en ello: el código JavaScript puede fallar, el navegador web puede ser que no ejecute el código JavaScript o puede que un usuario malintencionado con conocimientos envíe directamente datos incorrectos al servidor.

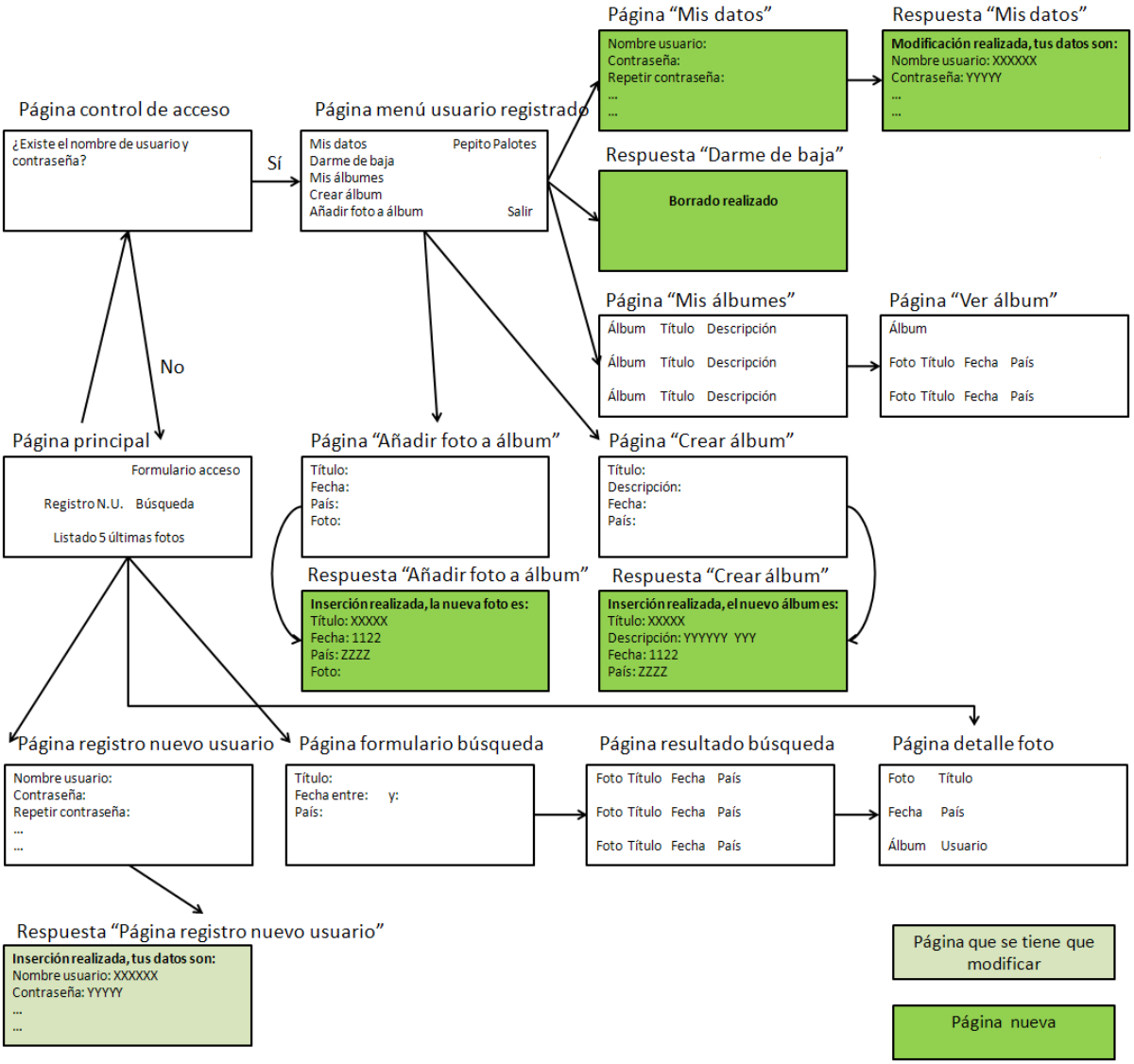

Figura 1: Diagrama de páginas que componen el sitio web

Por ejemplo, el complemento Tamper Data[12] para Mozilla Firefox permite visualizar y modificar los encabezados HTTP/HTTPS y los datos que se envían mediante post.

