Acceso a los datos de MySQL desde PHP

Lenguajes Interpretados en el Servidor

Objetivos

- Hacer uso de los diferentes controles de formulario para enviar datos al servidor web.
- Validar datos introducidos en campos de formulario por los usuarios.
- Utilizar las distintas extensiones de PHP para acceder a los datos de una base de datos MySQL.
- Desarrollar habilidad para lograr conexión a una base de datos MySQL desde un script PHP.
- Hacer uso de funciones y clases de PHP para interactuar con los datos de una base MySQL.

Contenidos

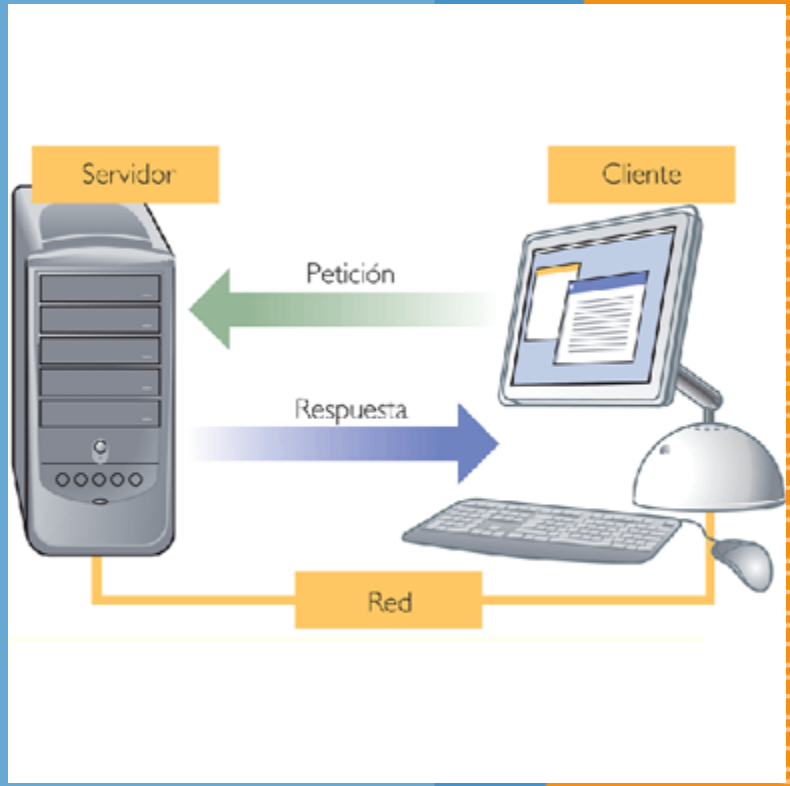
1. Formularios en PHP.
2. Matrices superglobables.
3. Validar datos provenientes de un formulario.
4. Gestionar campos vacíos.
5. Acceso a campos de múltiples valores.
6. Acceso a datos de MySQL desde PHP.
7. Extensiones de MySQL y PHP para conexión a bases de datos.
8. Uso de la extensión MySQL.
9. Uso de la extensión MySQLi.
10. Uso de la extensión PDO (*PHP Data Objects*).

Formularios en PHP

- Cuando se envía un formulario para su procesamiento en el servidor mediante un script, se produce un emparejamiento de los nombres de los controles con los respectivos valores actuales que poseen al momento del envío.
- Las parejas nombre_del_campo - valor constituyen el conjunto de datos del formulario que son enviados al servidor.

