Tema 5: Acceso a bases de datos

Introducción

En este capítulo nos sumergiremos en el fascinante mundo de las bases de datos y descubriremos por qué son la piedra angular del desarrollo web moderno.

I. Cimientos del Desarrollo Web:
Imagina construir una casa sin cimientos sólidos. Sería inestable y propensa a derrumbarse. Lo mismo ocurre en el desarrollo web. Las bases de datos son nuestros cimientos digitales, proporcionando estabilidad y estructura a la información que manejan los sitios y aplicaciones.

II. Almacenamiento Organizado:
En el desarrollo web, lidiar con grandes cantidades de información es la norma. Las bases de datos actúan como nuestro almacén digital, organizando datos de manera eficiente para que puedan ser recuperados, actualizados o eliminados de manera rápida y precisa.

III. Dinamismo y Escalabilidad:
Imagina un sitio web estático donde cada página es igual para todos los usuarios. Las bases de datos permiten la creación de sitios web dinámicos, donde el contenido se adapta según las preferencias del usuario. Además, proporcionan escalabilidad, permitiendo que los sitios crezcan sin perder rendimiento.

IV. Interacción con el Usuario:
Las bases de datos no solo almacenan datos, sino que también facilitan la interacción entre los usuarios y la aplicación. Desde registros de usuarios hasta comentarios en un blog, las bases de datos permiten una experiencia interactiva y personalizada para cada usuario.

V. Mantenimiento y Seguridad:
La integridad y seguridad de los datos son esenciales en el desarrollo web. Las bases de datos ofrecen mecanismos para garantizar que la información sea precisa y esté protegida contra accesos no autorizados. Además, facilitan el mantenimiento de la consistencia de los datos a lo largo del tiempo.

VI. Ejemplo Práctico:
Imaginemos un sitio de comercio electrónico. Las bases de datos almacenan información sobre productos, clientes, transacciones y más. Sin ellas, sería imposible gestionar pedidos, realizar seguimientos de inventario o personalizar recomendaciones para los usuarios.

Las bases de datos son el corazón pulsante del desarrollo web. Desde la gestión eficiente de datos hasta la creación de experiencias interactivas, su papel es fundamental. Comprender su importancia no solo enriquece nuestras habilidades como desarrolladores, sino que también abre la puerta a la creación de aplicaciones web robustas y efectivas.

Introducción al MySQL, y su integración con PHP

En este capítulo, descubriremos el papel vital que desempeña MySQL en el almacenamiento y gestión de datos, así como cómo se entrelaza perfectamente con PHP para dar vida a aplicaciones web dinámicas y poderosas.

I. MySQL: La Fuerza Tranquila de las Bases de Datos:
MySQL es más que un simple sistema de gestión de bases de datos; es el fiel custodio de la información en el vasto paisaje digital. Este sistema relacional de código abierto ofrece un rendimiento excepcional, confiabilidad robusta y una escalabilidad que lo ha convertido en la elección predilecta para desarrolladores y empresas de todo el mundo.

II. Estructura Relacional de MySQL:
Imagina MySQL como un gigantesco almacén digital organizado en tablas interconectadas. Cada tabla representa una entidad específica (como usuarios o productos) y las relaciones entre estas tablas crean un tejido coherente para nuestros datos. La estructura relacional de MySQL es la clave para mantener la integridad y eficiencia de la información.

III. PHP y MySQL: Una Pareja Poderosa:
PHP y MySQL se complementan como el dúo dinámico del desarrollo web. PHP, un lenguaje de programación del lado del servidor, se une con MySQL para gestionar la interacción con la base de datos. Mientras PHP maneja la lógica y la presentación, MySQL almacena y recupera los datos necesarios para que la aplicación cobre vida.

