Contenido

Introducción

Caso práctico

Luisa está sorprendida de la cantidad de tecnologías web que ha visto.

Juan: ¿Y que estáis viendo ahora?

Luisa: Vamos a empezar con el acceso a datos

Juan: Estupendo, con el acceso a datos verás lo más interesante, es lo realmente útil de la programación de servidor, enviar consultas y recibir una página web con los datos devueltos desde la base de datos.

Integración script de servidor con los sistemas gestores de base de datos

El desarrollo de sitios web que utilizan scripts de servidor ha adquirido tanto auge por el uso simultáneo de los scripts y el acceso a datos. Una de las principales ventajas que presenta el trabajar con páginas dinámicas del lado del servidor es el poder trabajar con contenidos que están alojados en bases de datos. Las páginas dinámicas que devuelven consultas a bases de datos las usamos a diario. Ejemplos de páginas con scripts de servidor y acceso a datos:

El buscador al cual indicamos una frase, busca en sus bases de datos y nos devuelve links a diferentes páginas web donde se encuentra información sobre lo solicitado. La página web de nuestro banco donde podemos consultar de la base de datos del banco movimientos de cuentas, recibos, etc.

Aunque desde PHP podemos conectar a multitud de gestores de base de datos sin duda la combinación más usada es PHP como scripts de servidor y MySQL como gestor de base de datos.

Conexión a bases de datos.

En general para acceder a bases de datos se sigue siempre el mismo guión: establecer la conexión con la base de datos, ejecutar las sentencias de consulta o modificación y finalmente cerrar la conexión.

Aunque PHP soporta compatibilidad con accesos a múltiples sistemas de bases de datos, sin embargo, la base de datos más utilizada es MySQL. Hay tres extensiones para conectar a mysql desde PHP: mysql obsoleta, mysqli, o PDO_MySQL. PHP 7 elimina la extensión mysql, dejando solamente las últimas dos opciones.

Elegiremos mysqli (i significa mejorada), evidentemente no vamos a elegir una obsoleta, y aunque PDO_MySQL permite acceder a diferentes gestores de bases de datos y mysqli solo vale para mysql, es más "sencillo" mysqli.

El primer paso para trabajar con bases de datos MySQL es conectar con el servidor, para ello utilizamos la función mysqli para iniciar una conexión indicando usuario, contraseña y servidor al que queremos conectar:

```
<?
$servidor= "localhost";
$usuario= "root";
$clave= "clave";
$conexion= new mysqli($servidor, $usuario, $clave);
if ($conexion->connect_error)
        {   die("Error de conexión: " . $conexion->connect_error);}
echo "Conectado";
?>
```

El símbolo -> es el operador de objeto de PHP, equivale al . en otros lenguajes. En el ejemplo connect_error es una función del objeto conexión que devuelve información sobre el error.

La conexión se cerrará automáticamente cuando termina la secuencia de comandos. Para cerrar la conexión antes se utiliza:

```
<?  $conexion->close(); ?>
```

Creación de bases de datos y tablas

Como veremos a lo largo de esta unidad la mayoría de tareas sobre las bases de datos se hacen con la función **query()** de PHP que, sencillamente ejecuta una orden SQL que realice la tarea, es decir, no es cuestión de saber PHP sino de saber SQL.

Para crear una base de datos añadimos al código del punto anterior la ejecución de la orden SQL: CREATE DATABASE:

```
<? $servidor= "localhost";
$usuario= "root";
$clave= "clave";
$conexion= new mysqli($servidor, $usuario, $clave);
if ($conexion->connect_error)
        {   die("Error de conexión: " . $conexion->connect_error);}
$sql = "CREATE DATABASE mibasededatos";
if ($conexion->query($sql) === TRUE)
         {   echo "Base de datos creada"; }
else
        {   echo "Error creando base de datos: " . $conexion->error;}
$conexion->close();  ?>
```

El comparador === es una comparación estricta. Este comparador indicaría como diferente el número 0 y un texto "0".

Para crear una tabla, al igual que antes, utilizamos la función query() para ejecutar la orden SQL de creación de tablas, OJO, ahora también indicamos la base de datos en la conexión:

```
<? $servidor= "localhost";
$usuario= "root";
$clave= "clave";
$bd= "mibasededatos";
$conexion= new mysqli($servidor, $usuario, $clave, $bd);
if ($conexion->connect_error)
        {   die("Error de conexión: " . $conexion->connect_error);}
$sql = "CREATE TABLE ALUMNOS (
CODIGO INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
NOMBRE VARCHAR(30) NOT NULL,
APELLIDOS VARCHAR(30) NOT NULL,
CORREO VARCHAR(50) )";
if ($conexion->query($sql) === TRUE) {
   echo "Se creó la tabla alumnos";
} else {
   echo "Error al crear tabla";
}
$conexion->close();  ?>
```

Recuperación de la información de la base de datos desde una página Web.