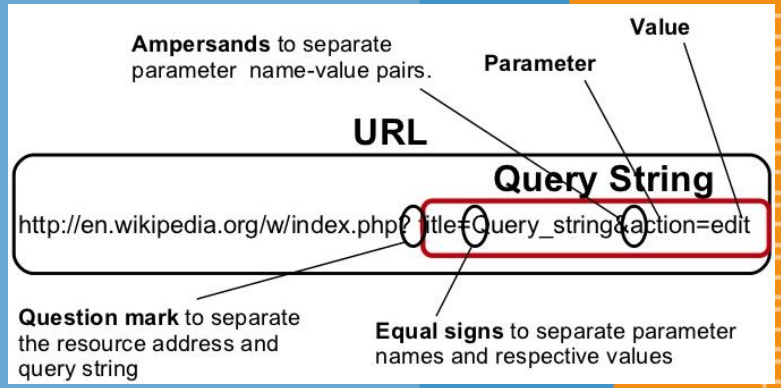
Formularios en PHP

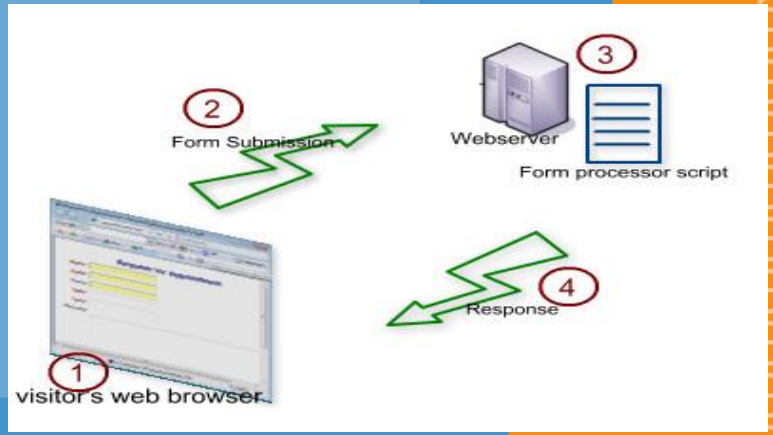
- El envío de estos datos, puede hacerse mediante uno de los métodos: GET, POST o REQUEST.
- Lo anterior significa que unas veces se enviarán formando parte de la **cadena de consulta** (*querystring*) de la URL que se envía al servidor y otras veces se enviarán como parte del **cuerpo del mensaje**.



Formularios en PHP

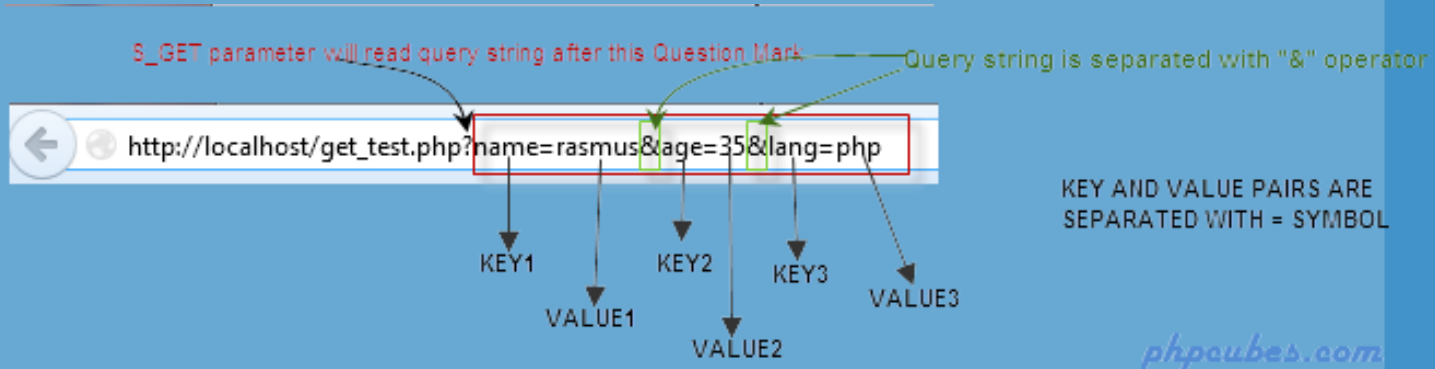
- PHP se encarga de recuperar el conjunto de datos que han sido enviados desde el cliente al servidor, proporcionando un conjunto de matrices superglobales.
- Esas matrices son tres principalmente:
  - _POST
  - _GET
  - _REQUEST





Formularios en PHP

- Ejemplo con *querystring*:



- Ejemplo con *querystring*:

Matrices superglobales

Matriz	Contenido
$_POST	Matriz asociativa que contiene las variables pasadas a través del método POST. También está disponible la matriz HTTP_POST_VARS, disponible por compatibilidad con versiones anteriores, pero obsoleta.
$_GET	Matriz asociativa que contiene las variables pasadas a través del método GET. También está disponible la matriz HTTP_GET_VARS de versiones anteriores, pero obsoleta.
$_REQUEST	Matriz asociativa que contiene las variables pasadas a través de cualquier mecanismo de entrada, diseñada para los desarrolladores que están más acostumbrados a usar ASP.