IV. Conexión Mágica: PHP y MySQL en Acción:
La conexión entre PHP y MySQL es como una conversación fluida entre dos amigos bien sincronizados. PHP utiliza funciones específicas, como mysqli_connect(), para establecer una conexión con MySQL. Una vez conectados, PHP puede enviar consultas SQL a MySQL y recibir los resultados para su procesamiento.

```
<?php
// Configuración de la conexión a MySQL
$servername = "localhost";
$username = "tu_usuario";
$password = "tu_contraseña";
$database = "tu_base_de_datos";
// Crear conexión
$conn = new mysqli($servername, $username, $password, $database);
// Verificar la conexión
if ($conn->connect_error) {
        die("Conexión fallida: " . $conn->connect_error);
 }
echo "Conexión exitosa";
?>
```

V. Consultas Dinámicas:
Con la conexión establecida, PHP puede enviar consultas SQL a MySQL para realizar operaciones como la inserción, actualización, selección y eliminación de datos. La armonía entre PHP y MySQL permite una manipulación ágil de la información según las necesidades de la aplicación.

La combinación de MySQL y PHP no solo es una alianza técnica, sino una puerta de entrada a la creación de aplicaciones web robustas y dinámicas. Al entender la profundidad de esta relación, los desarrolladores pueden aprovechar al máximo estas poderosas herramientas y llevar sus proyectos al siguiente nivel.

Configuración del Entorno para Desarrollo Web con PHP y MySQL

A continuación, veremos de los pasos esenciales para preparar tu espacio de trabajo.

I. Instalación de XAMPP: Tu Centro de Control Local:
XAMPP es un paquete que incluye Apache, MySQL, PHP y Perl, proporcionando un entorno de desarrollo completo. Sigue estos pasos:

- Descarga XAMPP desde https://www.apachefriends.org/index.html.
- Instala XAMPP según las instrucciones específicas de tu sistema operativo (Windows, macOS, o Linux).

- Inicia XAMPP y asegúrate de que los servicios de Apache y MySQL estén activos.

II. Verificación de la Instalación:
Para asegurarte de que todo funciona correctamente, abre tu navegador web y visita http://localhost/. Deberías ver la página de inicio de XAMPP. Además, verifica la página de administración de phpMyAdmin (http://localhost/phpmyadmin/) para asegurarte de que MySQL esté funcionando.

III. Configuración de php.ini:
El archivo php.ini contiene configuraciones importantes de PHP. Puedes ajustar estas configuraciones según tus necesidades. Abre el archivo php.ini (ubicado en la carpeta php dentro de tu directorio XAMPP) y haz ajustes según sea necesario. Por ejemplo, puedes cambiar el límite de memoria asignada a PHP o habilitar extensiones específicas.

IV. Creación de un Proyecto:
Crea un nuevo directorio en la carpeta htdocs dentro del directorio de instalación de XAMPP. Este será el lugar donde guardarás tus archivos de proyecto. Puedes acceder a tu proyecto desde el navegador usando http://localhost/tu_proyecto/.

V. Conexión a MySQL desde PHP:
Para asegurarte de que PHP puede conectarse a MySQL, crea un archivo PHP en tu proyecto con el siguiente código de prueba:

```
<?php
$servername = "localhost";
$username = "tu_usuario";
$password = "tu_contraseña";
// Crear conexión
$conn = new mysqli($servername, $username, $password);
// Verificar la conexión
if ($conn->connect_error) {
        die("Conexión fallida: " . $conn->connect_error);
}
echo "Conexión exitosa a MySQL";
?>
```

Abre este archivo desde tu navegador (http://localhost/tu_proyecto/tu_archivo.php) para verificar la conexión exitosa a MySQL.

Tu entorno de desarrollo está configurado y listo para la acción. Ahora puedes empezar a construir aplicaciones web dinámicas utilizando PHP y MySQL.

Crear un a base de datos simple en MySQL

En este paso, seguiremos el proceso de creación de una base de datos simple en MySQL. Para este ejemplo, crearemos una base de datos llamada "tienda" que contendrá información sobre productos y categorías. Utilizaremos el cliente MySQL de línea de comandos, pero también puedes realizar estas operaciones a través de phpMyAdmin u otra interfaz gráfica si lo prefieres.

