Clase 10

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 10

¡Te damos la bienvenida a la materia **Programación Web II**!

En esta clase vamos a ver los siguientes temas:

- Conexión a una base de datos mediante el módulo mysqli
- Métodos más usados
- Ejemplos de uso de los métodos citados.

Módulo MySQLi (Parte 1)

Conexión a una base de datos mediante el módulo MySQLi

El módulo MySQLi (MySQL improved) en PHP es una extensión que proporciona una interfaz mejorada y orientada a objetos para interactuar con una base de datos MySQL.

Fue introducido a partir de PHP 5 y se ha convertido en la opción preferida para trabajar con MySQL en PHP debido a su mayor flexibilidad y seguridad en comparación con la extensión MySQL original.

Antes de la introducción de mysqli, el módulo utilizado para interactuar con bases de datos MySQL en PHP era mysql.

Sin embargo, mysql fue declarado obsoleto en PHP 5.5 y eliminado completamente en PHP 7.0 debido a varias limitaciones y vulnerabilidades de seguridad.

El módulo MySQLi ofrece una serie de beneficios y características avanzadas, incluyendo:

- **Orientación a objetos**: MySQLi permite trabajar con una sintaxis orientada a objetos, lo que facilita la manipulación de la base de datos y el manejo de consultas y resultados.

- **Consultas preparadas (prepared statements):** MySQLi admite la ejecución de consultas preparadas, lo que mejora la seguridad al prevenir ataques de inyección de SQL. Las consultas preparadas separan los datos de la consulta en sí, evitando la concatenación directa de valores en la consulta y permitiendo que los valores se vinculen de forma segura.

- **Soporte para transacciones:** MySQLi permite realizar operaciones en modo transaccional, lo que garantiza que un conjunto de instrucciones SQL se ejecuten de manera completa y coherente, evitando resultados parciales en caso de fallos.

- **Manejo mejorado de errores:** MySQLi proporciona una gestión mejorada de errores y excepciones, lo que facilita la detección y el manejo de problemas relacionados con la base de datos.

- **Soporte para características avanzadas de MySQL:** MySQLi permite aprovechar características avanzadas de MySQL, como el manejo de conjuntos de resultados grandes, el uso de múltiples consultas en una sola llamada y la ejecución de consultas en segundo plano.

La utilización del módulo MySQLi en PHP implica los siguientes pasos básicos:

1. **Establecer una conexión**: Antes de interactuar con la base de datos, es necesario establecer una conexión utilizando la clase mysqli y proporcionando los detalles de conexión, como el servidor, el nombre de usuario, la contraseña y la base de datos.

2. **Ejecutar consultas SQL:** Una vez establecida la conexión, se pueden ejecutar consultas SQL utilizando los métodos proporcionados por la clase mysqli. Estas consultas pueden ser consultas directas o consultas preparadas, dependiendo de la necesidad y la seguridad requerida.

3. **Obtener y manipular resultados:** Después de ejecutar una consulta, es posible obtener y manipular los resultados utilizando métodos como fetch_assoc(), fetch_row(), fetch_object(), entre otros, dependiendo del formato deseado para los resultados.

4. **Manejo de errores y liberación de recursos:** Es importante realizar una gestión adecuada de errores y liberar los recursos utilizados, como cerrar la conexión a la base de datos utilizando el método close().

Existen tres variantes de uso de MySQLi:

1. **Modo procedural (Procedural style):** En este enfoque, se utilizan funciones procedurales para establecer la conexión, ejecutar consultas y realizar operaciones en la base de datos. Algunos ejemplos de funciones utilizadas en este modo son mysqli_connect(), mysqli_query(), mysqli_fetch_assoc(), etc.

2. **Modo Orientado a Objetos:** En este enfoque, se utilizan objetos para interactuar con la base de datos. Se crean instancias de la clase mysqli y se llaman a los métodos correspondientes para establecer la conexión, ejecutar consultas y manipular los resultados. Algunos ejemplos de métodos utilizados en este modo son __construct(), query(), fetch_assoc(), etc.

3. **Utilizando la extensión PDO (PHP Data Objects**): PDO es una abstracción de base de datos que permite trabajar con diferentes bases de datos utilizando una interfaz unificada. En otras palabras, con PDO podemos conectarnos a diferentes tipos de base datos utilizando sus respectivos controladores, como ser, para la base de datos MySQL el controlador a usar es PDO_MYSQL. Este enfoque es más flexible ya que nos permite cambiar fácilmente a otro tipo de base de datos sin modificar mucho el código. Algunos ejemplos de métodos utilizados con PDO son prepare(), execute(), fetch(), etc.

A continuación, un ejemplo de conexión a una base de datos MySQL en sus tres variantes.