Ejemplo de paso de datos desde formulario

```
<!-- Formulario de envío con método GET -->
<form method="GET" action="formget.php">
<label for="modelo">Modelo:</label>
<input type="text" name="modelo" id="modelo" size="30" /><br />
<label for="marca">Marca</label>
<input type="text" name="marca" id="marca" size="30" /><br />
<label for="motor">Motor</label>
<input type="text" name="motor" id="motor" size="10" /><br />
<label for="combustible">Combustible</label>
<input type="radio" name="combustible" id="gas"  value="gasolina"
checked="checked" /><label>Gasolina</label>
<input type="radio" name="combustible" id="die" value="diesel"
checked="checked" /><label>Diesel</label>
</form>
```

Ejemplo de paso de datos desde formulario

```php
<?php
//Procesamiento del formulario en el servidor
$metodo = $_SERVER['REQUEST_METHOD'];
$cadenaconsulta = $_SERVER['QUERY_STRING'];
echo utf8_decode("<h2>Formulario enviado con método: $metodo</h2>");
echo "<span>QueryString:</span><strong>$cadenaconsulta</strong>";
echo "<p>";
//Método sofisticado y versátil
foreach($_GET as $campo => $valor){
   echo "$campo = $valor</br>";
}
//Método habitual y poco versátil
echo "</p><hr /><p>" . $_GET['marca'] . " - " . $_GET['modelo'] . " - ";
echo $_GET['motor'] . " - " . $_GET['combustible'] . "</p><hr />";
echo "<a href=\"javascript:history.go(-1)\">Volver</a>";
echo "</p>";
?>
```

Validar los datos de formulario

- Debido a que en el mundo de la web las entradas de usuario pueden provenir de usuarios maliciosos o mal intencionados, en una aplicación real de producción, no se debe, como programador, confiar en la entrada de datos de parte de éstos.
- Siempre debe dedicar tiempo y esfuerzo para validar, comprobar o filtrar los datos ingresados por el usuario, para asegurarse que se trata de datos válidos y no maliciosos antes de procesarlos en los scripts del servidor.

Gestionar campos vacíos

- Cuando se envían campos de formulario en los que el usuario debe ingresar datos (campos de texto, de contraseña, áreas de texto, etc) sucede con frecuencia que los usuarios dejen campos vacíos, o simplemente no los llenen.
- Cuando esto sucede, al enviar el formulario, puede ocurrir que el campo se envíe con un valor asociado de **cadena vacía** o que **no se envíe el campo completo** (**ni el nombre del campo**, **ni el valor asociado**).
- La siguiente tabla muestra cada caso dependiendo del tipo de campo de formulario.

Gestión de campos de formulario vacíos

Campo	Código HTML	Contenido
Campos de texto	<input type="text“ … />	Se envía el nombre del campo junto con un valor vacío.
Campos de contraseña	<input type="password" … />	Se envía el nombre del campo, junto con un valor vacío.
Casillas de verificación	<input type="checkbox" … />	No se envía nada (ni nombre de campo ni valor).
Botones de opción	<input type="radio" … />	No se envía nada (ni nombre de campo ni valor).
Botón enviar	<input type="submit" … />	Si se hace clic sobre el botón enviar, se enviarán el nombre y el valor asociado al botón. En caso de presionarse la tecla Intro (Enter), no se enviará nada, a menos que sea el único botón de envío disponible.
Botón cancelar	<input type="reset" … />	No se envía nada.
Campo selección de archivo	<input type="file" … />	Se envía el nombre del campo junto con un valor vacío. En este tipo de control debe utilizar la matriz superglobal $_FILES.
Áreas de texto	<textarea name="campo"></textarea>	Se envía el nombre del campo junto con un valor vacío.

