UT 5: ACCESO A BASE DE DATOS

1. **Introducción**
2. Conectar a un servidor MySQL desde PHP
3. Controlar la conexión
4. Ejecutar consultas SQL
    1. Obtener datos
    2. Insertar datos
    3. Actualizar y borrar datos
5. Problemas de seguridad. Inyección de código

1. INTRODUCCIÓN

Una **página web dinámica** es aquella que, mediante tecnologías del lado del servidor, puede acceder y recuperar datos de una **base de datos** en tiempo real. Esta capacidad permite la presentación de contenido personalizado, interactividad y adaptabilidad según la información almacenada, ofreciendo experiencias más ricas y actualizadas para los usuarios.

Con la solución integrada de XAMPP, tenemos entre otras cosas el intérprete de PHP y MySQL instalado.

Vamos a ver los diferentes pasos necesarios para acceder a la base de datos MySQL desde PHP.

Al gestor de base de datos podemos acceder con http://localhost/phpmyadmin/

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. **Conectar a un servidor MySQL desde PHP**
3. Controlar la conexión
4. Ejecutar consultas SQL
    1. Obtener datos
    2. Insertar datos
    3. Actualizar y borrar datos
5. Problemas de seguridad. Inyección de código

2. CONECTAR A UN SERVIDOR MYSQL DESDE PHP

Para poder acceder a MySQL se puede utilizar la siguiente API: MySQLi.

Las ventajas de esta extensión son:

- Interfaz orientada a objetos
- Soporte de transacciones y diferentes declaraciones.
- Soporte de depuración mejorado.
- Funcionalidades más potentes.

Sintaxis para crear una nueva conexión

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "test";


// Crear conexión
$conn = new mysqli($servername, $username,
$password, $database);


// Verificar la conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn-
>connect_error);
} else{
    echo "Conexión establecidad <br>";
}
    echo $conn->host_info. "<br>";
?>
```

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. Conectar a un servidor MySQL desde PHP
3. **Controlar la conexión**
4. Ejecutar consultas SQL
    1. Obtener datos
    2. Insertar datos
    3. Actualizar y borrar datos
5. Problemas de seguridad. Inyección de código

3. CONTROLAR LA CONEXIÓN

Es necesario garantizar tanto la confiabilidad de la conexión como el uso adecuado de recursos. Para ello, hay que controlar los posibles errores que pudiera suceder y cerrar adecuadamente la conexión con la base de datos.

CERRAR BASE DE DATOS

- `close()`: Cierra la conexión

```
$conexión->close();
```

3. CONTROLAR LA CONEXIÓN

COMPROBACIÓN DE ERRORES

- `connect_error`: método booleano que determina si ha habido un error o no.
- `connect_errno`:devuelve el número del error que ha sucedido.
- `error_list`: devuelve la lista de todos los errores ocurridos durante la conexión

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. Conectar a un servidor MySQL desde PHP
3. Controlar la conexión
4. **Ejecutar consultas SQL**
    1. Obtener datos
    2. Insertar datos
    3. Actualizar y borrar datos
5. Problemas de seguridad. Inyección de código

4. EJECUTAR CONSULTAS

Para ejecutar consultas en PHP a MySQL se utiliza el método query().

A continuación, vamos a ver las diferentes opciones:

- Obtener datos
- Insertar datos
- Modificar datos
- Eliminar datos

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. Conectar a un servidor MySQL desde PHP
3. Controlar la conexión
4. **Ejecutar consultas SQL**
    1. **Obtener datos**
    2. Insertar datos
    3. Actualizar y borrar datos
5. Problemas de seguridad. Inyección de código

4.1. OBTENER DATOS (SELECT)

Se utilizan los métodos:

- query() para realizar la consulta
- fetch_assoc() para procesar el resultado.

Obtener datos

```php
<?php
//Conexion con la base de datos

// Ejecutar una consulta SELECT
$sql = "SELECT id, nombre FROM usuarios";
$result = $conn->query($sql);

//Verificar si la consulta fue exitosa
if ($result) {
    // Obtener los datos
    while ($row = $result->fetch_assoc()) {
        // Procesar cada fila de datos
        echo "ID: " . $row["id"] . ", Nombre: " . $row["nombre"]  . "<br>";
    }
    // Liberar el conjunto de resultados
    $result->free();
} else {
    // Manejar el error en caso de una consulta fallida
    echo "Error en la consulta: " . $conn->error;
}
$conn->close();
?>
```

4.1. OBTENER DATOS (SELECT)

Otros métodos y propiedades útiles:

- `num_rows`: devuelve el número de filas que devuelve la consulta.

```
$sql = "SELECT id, nombre, correo FROM usuarios";
$result = $conn->query($sql);
if ($result) {
     $num_rows = $result->num_rows;
}
```

4.1. OBTENER DATOS (SELECT)

- `data_seek`: función que permite colocarse en un determinado número de fila del conjunto de resultados.