Ejemplo de mysqli orientado a objetos:

```php
<?php

  $servidor = "localhost";
  $usuario = "root";
  $clave = "root";
  $basededatos = "davinci";

  // Crea la conexión a la base de datos
  $conexion = new mysqli($servidor, $usuario, $clave, $basededatos);

  // Prueba la conexión
  if($conexion->connect_error){
    die("Error al intentar conectar la base de datos: " . $conexion->connect_error);
  }
  else{
    echo "Conexión satisfactoria.";
  }

?>
```

Ejemplo de mysqli estilo procedural:

```php
<?php

  $servidor = "localhost";
  $usuario = "root";
  $clave = "root";
  $basededatos = "davinci";

  // Prueba la conexión
  $conexion = mysqli_connect($servidor, $usuario, $clave, $basededatos);

  // Crea la conexión a la base de datos
  if(!$conexion){
    die("Error al intentar conectar la base de datos: " . mysqli_connect_error());
  }
  else{
    echo "Conexión satisfactoria.";
  }

?>
```

Ejemplo de mysqli estilo orientado a objetos con PDO:

```php
<?php

  $servidor = "localhost";
  $usuario = "root";
  $clave = "root";
  $basededatos = "davinci";

  try{
    $conexion = new PDO("mysql:host=$servidor;dbname=$basededatos",
$usuario, $clave);
    // Establece el modo de error PDO en excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    echo "Conexión satisfactoria.";
  }
  catch(PDOException $e){
    echo "Error al intentar conectar la base de datos: " . $e-
>getMessage();
  }

?>
```

Hemos explorado las diferentes maneras de conectarse a una base de datos MySQL con PHP: el modo procedural, el modo orientado a objetos y utilizando la extensión PDO (PHP Data Objects).

Cada enfoque tiene sus propias características y ventajas, y la elección del método dependerá de las necesidades y preferencias del desarrollador.

- **El modo procedural es el enfoque más antiguo y sencillo**, donde se utilizan funciones específicas para conectarse a la base de datos y realizar consultas. Es adecuado para proyectos pequeños o cuando se necesita una solución rápida y directa. Sin embargo, puede ser menos estructurado y más difícil de mantener en proyectos más grandes y complejos.

- **El modo orientado a objetos es una forma más moderna y estructurada de interactuar con la base de datos**. Se utilizan clases y objetos para representar la conexión y las consultas, lo que facilita la reutilización de código y mantiene una mejor organización. Este enfoque es preferido por muchos desarrolladores debido a su legibilidad y escalabilidad.

- **Utilizando la extensión PDO, se puede lograr una mayor flexibilidad al conectar a diferentes bases de datos**, ya que PDO es compatible con varios sistemas de gestión de bases de datos, no solo MySQL. Además, PDO proporciona una capa de abstracción que puede mejorar la seguridad y facilitar la migración a otra base de datos si fuera necesario.

Recordá que al momento de elegir la forma de conectarse a una base de datos MySQL con PHP es importante considerar el tamaño y complejidad del proyecto, así como la compatibilidad con otros sistemas de bases de datos.

Métodos MySQLi más usados

La siguiente lista comprende algunos de los métodos más utilizados en MySQLi:

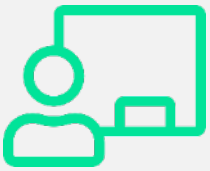
- **mysqli_connect():** Establece una nueva conexión con el servidor de base de datos MySQL.
- **mysqli_query():** Ejecuta una consulta SQL en la base de datos y devuelve un objeto mysqli_result o FALSE en caso de error.
- **mysqli_fetch_assoc():** Obtiene la siguiente fila de un conjunto de resultados como un array asociativo.
- **mysqli_real_escape_string()**: Escapa los caracteres especiales en una cadena para su uso en una consulta SQL.
- **mysqli_prepare():** Prepara una consulta SQL para su ejecución y devuelve un objeto mysqli_stmt o FALSE en caso de error.}
- **mysqli_stmt_bind_param():** Vincula parámetros a una sentencia preparada antes de su ejecución.
- **mysqli_stmt_execute():** Ejecuta una sentencia preparada.
- **mysqli_stmt_get_result():** Obtiene el resultado de una sentencia preparada como un objeto mysqli_result.
- **mysqli_fetch_array():** Obtiene la siguiente fila de un conjunto de resultados como un array numérico, asociativo o ambos.
- **mysqli_num_rows():** Obtiene el número de filas en un conjunto de resultados.

- **mysqli_error():** Devuelve una descripción del último error ocurrido durante la ejecución de una consulta.

- **mysqli_connect_errno():** Devuelve el número de error de la última llamada a mysqli_connect(), permitiendo detectar errores de conexión de forma programática. Si la conexión a la base de datos falla, el valor devuelto será un número de error distinto de cero.