Gestión de campos de formulario vacíos

Matriz	Contenido
Campo oculto	Se envía el nombre del campo junto con un valor vacío.
Campo de imagen	Si se hace clic sobre el botón del campo de imagen, se enviarán el nombre y el valor asociado al campo. En caso de presionarse la tecla Intro (Enter) no se enviará nada, a menos que sea el único botón de imagen disponible.
Menú desplegable	Imposible no seleccionar una opción, por ello siempre se enviará un valor junto al nombre del campo.
Cuadro de lista	No se envía nada.


```php
<?php
if ( isset ($_POST['submit'])) {  // Check for each form value when the form is submit
        $problem = FALSE; // no problems!
        if ( empty ($_POST['firstName'])) { // alert the user that they forgot to fill
                $problem = TRUE;
                print ("<p>You forgot to fill in your <b>&amp;quot;First Name&amp;quot;</b
                }
        if ( empty ($_POST['lastName']))  { // alert the user that they forgot to fill
                $problem = TRUE;
                print ("<p>You forgot to fill in your <b>&amp;quot;Last Name&amp;quot;</b>
```

Gestionar campos vacíos

- La importancia que tiene la gestión de campos vacíos es que cuando no se ingresa información en un campo de formulario, PHP no crea un elemento para la matriz $_POST, $_GET o $_REQUEST.
- Intentar acceder a un campo en estas condiciones genera un aviso de PHP (Notice).
- Debido a que es importante escribir código fuente que no genere avisos, ni advertencias conviene usar funciones de PHP que garanticen la presencia de un campo de formulario, antes de intentar usar su valor. Esas funciones son, entre otras: isset(), empty() y array_key_exists().

Acceso a campos de múltiples valores

- En el caso de campos de formulario que permiten múltiples valores, como los cuadros de lista con múltiple selección o un conjunto de casillas de verificación con el mismo nombre, es necesario indicar de alguna forma en el código HTML que se trata de un campo que posee múltiples valores, no solo uno:

```
<select name="marcas[]" multiple="multiple" size="3">
   <option value="Toyota">Toyota</option>
   <option value="Nissan">Nissan</option>
   <option value="Chevrolet">Chevrolet</option>
   <option value="Honda">Honda</option>
</select>
```

Acceso a campos de múltiples valores

- Notará que en el atributo name del elemento select se agregan unos corchetes al nombre del campo.
- El motor PHP al detectar la utilización de estos corchetes, crea un array asociado a la matriz super global $_GET, $_POST o $_REQUEST, en lugar de asignarle un valor único.
- De modo que en un script PHP, el acceso al elemento $_GET['marcas'] o $_POST['marcas'] da acceso a una matriz con todas las marcas seleccionadas en el campo de formulario de múltiple selección y no a un solo valor.
- Puede utilizar una estructura de control repetitiva para extraer uno a uno los elementos de la matriz.

Acceso a campos de múltiples valores

- Como sigue:

```
foreach($_POST['marcas'] as $marca){
   echo $marca . "<br />\n";
}
```

- En el caso de utilizar campos de casillas de verificación con el mismo nombre, el marcado HTML debería ser:

```
<input type="checkbox" name="accesorios[]" value="CD Player" />
<label>CD Player</label>
<input type="checkbox" name="accesorios[]" value="Aire acondicionado" />
<label>Aire acondicionado</label>
<input type="checkbox" name="accesorios[]" value="Bolsa de aire" />
<label>Bolsa de aire</label>
/*****************************/
foreach($_POST['accesorios'] as $valor){
   $unordlist .= "$valor< br />\n";
}
```

Acceso a datos de MySQL desde PHP

- Los pasos para acceder desde una secuencia de comandos PHP a una base de datos MySQL son:
  1. Configurar la conexión a la base de datos. Esto requiere especificar el usuario y la contraseña de acceso a la base de datos, además del nombre del servidor donde está alojado.
  2. Seleccionar la base de datos con la que se va a trabajar. Dependiendo del método, puede ser que este paso se realice junto al primero que mencionamos antes.

Acceso a datos de MySQL desde PHP

3. Construir la instrucción MySQL que se va a ejecutar en el gestor, siendo recomendable asignarla a una variable.
4. Enviar la instrucción al servidor MySQL para que se ejecute.
5. Procesar el resultado en la secuencia de comando PHP para mostrar los resultados en la página web.
6. Liberar los recursos utilizados por la conexión y en la consulta.
7. Cerrar la conexión.