PHP 5 proporciona un conjunto de funciones básicas para validar y sanear los datos de entrada que pueden ayudar a filtrar los datos de entrada. Para validaciones más complejas será necesario desarrollar un conjunto propio de funciones de filtrado para los distintos casos que se pueden dar.

Las funciones de filtrado de datos en PHP son:

- `filter_has_var()`: verifica si la variable del tipo especificado existe.
- `filter_id()`: devuelve el ID del filtro con el nombre dado.
- `filter_input_array()`: obtiene múltiples variables desde afuera de PHP y opcionalmente las filtra.
- `filter_input()`: obtiene una variable desde afuera de PHP y opcionalmente la filtra.
- `filter_list()`: devuelve una lista de todos los filtros soportados.
- `filter_var_array()`: filtra múltiples variables con el mismo o con diferentes filtros.
- `filter_var()`: filtra una variable con un filtro específico.

Estas funciones se pueden emplear para realizar dos tipos de filtrados:

- Validación
  - Aplica reglas estrictas de formato (por ejemplo, URL o email).
  - Devuelve el dato original o `false` si no pasa el filtro.
- Saneamiento
  - Se emplean para permitir o no permitir ciertos caracteres en los datos.
  - No emplea reglas de formato.
  - Devuelve la cadena original tras eliminar los caracteres no permitidos.

Para indicar el tipo de filtrado a realizar (validación o saneamiento), se emplean unas constantes que se utilizan al llamar a las funciones de filtrado. Las principales constantes de filtros de validación son:

- `FILTER_VALIDATE_INT`: valida un entero.
- `FILTER_VALIDATE_IP`: valida una dirección IP.
- `FILTER_VALIDATE_FLOAT`: valida un número real.
- `FILTER_VALIDATE_URL`: valida una URL.
- `FILTER_VALIDATE_EMAIL`: valida un email.

Las principales constantes de filtros de saneamiento son:

- `FILTER_SANITIZE_STRING`: sanea una cadena.
- `FILTER_SANITIZE_NUMBER_INT`: sanea un entero.
- `FILTER_SANITIZE_NUMBER_FLOAT`: sanea un número real.
- `FILTER_SANITIZE_URL`: sanea una URL.
- `FILTER_SANITIZE_EMAIL`: sanea un email.

Además, existen una serie de opciones y *flags* para especificar información adicional en el proceso de filtrado, como las opciones `min_range` y `max_range` para definir intervalos numéricos o el *flag* `FILTER_FLAG_ALLOW_THOUSAND` para permitir el empleo de la coma como separador de miles en un número real.

Por ejemplo, en el siguiente código se realizan dos validaciones, una de número entero y otra de dirección IP, además se sanea un número real y un correo electrónico eliminando los caracteres que no son válidos:

[12] https://addons.mozilla.org/es-ES/firefox/addon/966

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de funciones de filtrado y saneamiento</title>
</head>
<body>
<p>
<?php
  // Filtra un número entero en un intervalo
  $int_options = array("options" => array("min_range" => 0, "max_range" => 256));
  $var = 123;
  if(filter_var($var, FILTER_VALIDATE_INT, $int_options))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto<br />";
  $var = 333;
  if(filter_var($var, FILTER_VALIDATE_INT, $int_options))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto<br />";

  // Filtra dirección IP
  $var = "125.125.125.125";
  if(filter_var($var, FILTER_VALIDATE_IP))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto";
  $var = "265.125.125.125";
  if(filter_var($var, FILTER_VALIDATE_IP))
    echo "$var: Correcto<br />";
  else
    echo "$var: Incorrecto<br />";

  // Sanea un número real
  $var = "1,345kg";
  echo "$var: " . filter_var($var, FILTER_SANITIZE_NUMBER_FLOAT) . "<br />";
  echo "$var: " . filter_var($var, FILTER_SANITIZE_NUMBER_FLOAT,
                             FILTER_FLAG_ALLOW_THOUSAND) . "<br />";