Paso 1: Accede a MySQL desde localhost/phpmyadmin

Paso 2: Creación de la Base de Datos:
Una vez dentro, crea la base de datos con el siguiente comando:

```
CREATE DATABASE tienda;
```

Paso 3: Selección de la Base de Datos:
Selecciona la base de datos que acabas de crear para comenzar a trabajar con ella:

```
USE tienda;
```

Paso 4: Creación de Tablas:
Ahora crearemos dos tablas simples: una para productos y otra para categorías.

```
-- Tabla de Categorías
CREATE TABLE categorias (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nombre VARCHAR(50) NOT NULL );

-- Tabla de Productos
CREATE TABLE productos (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nombre VARCHAR(100) NOT NULL,
        precio DECIMAL(10, 2) NOT NULL,
        categoria_id INT,
        FOREIGN KEY (categoria_id) REFERENCES categorias(id)
);
```

En este ejemplo, la tabla de productos tiene una columna llamada categoria_id que establece una relación con la tabla de categorías a través de una clave foránea (FOREIGN KEY). Esto permite relacionar cada producto con una categoría específica.

Tu base de datos "tienda" con las tablas de productos y categorías está lista para ser utilizada. Puedes comenzar a insertar datos y realizar consultas.

Este es un ejemplo básico para empezar. En un entorno de producción, es importante considerar la seguridad, la normalización de datos y otros aspectos avanzados del diseño de bases de datos.

Conexión a la Base de Datos

La conexión a la base de datos desde PHP es un paso fundamental para interactuar con MySQL. Aquí te dejo un ejemplo básico de cómo realizar la conexión utilizando el código PHP:

```
<?php
// Configuración de la conexión a MySQL
$servername = "localhost";
// El servidor de la base de datos (puede ser "localhost" o una dirección IP)
$username = "tu_usuario";
// Tu nombre de usuario de MySQL
$password = "tu_contraseña";
// Tu contraseña de MySQL
$database = "tu_base_de_datos";
// El nombre de tu base de datos
// Crear conexión
$conn = new mysqli($servername, $username, $password, $database);
// Verificar la conexión
if ($conn->connect_error) {
        die("Conexión fallida: " . $conn->connect_error);
}
 // Si llegas a este punto, la conexión fue exitosa
echo "Conexión exitosa a la base de datos";
?>
```

Recuerda reemplazar "tu_usuario", "tu_contraseña", y "tu_base_de_datos" con tus propias credenciales y el nombre de tu base de datos.

Este código utiliza la clase mysqli de PHP para establecer una conexión a MySQL. La función die se utiliza para terminar el script si la conexión no es exitosa, mostrando un mensaje de error.

Si la conexión es exitosa, el script imprimirá "Conexión exitosa a la base de datos". A partir de este punto, puedes realizar operaciones como consultas, inserciones o actualizaciones en tu base de datos. No olvides cerrar la conexión cuando hayas terminado usando:

```
$conn->close();
```

Este ejemplo es bastante básico, y en un entorno de producción, es recomendable manejar las conexiones de manera más segura, como mediante el uso de PDO y el manejo adecuado de errores.

Inserción de datos

Veamos un ejemplo básico de cómo puedes realizar la inserción de datos en una tabla de MySQL desde PHP:

```
<?php
// Configuración de la conexión a MySQL
$servername = "localhost";
$username = "tu_usuario";
$password = "tu_contraseña";
$database = "tu_base_de_datos";
// Crear conexión
$conn = new mysqli($servername, $username, $password, $database);
// Verificar la conexión
if ($conn->connect_error) {
        die("Conexión fallida: " . $conn->connect_error);
 }
 // Datos a insertar
$nombre_producto = "Ejemplo Producto";
$precio = 19.99;
$categoria_id = 1;
// Ajusta esto según tu estructura de categorías
// Consulta SQL para la inserción
$sql = "INSERT INTO productos (nombre, precio, categoria_id) VALUES ('$nombre_producto',
$precio, $categoria_id)";
// Ejecutar la consulta
if ($conn->query($sql) === TRUE) {
        echo "Datos insertados correctamente";
} else {
        echo "Error al insertar datos: " . $conn->error;
 }
// Cerrar la conexión
$conn->close();
?>
```

Asegúrate de ajustar "tu_usuario", "tu_contraseña", y "tu_base_de_datos" con tus propias credenciales, y verifica la estructura de tu base de datos y tablas.