Extensiones de MySQL y PHP para acceso a los datos de MySQL

- La forma de conectarse a MySQL desde PHP ha tenido cambios desde la primera extensión creada, conocida como extensión MySQL.
- Existen en la actualidad, tres formas de acceder desde PHP a datos de una base de datos MySQL. Estas formas son conocidas como extensiones:
  - Extensión MySQL.
  - Extensión MySQLi.
  - Extensión PDO (PHP Data Objects).

Uso de la extensión MySQL

- Esta extensión es la tradicional forma de conexión a bases de datos MySQL desde PHP.
- El proceso para acceder a datos de MySQL desde PHP utilizando esta extensión requiere los siguientes pasos:
  1. Configurar la conexión a la base de datos.
  2. Seleccionar la base de datos de trabajo.
  3. Construir la instrucción MySQL que se va a ejecutar en el gestor.
  4. Enviar la instrucción al servidor MySQL para que la ejecute.
  5. Procesar el resultado en la secuencia de comando para mostrar los resultados en la página web.
  6. Liberar los recursos utilizados por la conexión y los resultados devueltos por la consulta.
  7. Cerrar la conexión.

Configuración de la conexión

- Para realizar la conexión con el gestor desde PHP se utiliza la función mysql_connect().
- La sintaxis es la siguiente:

```
resource mysql_connect([string $host[,string $user[,string $password]]])
```

- Los argumentos de la función son el `$host` que es el nombre o IP del equipo donde está el gestor de la base de datos. Este argumento, también puede incluir un número de puerto. En el caso de que sea el mismo servidor donde está alojado el script, se utilizará el valor por defecto, localhost:3306. El valor devuelto es un identificador de conexión. De no lograrse conexión con el servidor, la función devolverá falso.

Configuración de la conexión

- El parámetro `$user` es el nombre de usuario configurado en la base de datos. Por defecto es root.
- El parámetro `$password` es la contraseña asignada en el servidor MySQL al usuario indicado en el segundo parámetro. En la mayor parte de instalaciones MySQL, para el usuario root es cadena vacía, algo que representa un serio riesgo de seguridad. Tendrá que asignar una.

Configuración de la conexión

- Si se estableció correctamente la conexión `mysql_connect()` devuelve un identificador de conexión que puede utilizarse posteriormente, en caso de que sea necesario distinguirlo de otro.
- Si la conexión falla, entonces devuelve False.
- La conexión se cierra automáticamente al terminar la ejecución del script o al utilizar la función `mysql_close()`.

Seleccionar la base de datos

- Para seleccionar la base de datos de trabajo se utiliza la función mysql_select_db().
- La sintaxis es la siguiente:

```
bool mysql_select_db(string $db[,resource $link=NULL]);
```

- La función establece la base de datos activa en el servidor asociada con el identificador de enlace especificado en el segundo argumento. Si se logra conectar devuelve True, si no lo consigue, devuelve False.
- El primer argumento, $db es el nombre de la base de datos que se intenta seleccionar en el servidor MySQL.
- El segundo argumento, $link es el identificador de enlace devuelto por la función `mysql_connect()`.

Preparación de la consulta

- Este proceso consiste en construir la sentencia SQL que se enviará al servidor MySQL para obtener la información que se requiere mostrar en la página web.
- Este proceso puede ser tan fácil como asignar a una variable PHP una sentencia de forma directa, como:

```
$qr = "SELECT * FROM materias";
```

- O tan compleja como realizar operaciones de concatenación con datos provenientes de formularios. Por ejemplo:

```
$qr = "SELECT * FROM libros WHERE " . $tipobusqueda . " LIKE
'%" . $criteriobusqueda . "%'";
```

Enviar la consulta al servidor

- Una vez creada la consulta, se puede enviar para su ejecución en el servidor MySQL utilizando la función `mysql_query()`.

```
resource mysql_query(string $query[,resource $link=NULL]);
```

- La función recibe dos parámetros. El primero es precisamente, la cadena de consulta, `$query` y, el segundo, el identificador de conexión, `$link`, el cual, si se está trabajando con una sola conexión, es opcional.
- En caso que la sentencia SQL enviada al servidor sea una consulta de datos (SELECT) o un comando de consulta específico de MySQL (SHOW, DESCRIBE, EXPLAIN) la función devolverá un identificador de cursor.