Para recuperar información de una base de datos seguimos la misma técnica, usar la función query() con la orden SQL. La diferencia es que ahora habrá un código PHP adicional para mostrar esta información dentro de la página web. Con la función **fetch_assoc()** se recupera uno a uno los registros devueltos

Recuperar un registro. Veamos primero un caso sencillo, solo recuperamos un registro, supongamos que se recibe de un formulario el código del registro a recuperar:

```
<? $servidor= "localhost";
$usuario= "root";
$clave= "clave";
$bd= "mibasededatos";
$conexion= new mysqli($servidor, $usuario, $clave, $bd);
if ($conexion->connect_error)
        {   die("Error de conexión: " . $conexion->connect_error);}
$sql = "SELECT * FROM ALUMNOS WHERE CODIGO=".$_POST["CODIGO"];
$resultado = $conexion->query($sql);
if ($resultado->num_rows > 0)
{ $registro = $resultado->fetch_assoc();
   echo "Nombre: " . $registro["NOMBRE"]." ". $registro["APELLIDOS"];}
 else
   { echo "No se encontró alumno con ese código";
}
$conexion->close();  ?>
```

Recuperar un conjunto de registros Si la información recuperada es un conjunto de registros habrá que realizar un bucle para mostrarlo.

```
<? $servidor= “localhost”;
$usuario= “root”;
$clave= “clave”;
$bd= “mibasededatos”;
$conexion= new mysqli($servidor, $usuario, $clave, $bd);
if ($conexion->connect_error)
        {    die("Error de conexión: " . $conexion->connect_error);}
$sql = "SELECT * FROM ALUMNOS”;
$resultado = $conexion->query($sql);
if ($resultado->num_rows > 0)
{ while( $registro = $resultado->fetch_assoc() )
{echo "Nombre:" . $registro["NOMBRE"]." ". $registro["APELLIDOS"].”<br>”;}}
 else
 { echo "No hay datos en la tabla de alumnos";}
$conexion->close();  ?>
```

El bucle (while) se repetirá mientras que del conjunto de registros ($resultado) se puedan ir sacando (fech_assoc()) uno a uno los registros que lo componen.

Modificación de la información almacenada: inserciones, actualizaciones y borrados

Debemos de tener en cuenta a la hora de manipular datos en SQL desde PHP que:

- La consulta SQL debe ir entre comillas en PHP
- Los valores que son textos deben llevar comillas en la consulta SQL. Para poner los valores podemos utilizar comillas simples y la consulta SQL ponerla con comillas dobles.
- Los valores numéricos no tienen que llevar comillas
- La palabra NULL no tiene que llevar comillas

Inserción. Para insertar datos utilizaremos la orden SQL INSERT, si al ejecutar la sentencia la función query devuelve el valor TRUE es que el registro se habrá añadido correctamente.

```
<? $servidor= “localhost”; $usuario= “root”; $clave= “clave”; $bd=
“mibasededatos”;
$conexion= new mysqli($servidor, $usuario, $clave, $bd);
if ($conexion->connect_error)
        {    die("Error de conexión: " . $conexion->connect_error);}
$sql = "INSERT INTO ALUMNOS (NOMBRE, APELLIDOS, CORREO)
VALUES ('Luis', 'Suárez Sánchez', 'l.s.s@suarezdefigueroa.es')";
if ($conexion->query($sql) === TRUE)
{    echo "Registro añadido";}
 else
{  echo "Error al añadir registro";}
$conexion->close();  ?>
```

Cuando, como en nuestro caso, la clave es generada automáticamente por SQL podemos recuperar la clave asignada al registro con el código PHP:

```
if ($conexion->query($sql) === TRUE)
{   $codigoasignado = $conexion->insert_id;
    echo "Registro añadido con código: ";
    echo $codigoasignado;}
```

Actualización. Al editar los registros de la tabla nunca se cambia la clave primaría, por tanto, lo normal es solicitar en un formulario el dato de la clave primaria, a partir de esta recuperar de la base de datos el contenido del registro y mostrarlo en un formulario para la edición. Este formulario enviará los cambios a una página que ejecutará la orden UPDATE.

```
<? $servidor= “localhost”; $usuario= “root”; $clave= “clave”; $bd= “mibasededatos”;
$conexion= new mysqli($servidor, $usuario, $clave, $bd);
if ($conexion->connect_error)
      {   die("Error de conexión: " . $conexion->connect_error);}
$sql = "UPDATE ALUMNOS SET NOMBRE='$NOMBRE', APELLIDOS='$APELLIDOS',
CORREO='$CORREO' WHERE CODIGO=$CODIGO";
if ($conexion->query($sql) === TRUE)
{ echo "Registro actualizado";}
else
{ echo "Error al actualizar";}
$conexion->close();  ?>
```