  // Sanea un correo electrónico
  $var = "sergio\.\lujan( @ )ua\.\es";
  echo "$var: " . filter_var($var, FILTER_SANITIZE_EMAIL) . "<br />";
?>
</p>
</body>
</html>
```

El ejemplo anterior produce la siguiente salida:

```
123: Correcto
333: Incorrecto
125.125.125.125: Correcto
265.125.125.125: Incorrecto
1,345kg: 1345
1,345kg: 1,345
sergio\.\lujan( @ )ua\.\es: sergio.lujan@ua.es
```

4.2. Expresiones regulares

Recordemos que una expresión regular es un patrón que define un conjunto de cadenas sin enumerar todos sus elementos. Una expresión regular está formada de caracteres (letras, números y signos) y metacaracteres que tienen una función definida (representar otros caracteres o indicar la forma de combinar los caracteres). Los metacaracteres reciben este nombre porque no se representan a ellos mismos, sino que son interpretados de una manera especial.

Los principales metacaracteres que se emplean en las expresiones regulares son:

- `^`: indica que el patrón que lo acompaña está al principio de la cadena.
- `$`: indica que el patrón que lo acompaña está al final de la cadena.
- `.`: representa cualquier carácter.
- `*`: el patrón que lo precede se repite 0 o más veces.
- `?`: el patrón que lo precede se repite 0 o 1 vez.
- `+`: el patrón que lo precede se repite 1 o más veces.
- `{x,y}`: el patrón que lo precede se repite un mínimo de `x` veces y un máximo de `y` veces.
- `|`: permite alternar expresiones.
- `[]`: indica un rango de caracteres válidos.
- `[^]`: indica un rango de caracteres no válidos.
- `()`: permite agrupar expresiones.
- \: permite escapar los metacaracteres para que aparezcan como literales.

En PHP existen dos conjuntos distintos de funciones que permiten utilizar las expresiones regulares, las funciones POSIX y las funciones PCRE. En algunos aspectos ambas funciones comparten la misma sintaxis, aunque por otro lado tienen sus particularidades propias.

Las funciones POSIX son `ereg()`, `eregi()`, `ereg_replace()`, `eregi_replace()`, `split()` y `spliti()`.

Las funciones PCRE (*Perl Compatible Regular Expressions*) implementan las expresiones regulares con la misma sintaxis y semántica que Perl 5. Estas funciones son más potentes (y por tanto, más complejas) y más rápidas que las funciones POSIX. Las funciones PCRE en PHP son: `preg_match()`, `preg_match_all()`, `preg_replace()`, `preg_split()` y `preg_grep()`. En las funciones PCRE la expresión regular debe empezar y acabar con una barra inclinada "/".

El siguiente ejemplo genera la página web que se muestra en la Figura 2; este ejemplo permite introducir una expresión regular, un texto y la función que se desea emplear (POSIX o Perl) y utiliza la función seleccionada para verificar si el texto introducido cumple la expresión regular:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de expresiones regulares</title>
</head>
<body>
<form action="expreg.php" method="post">
<p>
Expresión regular: <input type="text" name="expreg" size="20" /><br />
Ejemplo: ^[0-9]{1,4}$ o ^[^@ ]+@[^@ ]+\.[^@ .]+$<br />
Texto: <input type="text" name="texto" size="20" /><br />
Función:
<input type="radio" name="func" value="perl" /> Perl
<input type="radio" name="func" value="posix" /> Posix<br />
```

```
<input type="submit" value="Enviar" />
<input type="reset" value="Borrar" />
</p>
</form>
<p>
<?php
  if(isset($_POST['expreg']))
  {
    $expreg = stripslashes($_POST['expreg']);
    $texto = stripslashes($_POST['texto']);
    echo 'Expresión regular: ' . $expreg . '<br />';
    echo 'Texto: ' . $texto . '<br />';
    if($_POST['func'] == 'perl')
    {
      if(preg_match('/' . $expreg . '/', $texto) == 0)
        echo 'No';
      else
        echo 'Ok';
    }
    else if($_POST['func'] == 'posix')
    {
      if(ereg($expreg, $texto) == false)
        echo 'No';
      else
        echo 'Ok';
    }
    else
      echo 'Debe seleccionar una función correcta';
  }
?>
</p>
</body>
</html>
```