Enviar la consulta al servidor

- El resultado de la ejecución de la consulta está almacenado en este identificador y para poder acceder a esta información se requerirá de un procesamiento antes de poder mostrarla en la página web.
- En caso que la sentencia SQL ejecutada sea un INSERT, UPDATE, DELETE, CREATE, etc. La función devuelve TRUE si se ha realizado correctamente o FALSE si no ha podido hacerlo.

Procesar la información

- El procesamiento de la información almacenada en el cursor puede hacerse utilizando una de cuatro funciones:
  1. mysql_fetch_row().
  2. mysql_fetch_assoc().
  3. mysql_fetch_array().
  4. mysql_fetch_object().
- Todas las funciones anteriores están diseñadas para recorrer el cursor que contiene los datos devueltos por la consulta y avanzan de forma automática a la siguiente posición de la fila o tupla activa en el cursor. Cada vez que se accede a una nueva fila se devuelve un array del registro.

Procesar la información

- Para recorrer todas las filas resultantes de la consulta que están almacenadas en el cursor se suele hacer uso de una sentencia repetitiva en conjunto con una de las funciones anteriores. La construcción de una sentencia repetitiva para este propósito puede hacerse de la siguiente forma:

```
while($fila = mysql_fetch_array($rs)){
    echo $fila['campo1'] . "|" . $fila['campo2'];
    //resto de instrucciones;
}
```

- El acceso a los campos de cada registro para esta función puede hacerse a través del nombre del campo o mediante un índice numérico.

Procesar la información

- En el caso de utilizar mysql_fetch_row(), el acceso a los campos de cada fila o registro se debe hacer mediante un índice numérico comenzando desde cero para el primer campo indicado en la sentencia SELECT, siguiendo con uno, para el segundo y así sucesivamente:

```
while($fila = mysql_fetch_row($rs)){
   echo $fila[0] . "|" . $fila[1];
   //resto de instrucciones;
}
```

- El acceso a los campos de cada registro para esta función únicamente puede hacerse con un índice numérico.

Procesar la información

- En el caso de utilizar mysql_fetch_assoc(), el acceso a los campos de cada fila o registro se realiza mediante un índice asociativo en donde la cadena utilizada como índice debe ser el nombre de cada campo indicado en la sentencia SELECT:

```
while($fila = mysql_fetch_assoc($rs)){
   echo $fila['campo1'] . "|" . $fila['campo2'];
   //resto de instrucciones;
}
```

- El acceso a los campos de cada registro para esta función únicamente puede hacerse con un índice asociativo.

Procesar la información

- En el caso de utilizar mysql_fetch_object(), el acceso a los campos de cada fila o registro se se realiza mediante una sintaxis de objetos, en donde el nombre del campo se indica como una propiedad de objeto:

```
while($fila = mysql_fetch_object($rs)){
    echo $fila->campo1 . "|" . $fila->campo2;
    //resto de instrucciones;
}
```

- El acceso a los campos de cada registro para esta función debe hacerse con notación de objetos, en donde la referencia a cada campo debe hacerse como una propiedad de objeto.

Liberar recursos

- Con **`mysql_free_result()`** se liberan los recursos de la memoria utilizados durante la consulta SQL, en especial el espacio asignado a las filas devueltas en el cursor.
- La función devuelve True si se han podido liberar los recursos o False si no ha sido así.
- Cuando se intenta utilizar esta función después de ejecutar una sentencia de tipo INSERT o UPDATE PHP devolverá un error de nivel E_WARNING por el hecho de que en este caso mysql_query() no devuelve ningún cursor, como se mencionó antes.

Cerrar la conexión

- `mysql_close()` es la función que se debe utilizar para cerrar la conexión con el gestor MySQL.
- El único argumento que requiere esta función es el identificador de conexión creado después de la ejecución de la función `mysql_connect()`. El argumento es opcional, y si no se usa se utilizará la conexión activa.
- La conexión será cerrada por PHP al terminar el script aunque no se utilice esta función.