Este script PHP realiza lo siguiente:

- Establece una conexión con MySQL.
- Define los datos que deseas insertar en la tabla de productos.
- Crea una consulta SQL para la inserción utilizando la sentencia INSERT INTO.
- Ejecuta la consulta y verifica si fue exitosa.
- Imprime un mensaje indicando si la inserción fue exitosa o muestra un mensaje de error en caso contrario.

- Cierra la conexión a la base de datos.

Este es un ejemplo simple, pero en un entorno de producción, es crucial validar y limpiar los datos antes de insertarlos para prevenir ataques de inyección SQL. Considera también el uso de sentencias preparadas para mejorar la seguridad de tu aplicación.

Consulta de datos

Veamos un ejemplo básico de cómo realizar una consulta de datos desde una base de datos MySQL utilizando PHP:

```
<?php
// Configuración de la conexión a MySQL
$servername = "localhost";
$username = "tu_usuario";
$password = "tu_contraseña";
$database = "tu_base_de_datos";
// Crear conexión
$conn = new mysqli($servername, $username, $password, $database);
// Verificar la conexión
if ($conn->connect_error) {
        die("Conexión fallida: " . $conn->connect_error);
}
 // Consulta SQL para seleccionar todos los productos
$sql = "SELECT id, nombre, precio FROM productos";
// Ejecutar la consulta
$result = $conn->query($sql);
// Verificar si hay resultados
if ($result->num_rows > 0) {
         // Mostrar los datos
        while($row = $result->fetch_assoc()) {
                echo "ID: " . $row["id"]. " - Nombre: " . $row["nombre"]. " - Precio: $" . $row["precio"].
"<br>";
        }
} else {
        echo "No se encontraron resultados";
}
// Cerrar la conexión
$conn->close();
?>
```

Asegúrate de ajustar "tu_usuario", "tu_contraseña", y "tu_base_de_datos" con tus propias credenciales, y verifica la estructura de tu base de datos y tablas.

Este script PHP realiza lo siguiente:

- Establece una conexión con MySQL.
- Realiza una consulta SQL para seleccionar todos los productos de la tabla.
- Ejecuta la consulta y almacena los resultados en la variable $result.
- Verifica si hay resultados y, en caso afirmativo, imprime los datos usando un bucle while.
- Si no hay resultados, imprime un mensaje indicando que no se encontraron resultados.
- Cierra la conexión a la base de datos.

Otras funciones PHP útiles para trabajar con bases de datos

Existen varias funciones en PHP que son útiles al trabajar con bases de datos. Aquí hay algunas de las más comunes:

mysqli_query($conn, $sql): Ejecuta una consulta en la base de datos especificada por la conexión $conn. Devuelve un objeto que puede ser utilizado para recuperar los resultados de la consulta.

```
$result = mysqli_query($conn, "SELECT * FROM tabla");
```

mysqli_fetch_assoc($result): Recupera una fila de resultados como un array asociativo. Puedes usar esto en un bucle para iterar sobre todos los resultados.

```
while ($row = mysqli_fetch_assoc($result)) {
        echo $row['columna'];
        }
```

mysqli_num_rows($result): Devuelve el número de filas en un conjunto de resultados.

```
$num_rows = mysqli_num_rows($result);
echo "Número de filas: " . $num_rows;
```