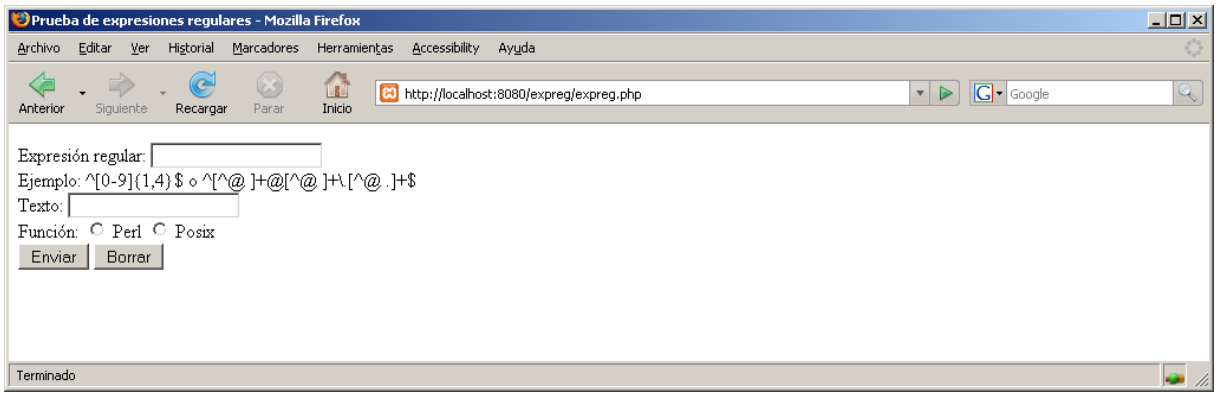

Figura 2: Ejemplo de uso de expresiones regulares en PHP

4.3. Inserción, modificación y borrado en una base de datos

La inserción (INSERT), la modificación (UPDATE) y el borrado (DELETE) de datos mediante SQL en una base de datos creada en MySQL se realiza de la misma forma que se ejecuta una consulta (SELECT), excepto que mientras una consulta devuelve un resultado, la inserción, la modificación y el borrado no devuelven un resultado. Si se quiere saber cuántas filas (tuplas o registros) se han visto afectadas por una inserción, una modificación o un borrado se puede emplear la función `mysqli_affected_rows()`.

Cuando se construya la sentencia SQL de inserción o modificación hay que recordar que los valores de tipo cadena y de tipo fecha deben estar encerrados entre comillas y que el orden del tipo fecha puede ser año-mes-día (depende del SGBD y de su configuración).

A continuación se muestra un ejemplo compuesto por dos páginas. La primera página muestra un formulario web que permite al usuario insertar datos en una tabla de una base de datos; la segunda página, programada en PHP, realiza la inserción de los datos recibidos desde el formulario en la base de datos.