Uso de la extensión MySQLi

- La extensión MySQLi, conocida también como MySQL mejorada, viene disponible en las versiones PHP 5.0 o posteriores.
- Esta extensión fue desarrollada para aprovechar las nuevas funcionalidades disponibles en MySQL a partir de la versión 4.1.3.
- Comparada con la extensión MySQL, MySQLi ofrece las siguientes mejoras:
  - Interfaz orientada a objetos y por funciones.
  - Soporte para consultas preparadas.
  - Soporte para transacciones.
  - Soporte para múltiples declaraciones.
  - Mayor seguridad en las conexiones.

Uso de la extensión MySQLi

- Los pasos básicos de acceso a una base de datos MySQL, desde una secuencia de comando PHP, son los siguientes:
  1. Comprobar y filtrar los datos ingresados por el usuario.
  2. Configurar la conexión a la base de datos.
  3. Establecer la base de datos de trabajo.
  4. Preparar la consulta que se desea ejecutar.
  5. Ejecutar la consulta en el servidor MySQL.
  6. Recuperar los datos devueltos por la consulta ejecutada.
  7. Cerrar la conexión con la base de datos.

Comprobar y filtrar datos

- En toda aplicación web que requiera ingreso de datos de parte del usuario a través de un formulario es recomendable comprobar y filtrar los datos ingresados antes de que sean enviados al gestor de base de datos.
- Las comprobaciones mínimas necesarias son:
  1. Comprobar espacios en blanco innecesarios y eliminarlos utilizando la función `trim()`.
  2. Escapar caracteres de control que puedan haber sido ingresados desde la interfaz de usuario utilizando funciones como `addslashes()`, `stripslashes()` o `get_magic_quotes()`.

Configurar la conexión

- La extensión MySQLi brinda la posibilidad de conectarse a MySQL usando una sintaxis orientada a objetos o basada en funciones o procedimientos.
- Orientada a objetos:

```
$db = new mysqli("localhost", "root", "", "base");
```

- Basada en procedimientos:

```
$db = mysqli_connect("localhost", "root", "", "base");
```

Configurar la conexión

- En el caso del enfoque orientado a objetos se crea una instancia de la clase MySQLi y una conexión al host localhost con el nombre de usuario y contraseña proporcionados.
- La conexión se configura para que utilice la base de datos proporcionada en el cuarto argumento del constructor.
- Al utilizar este enfoque se pueden invocar métodos que darán acceso a la base de datos.

Configurar la conexión

- En el caso de utilizar el enfoque por funciones, se devuelve un puntero, en lugar de un objeto.
- El puntero representa la conexión a la base de datos que deberá ser pasado como argumento en todas las funciones mysqli restantes. Su comportamiento es similar al funcionamiento del descriptor de archivos utilizado en las funciones de procesamiento de archivos, como fopen().

Configurar la conexión

- Cualquiera que sea el enfoque utilizado, orientado a objetos o por funciones, siempre es aconsejable comprobar el intento de conexión y así evitar trabajar en vano con el resto de las funciones de acceso a datos.
- El siguiente bloque de código realiza esta comprobación:

```
if(mysqli_connect_errno()){
    echo "Error: no es posible conectar con el servidor";
    exit();
}
```

- La función `mysqli_connect_errno()` devuelve un número de error o cero si no se produce ninguno.

Establecer la base de datos

- Aunque en la extensión MySQLi se ha incluido la base de datos predeterminada como argumento del constructor mysqli o de la función mysqli_connect(), siempre se puede cambiar la base de datos de trabajo usando el método select_db() del objeto conexión a la base de datos:

```
$db->select_db("base");
```