- **mysqli_connect_error():** Devuelve la descripción del error de la última llamada a mysqli_connect(), útil para mostrar mensajes de error más informativos o registrar detalles del error.

- **mysqli_close():** Cierra la conexión al servidor de base de datos.



Estos son solo algunos ejemplos de los métodos más comúnmente utilizados en **MySQLi.**
pero hay muchos más métodos disponibles en la **documentación oficial de PHP**.

Ejemplos de uso de los métodos MySQLi citados

En nuestros ejemplos usaremos la base de datos MySQL y el modo de conexión procedural.

mysqli_connect()

Ejemplo del método mysqli_connect():

```php
<?php


  $servidor = "localhost";

  $usuario = "root";

  $clave = "root";

  $basededatos = "davinci";

  $conexion = mysqli_connect($servidor, $usuario, $clave,
$basededatos);


?>
```

Este método establece una nueva conexión con el servidor de base de datos MySQL utilizando los parámetros especificados: **$servidor** (nombre del servidor), **$usuario** (nombre de usuario), **$clave** (contraseña) y **$basededatos** (nombre de la base de datos). Retorna un objeto de conexión de MySQLi o FALSE si la conexión falla.

mysqli_query()

Ejemplo del método mysqli_query():

```php
<?php


  $resultados = mysqli_query($conexion, $consulta_sql);


?>
```

Este método ejecuta una consulta SQL (**$sql**) en la base de datos utilizando la conexión establecida (**$conexion**).
Retorna un objeto mysqli_result que contiene el resultado de la consulta o FALSE en caso de error.

mysqli_fetch_assoc()

Ejemplo del método mysqli_fetch_assoc():

```php
<?php

  while($registro = mysqli_fetch_assoc($resultados)){
    echo '<br /><br />';
    echo '<b>Nombre: </b>' . $registro["nombre"] . '<br />';
    echo '<b>Apellido: </b>' . $registro["apellido"] . '<br />';
    echo '<b>Usuario: </b>' . $registro["usuario"] . '<br />';
  }

?>
```

Este método obtiene la siguiente fila del conjunto de resultados (**$result**) como un array asociativo, donde las claves del array corresponden a los nombres de las columnas de la tabla.
En este ejemplo, se muestra el ID y el nombre de usuario de cada fila.

mysqli_real_escape_string()

Ejemplo del método mysqli_real_escape_string():

```php
<?php

  $usuario = mysqli_real_escape_string($conexion, $resultados);

?>
```

Este método escapa los caracteres especiales en una cadena (**$usuario**) para su uso seguro en una consulta SQL.
Evita la inyección de SQL al asegurarse de que los datos se traten como literales y no como parte de la sintaxis SQL.

mysqli_prepare()

Ejemplo del método mysqli_prepare():

```php
<?php

  $stmt = mysqli_prepare($conexion, $consulta_sql);

?>
```

Este método prepara una consulta SQL (**$consulta_sql**) para su ejecución utilizando la conexión establecida (**$conexion**).
Retorna un objeto mysqli_stmt que representa la sentencia preparada.

mysqli_stmt_bind_param()

Ejemplo del método mysqli_stmt_bind_param():

```php
<?php

  mysqli_stmt_bind_param($stmt, "s", $usuario);

?>
```

Este método vincula parámetros a una sentencia preparada (**$stmt**) antes de su ejecución.
En este ejemplo, se vincula un parámetro de tipo cadena ("s") con el valor de la variable **$username**.

mysqli_stmt_excute()

Ejemplo del método mysqli_stmt_excute():

```php
<?php

  mysqli_stmt_execute($stmt);

?>
```

Este método ejecuta una sentencia preparada (**$stmt**) que ha sido previamente preparada y vinculada con parámetros. Ejecuta la consulta SQL en la base de datos.

mysqli_stmt_get_result()

Ejemplo del método mysqli_stmt_get_result():

```php
<?php

  mysqli_stmt_get_result($stmt);

?>
```

Este método obtiene el resultado de una sentencia preparada (**$stmt**) como un objeto mysqli_result. Permite acceder a los datos resultantes de la consulta preparada.

mysqli_fetch_array()

Ejemplo del método mysqli_fetch_array():

```php
<?php

  while($registro = mysqli_fetch_array($resultados, MYSQLI_ASSOC)){
    echo '<br /><br />';
    echo '<b>Nombre: </b>' . $registro["nombre"] . '<br />';
    echo '<b>Apellido: </b>' . $registro["apellido"] . '<br />';
    echo '<b>Usuario: </b>' . $registro["usuario"] . '<br />';
  }

?>
```

Este método obtiene la siguiente fila del conjunto de resultados (**$resultados**) como un array numérico, asociativo o ambos.