El código de la página con el formulario es:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de INSERT y MySQL</title>
</head>
<body>
<h1>Insertar un libro nuevo</h1>
<form action="insert.php" method="post">
<p>
Título: <input type="text" name="titulo" /><br />
Resumen: <textarea name="resumen" rows="5" cols="80"></textarea><br />
Autor: <select name="autor">
<option value="3">Bob Clancy</option>
<option value="2">Fran London</option>
<option value="1">Luis Verne</option>
</select><br />
Categoría: <select name="categoria">
<option value="3">Aventuras</option>
<option value="1">Ciencia ficción</option>
<option value="2">Terror</option>
</select><br />
Editorial: <select name="editorial">
<option value="1">Escribe y publica</option>
<option value="2">Libros a GoGo</option>
<option value="3">Todo libros</option>
</select><br />
Año: <input type="text" name="anyo" size="4" maxlength="4" /><br />
<input type="submit" value="Enviar" />
<input type="reset" value="Borrar" />
</p>
</form>
</body>
</html>
```

El código de la página PHP (`insert.php`) que realiza la inserción en la base de datos es:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Prueba de INSERT y MySQL</title>
</head>
<body>
<p>
<?php
```

```
  // Recoge los datos del formulario
  $titulo = $_POST['titulo'];        // text
  $resumen = $_POST['resumen'];      // text
  $autor = $_POST['autor'];          // int
  $categoria = $_POST['categoria'];  // int
  $editorial = $_POST['editorial'];  // int
  $anyo = $_POST['anyo'];            // int

  // Se conecta al SGBD
  if(!($iden = mysqli_connect("localhost", "wwwdata", "")))
    die("Error: No se pudo conectar");

  // Selecciona la base de datos
  if(!mysqli_select_db($iden, "biblioteca"))
    die("Error: No existe la base de datos");

  // Sentencia SQL: inserta un nuevo libro
  $sentencia = "INSERT INTO libros VALUES (null, '$titulo', '$resumen', $autor, ";
  $sentencia .= "$categoria, $editorial, $anyo)";
  // Ejecuta la sentencia SQL
  if(!mysqli_query($iden, $sentencia))
    die("Error: no se pudo realizar la inserción");

  echo 'Se ha insertado un nuevo libro en la base de datos';

  // Cierra la conexión con la base de datos
  mysqli_close($iden);
?>
</p>
</body>
</html>
```

Después de cada operación con la base de datos hay que comprobar si se ha producido un error. Para ello, se puede comprobar el valor devuelto por las funciones de acceso a MySQL, ya que devuelven `false` si la operación ha fallado. Ante una situación de error, al usuario se le puede mostrar un mensaje de error propio o se puede emplear la función `mysqli_error()` que devuelve una descripción del último error producido.

5. Recomendaciones

Recuerda que las páginas que contengan código PHP tienen que tener la extensión `.php`. Si modificas alguna página web que ya tengas hecha de prácticas anteriores para añadirle código PHP, tendrás que cambiarle la extensión y corregir todos los enlaces que apunten a esa página.

Recuerda que para que el código PHP se ejecute, la página web tiene que pasar por el servidor web para que éste se la pase al intérprete de PHP. Para ello, tienes que cargar la página a través del servidor web: la URL de conexión tiene que tener la forma `http://localhost`.

El manual de PHP te lo puedes descargar en diferentes formatos de su sitio web[13] para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente. También puedes acceder a través de Internet a la ayuda de cualquier función de PHP escribiendo el nombre de la función a continuación de la URL `http://php.net/`. Por ejemplo, `http://php.net/header` muestra la ayuda de la función `header()`.

El manual de MySQL te lo puedes descargar en diferentes formatos de su sitio web[14] para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente.

No filtrar correctamente los datos de entrada es uno de los principales problemas de seguridad de las aplicaciones. Todos los datos que provengan de una fuente externa tienen que ser validados. En una aplicación web los datos pueden provenir de:

[13] http://www.php.net/download-docs.php
[14] http://dev.mysql.com/doc/

- Datos de un formulario.
- Cookies.
- Variables de entorno.
- Resultado de una llamada a un servicio web.
- Resultado de una consulta a una base de datos.
- Ficheros.

Las expresiones regulares son un mecanismo de programación muy potente, pero esa potencia origina que sean complejas de utilizar. Comienza a trabajar con expresiones regulares sencillas y poco a poco intenta escribir expresiones más complejas.

Es recomendable solicitar siempre una confirmación del usuario en aquellas operaciones que puedan modificar o eliminar los datos ya existentes en la base de datos. De este modo se evitará la pérdida accidental de información.

Recuerda que algunas funciones de PHP muestran directamente mensajes de advertencia o de error cuando se produce alguna situación errónea. Para evitar que suceda esto puedes emplear el operador de control de errores "`@`".

Evidentemente, antes de enviar los datos del formulario al servidor, se debe realizar en el cliente la validación mediante JavaScript, para posteriormente repetirla en el servidor con PHP. En esta práctica, para poder comprobar el correcto funcionamiento de la validación en el servidor, desactiva la validación en el cliente.