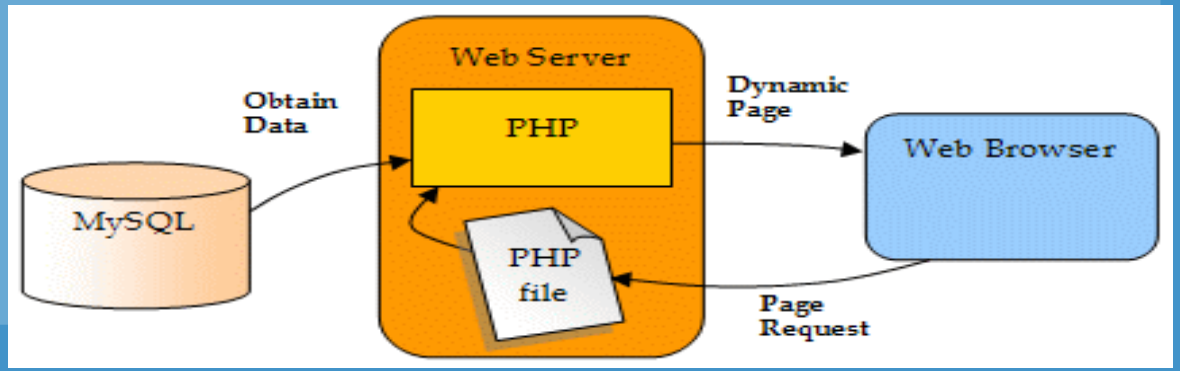
- O usando la función mysqli_select_db() en caso de estar usando el enfoque por funciones:

```
mysqli_select_db($db, "base");
```

Preparar la consulta

- Siempre es aconsejable asignar la consulta SQL que se desea ejecutar en el servidor a una variable, con el propósito de utilizar esta variable como argumento del método query() en el objeto conexión o de la función mysqli_query() en el caso de estar trabajando con un enfoque por funciones.

```
$qr = "SELECT * FROM base WHERE campo LIKE '%criterio%'";
```



Ejecutar la consulta

- Para enviar la consulta SQL al servidor MySQL para su ejecución puede seguir el enfoque orientado a objetos usando el método query():

```
$rs = $db->query($qr);
```

- O el enfoque por funciones usando la función mysqli_query():

```
$rs = mysqli_query($db, $qr);
```

Recuperar los datos de la consulta

- Para procesar el conjunto de resultados devuelto por la consulta se pueden utilizar varios métodos del objeto resultados, en el caso de trabajar con el enfoque orientado a objetos o varias funciones en el caso de trabajar con un enfoque orientado a funciones:

```
$row = $rs->fetch_assoc();
```

- O usando la función mysqli_fetch_assoc() en caso de estar usando el enfoque por funciones:

```
$row = mysqli_fetch_assoc($db, "base");
```

Recuperar los datos de la consulta

- Un paso necesario después de recuperar los resultados es determinar el número de filas (registros) que ha devuelto la consulta. Si se utiliza el enfoque orientado a objetos se puede consultar la propiedad `num_rows` del objetos resultados:

```
$filas = $rs->num_rows;
```

- O usando la función `mysqli_num_rows()` en caso de estar usando el enfoque por funciones:

```
$filas = mysqli_num_rows($rs);
```

Recuperar los datos de la consulta

- Lo anterior permitirá recorrer fila por fila los resultados haciendo uso de un ciclo, lazo o bucle:

```
for($i=0; $i<$filas; $i++){
    //Procesar registro por registro en una lista o tabla
    //HTML para mostrar los resultados en un navegador.
}
```

Ejemplo por funciones

- El siguiente ejemplo muestra cómo utilizar la extensión MySQL, utilizando un enfoque por funciones:

```
$cn = mysqli_connect("localhost","root","","base",3306);
$rs = mysqli_query($cn, "SELECT * FROM clientes");
while($row = mysqli_fetch_array($rs)){
   foreach($row as $key => $value){
      echo $key . ":" . $value . "<br />\n";
   }
}
mysqli_free_result($rs);
mysqli_close($cn);
```

Ejemplo orientado a objetos

- También se puede hacer uso de un enfoque orientado a objetos para acceder a los datos:

```
$cn = new mysqli("localhost","root","","base",3306);
$rs = $cn->query("SELECT * FROM clientes");
while($row = $rs->fetch_array()){
   foreach($row as $key => $value){
      echo $key . ":" . $value . "<br />\n";
   }
}
$rs->close();
$cn->close();
```

FIN

Lenguajes Interpretados en el Servidor