Eliminación. Para eliminar un registro lo normal es solicitar la clave primaria, si no hubiera clave primaria deberemos solicitar todos los campos y buscar un posible registro con todos los campos iguales.

```
<? $servidor= “localhost”; $usuario= “root”; $clave= “clave”; $bd= “mibasededatos”;
$conexion= new mysqli($servidor, $usuario, $clave, $bd);
if ($conexion->connect_error)
      {   die("Error de conexión: " . $conexion->connect_error);}
$sql = "DELETE FROM ALUMNOS WHERE CODIGO=".$_POST["CODIGO"];
if ($conexion->query($sql) === TRUE)
{ echo "Registro borrado";}
else
{ echo "Error al borrar";}
$conexion->close();  ?>
```

Verificación de la información

A la hora de trabajar con datos es muy importante la integridad de los datos, es decir, que sean correctos y lógicos. Por ejemplo, no podemos eliminar un artículo si en alguna tabla se hace menciona este artículo por su referencia, no sabríamos a que corresponde esa referencia.

En las asignaturas de bases de datos tratadas en el ciclo se tratan estos temas, como hacer un diseño de base de datos correcto, la integridad de los datos, los accesos a la base de datos, las transacciones, etc.

El principal responsable de esta integridad es el motor de base de datos, este es el servicio principal para almacenar, procesar y proteger los datos. El Motor de base de datos proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones que acceden a estos datos.

MyISAM era el motor de almacenamiento (storage engine) por defecto en MySQL hasta la versión 5.5.5. Está basado en el antiguo motor de almacenamiento ISAM (Indexed Sequential Access Method) desarrollado por IBM.

InnoDB es una tecnología de almacenamiento de datos de código abierto para la base de datos MySQL, incluido como formato de tabla estándar en todas las distribuciones de MySQL a partir de las versiones 4.0. Su característica principal es que soporta transacciones de tipo ACID y bloqueo de registros e integridad referencial. InnoDB ofrece una fiabilidad y consistencia muy superior a MyISAM, la anterior tecnología de tablas de MySQL.

Con InnoDB cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes.

Gestión de errores

Para poder controlar los errores producidos en nuestro acceso a bases de datos, contamos también con funciones de PHP específicas para su tratamiento, algunas de ellas las hemos visto en los códigos de ejemplo anteriores.

Mecanismos de seguridad y control de accesos

La mejor manera que tenemos de controlar el acceso a nuestra base de datos es utilizando usuarios adecuados a la tarea que haya que desempeñar dentro del script PHP.

Por esta razón, los permisos con los que debe contar el usuario de la BD deben ser, además de los adecuados, los más restringidos posibles. De esta manera complicamos el acceso a personas que no están registradas e intentan conseguir entrar a nuestro sitio web restringido por medios no legales.

Como resulta necesario realizar la conexión en cada carga de la página, podemos tener un conjunto de usuarios de la BD con distintos permisos para utilizar en cada momento el más adecuado.

Verificación del funcionamiento y pruebas de rendimiento

Una vez realizada nuestra aplicación debemos asegurarnos de su correcto funcionamiento.

Es conveniente diseñar una batería de pruebas que recojan todos los casos posibles de la ejecución del mismo. Y con todos queremos remarcar tanto cuando introducimos datos correctos y nuestro sitio funciona correctamente como de qué manera se comporta cuando los datos de prueba buscan provocar un error.

Luego es conveniente realizar pruebas de rapidez en la ejecución de las consultas y el tiempo en el que carga y se transmite nuestra página web, intentando probar el acceso simultáneo de varios usuarios e intentando extrapolar los resultados obtenidos.

Librerías y aplicaciones externas integrables en PHP

Como hemos podido ver a lo largo del curso desarrollar en PHP no es una tarea fácil y requiere muchos conocimientos y tiempo.

Para aliviar la carga existen multitud de librerías libres que podemos acoplar a nuestro sitio web dinámico y que evitan tener que desarrollar todas las funcionalidades dentro de un proyecto global.

Generalmente, basta con descargar la librería, descomprimirla y ya está lista para usar.

En concreto son de mucha utilidad:

FPDF es una librería libre para generación de documentos en formato PDF. Lo primero que tenemos que hacer es bajar la librería y descomprimirla en nuestro sitio. Para generar un pdf utilizaremos un código similar a:

```
<?
require('fpdf.php');
$pdf = new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','',12);
$pdf->Cell(40,10,'Texto a  imprimir');
$pdf->Output();
?>
```

Phpmyadmin es una librería que nos permite administrar nuestra base de datos MySQL a través de un entorno web, es muy útil para realizar mantenimiento y tareas ocasionales para las que no merece la pena desarrollar una página web.