mysqli_insert_id($conn): Devuelve el ID generado por una consulta INSERT anterior.

```
$id_insertado = mysqli_insert_id($conn);
```

mysqli_error($conn): Devuelve una cadena de descripción del último error ocurrido durante la ejecución de la última consulta.

```
if (!$result) {
        echo "Error: " . mysqli_error($conn);
}
```

mysqli_real_escape_string($conn, $string): Escapa caracteres especiales en una cadena para usarla en una consulta SQL. Esto ayuda a prevenir inyecciones SQL.

```
$nombre = mysqli_real_escape_string($conn, $_POST['nombre']);
```

mysqli_close($conn): Cierra la conexión a la base de datos.

```
mysqli_close($conn);
```

Estas son solo algunas funciones básicas. Hay otras funciones y conceptos más avanzados, como el uso de sentencias preparadas y el manejo de transacciones, que son esenciales para escribir código seguro y eficiente al trabajar con bases de datos. Además, considera el uso de la extensión PDO (PHP Data Objects) que proporciona una interfaz más flexible y segura para trabajar con bases de datos en PHP.

La libreria PDO

PDO (PHP Data Objects) es una extensión en PHP que proporciona una interfaz consistente y segura para acceder a bases de datos relacionales. A diferencia de la extensión mysqli que está específicamente diseñada para MySQL, PDO es una capa de abstracción que funciona con una variedad de sistemas de gestión de bases de datos (MySQL, PostgreSQL, SQLite, SQL Server, etc.).
Aquí hay algunos puntos clave sobre PDO:

Conexión a Bases de Datos:
La conexión a la base de datos se realiza mediante el uso de la clase PDO y su constructor.
PDO admite múltiples sistemas de gestión de bases de datos, lo que hace que tu código sea más portátil si alguna vez decides cambiar de sistema.

```
$dsn = 'mysql:host=localhost;dbname=tu_base_de_datos';
$usuario = 'tu_usuario';
$contraseña = 'tu_contraseña';
try {
	$pdo = new PDO($dsn, $usuario, $contraseña);
}
catch (PDOException $e) {
	echo 'Conexión fallida: ' . $e->getMessage();
}
```

Consultas Preparadas:
Las consultas preparadas son una característica clave de seguridad al trabajar con bases de datos. Ayudan a prevenir ataques de inyección SQL.

```
$nombre = 'Juan';
$stmt = $pdo->prepare('SELECT * FROM usuarios WHERE nombre = :nombre');
$stmt->bindParam(':nombre', $nombre, PDO::PARAM_STR);
$stmt->execute();
```

Manejo de Errores:
PDO lanza excepciones por defecto en caso de errores, lo que facilita el manejo de errores de manera más eficiente.

```
try {
	// Tu código con consultas PDO
}
catch (PDOException $e) {
	echo 'Error: ' . $e->getMessage();
 }
```

Obtención de Resultados:
PDO ofrece varios métodos para obtener resultados, incluyendo fetch, fetchAll, y fetchColumn.

```
$stmt = $pdo->query('SELECT * FROM usuarios');
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
	echo $row['nombre'];
}
```

Transacciones:
PDO admite transacciones, lo que te permite agrupar varias consultas en una única transacción atómica.

```
try {
	$pdo->beginTransaction();
	// Tus consultas
	$pdo->commit();
}
catch (PDOException $e) {
	$pdo->rollBack();
	echo 'Error en la transacción: ' . $e->getMessage();
}
```

Diferencias con mysqli:
Mientras mysqli es específico para MySQL, PDO es una interfaz más general que funciona con múltiples sistemas de gestión de bases de datos.

PDO utiliza una sintaxis de marcador de posición consistente (:nombre) en lugar de los marcadores de posición basados en posición (?) de mysqli.

El uso de PDO aporta versatilidad, seguridad y portabilidad a tu código, ya que puedes cambiar de base de datos con relativa facilidad si es necesario. Además, la interfaz orientada a objetos de PDO facilita la escritura de código limpio y organizado.