En este ejemplo, se utiliza el modo MYSQLI_ASSOC para obtener un array asociativo donde las claves son los nombres de las columnas, también se puede optar por el modo MYSQLI_NUM (array numérico) o bien por el modo MYSQLI_BOTH. Nota: Si no se especifica nada el valor por defecto es MYSQLI_BOTH.

```php
<?php

  while($registro = mysqli_fetch_array($resultados, MYSQLI_NUM)){
    echo '<br /><br />';
    echo '<b>Nombre: </b>' . $registro[0] . '<br />';
    echo '<b>Apellido: </b>' . $registro[1] . '<br />';
    echo '<b>Usuario: </b>' . $registro[2] . '<br />';
  }

?>
```

mysqli_num_rows()

Ejemplo del método mysqli_num_rows():

```php
<?php

  $registros_encontrados = mysqli_num_rows($resultados);
  echo $registros_encontrados;

?>
```

Este método devuelve el número de filas en un conjunto de resultados (**$resultados**).

En este ejemplo, se almacena el número de filas en la variable $registros_encontrados y se muestra en pantalla.

mysqli_error()

Ejemplo del método mysqli_error():

```php
<?php

  if(mysqli_error($conexion)){
        echo mysqli_error($conexion);
  }

?>
```

Este método devuelve una descripción del último error ocurrido durante la ejecución de una consulta en la conexión (**$conexion**).

En este ejemplo, se verifica si hay algún error y se muestra en pantalla en caso de que exista.

mysqli_connect_errno()

Ejemplo del método mysqli_connect_errno():

```php
<?php

  if(!$conexion){
    die("Error al intentar conectar la base de datos: " .
mysqli_connect_errno());
  }
  else{
    echo "Conexión satisfactoria.";
  }

?>
```

Este método se utiliza para obtener el número de error de la última llamada a **mysqli_connect()**. Si la conexión a la base de datos falla, **mysqli_connect_errno()** devuelve el número de error correspondiente.
Si no hay ningún error, devuelve cero.
Es útil para detectar y manejar errores de conexión de manera programática. Por ejemplo, si el valor devuelto por **mysqli_connect_errno()** es diferente de cero, podemos mostrar un mensaje de error específico o realizar alguna acción de manejo de errores.

mysqli_connect_error()

Ejemplo del método mysqli_connect_error():

```php
<?php

  if(!$conexion){
    die("Error al intentar conectar la base de datos: " .
mysqli_connect_error());
  }
    echo "Conexión satisfactoria.";

?>
```

Este método se utiliza para obtener la descripción del error de la última llamada a **mysqli_connect()**.
Devuelve una cadena que describe el error de conexión.

Es útil para mostrar un mensaje de error significativo al usuario o para registrar detalles del error en un archivo de registro.
Por ejemplo, se puede utilizar **mysqli_connect_error()** para mostrar un mensaje personalizado de error en caso de que la conexión a la base de datos falle.

mysqli_close()

Ejemplo del método mysqli_close():

```php
<?php

  // Conexión a la base de datos
  $servidor = "localhost";
  $usuario = "root";

  $clave = "root";

  $basededatos = "davinci";

  $conexion = mysqli_connect($servidor, $usuario, $clave,
$basededatos);


  // Verificar la conexión
  if ($mysqli->connect_error) {
      die("Error de conexión: " . $mysqli->connect_error);
  }

  // Realizar consultas y otras operaciones con la base de datos...

  // Cerrar la conexión
  mysqli_close($mysqli);

?>
```

El método **mysqli_close()** se utiliza para realizar precisamente esto: cerrar la conexión abierta a la base de datos.
Al invocar este método, PHP liberará los recursos asociados con la conexión y la cerrará.
Es especialmente útil en escenarios donde se establecen múltiples conexiones a lo largo del script y deseas asegurarte de que todas se cierren adecuadamente.

Es importante destacar que, en PHP, las conexiones a la base de datos se cierran automáticamente cuando el script finaliza su ejecución.
Sin embargo, es una buena práctica cerrarlas explícitamente tan pronto como termines de usarlas, en lugar de depender únicamente de la clausura automática, para asegurar que la conexión se libere oportunamente.

Hemos llegado así al final de esta clase en la que vimos:

- Conexión a una base de datos mediante el módulo mysqli.
- Métodos más usados.
- Ejemplos de uso de los métodos citados.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP

W3Schools: PHP Tutorial

Cabezas Granado, Luis. González Lozano, F. (2018) Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O. (2016) Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, Luke Thompson, L. (2017) Desarrollo Web con PHP y MySQL. Quinta Edición (2017, Editorial Anaya).

Para ampliar la información

PHP: MySLQi

PHP: The Right Way

Todo sobre PHP

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero

MDN: Introducción al lado Servidor

Guía de HTML