```
while ($row = $result->fetch_assoc()) {
    echo "ID: " . $row["id"] . ", Nombre: " . $row["nombre"]. "<br>";
}
// Mover el puntero interno al principio del conjunto de resultados
$result->data_seek(0);
// Recorrer el conjunto de resultados desde el principio
while ($row = $result->fetch_assoc()) {
    echo "ID: " . $row["id"] . ", Nombre: " . $row["nombre"] . "<br>";
}
```

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. Conectar a un servidor MySQL desde PHP
3. Controlar la conexión
4. **Ejecutar consultas SQL**
    1. Obtener datos
    2. **Insertar datos**
    3. Actualizar y borrar datos
5. Problemas de seguridad. Inyección de código

4.2. INSERTAR DATOS (INSERT)

Se utiliza:

- el método `query()`
- Y opcionalmente, la propiedad `insert_id` si la clave primaria es autoincremental.

Insertar datos

```php
<?php
//conexión con la base de datos

$nombre = "Juan";

$sql = "INSERT INTO usuarios (nombre) VALUES ('$nombre')";
$result = $conn->query($sql);

// Verificar si la consulta fue exitosa
if ($result) {
    echo "Datos insertados correctamente.<br>";

    // Obtener el ID del último registro insertado (si la tabla tiene una columna autoincremental)
    $last_inserted_id = $conn->insert_id;
    echo "ID del último registro insertado: " . $last_inserted_id;
} else {
        echo "Error en la consulta: " . $conn->error;
}

// Cerrar la conexión
$conn->close();
?>
```

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. Conectar a un servidor MySQL desde PHP
3. Controlar la conexión
4. **Ejecutar consultas SQL**
    1. Obtener datos
    2. Insertar datos
    3. **Actualizar y borrar datos**
5. Problemas de seguridad. Inyección de código

4.3. ACTUALIZAR Y BORRAR DATOS

Se utiliza:

- el método `query()`
- Y opcionalmente, la propiedad **`affected_rows`** para evaluar las filas que se han afectado

Actualizar datos

```
<?php
//conexión a la base de datos

$idUsuario = 1;
$nuevoNombre = "Juan Antonio";
$sql = "UPDATE usuarios SET nombre= '$nuevoNombre' WHERE id
= $idUsuario";
$result = $conn->query($sql);

if ($result) {
    $numRowsAffected = $conn->affected_rows;

    if ($numRowsAffected > 0) {
        echo "Actualización exitosa. Número de filas afectadas:
$numRowsAffected.<br>";
    } else {
        echo "No se realizaron cambios. El usuario con ID $idUsuario no
fue encontrado.<br>";
    }
} else {
      echo "Error en la consulta: " . $conn->error;
}
?>
```

Eliminar datos

```php
<?php
//conexión a la base de datos

$idUsuario = 1;
$sql = "DELETE FROM usuarios WHERE id = $idUsuario";
$result = $conn->query($sql);

if ($result) {
    $numRowsAffected = $conn->affected_rows;

    if ($numRowsAffected > 0) {
        echo "Eliminación exitosa. Número de filas afectadas: $numRowsAffected.<br>";
    } else {
        echo "No se realizaron cambios. El usuario con ID $idUsuario no fue encontrado.<br>";
    }
} else {
    // Manejar el error en caso de una consulta fallida
    echo "Error en la consulta: " . $conn->error;
}
?>
```

UT 5: ACCESO A BASE DE DATOS

1. Introducción
2. Conectar a un servidor MySQL desde PHP
3. Controlar la conexión
4. Ejecutar consultas SQL
    1. Obtener datos
    2. Insertar datos
    3. Actualizar y borrar datos
5. **Problemas de seguridad. Inyección de código**

5. PROBLEMAS DE SEGURIDAD. INYECCIONES DE SQL

Muchas veces, los datos que se envían a SQL vienen de formularios que rellenan los usuarios, por lo que se puede producir inyección SQL o SQL injection.

Por ejemplo, tenemos la consulta:

```
$sql = "DELETE FROM alumnos where nombre =' $nombre' ";
```

Esa variable viene de un formulario y el usuario introduce: Juan' or '1'='1

La consulta SQL se convertira en `DELETE FROM alumnos where nombre =' Juan' or '1'='1' "`, lo que provocará que se eliminen todos los registros de la tabla.

5. PROBLEMAS DE SEGURIDAD. INYECCIONES DE SQL

SQL injection es una técnica maliciosa donde se insertan instrucciones SQL no deseadas en las consultas para manipular la base de datos.

5. PROBLEMAS DE SEGURIDAD. INYECCIONES DE SQL

Medidas para evitarlo:

- Utilizar la función real_escape_string que recibe un texto y lo devuelve en un formato seguro. Lo que hace esta función es escapar los caracteres peligrosos como comillas, saltos de línea...

```
$nombre = "Juan";
$nombreEscapado = $conn->real_escape_string($nombre);
$sql = "SELECT * FROM usuarios WHERE nombre = '$nombreEscapado'";
$resultado = $conn->query($sql);
```

- No utilizar la función multi_query, ya que permite ejecutar varias intrucciones a la vez.

5. PROBLEMAS DE SEGURIDAD. INYECCIONES DE SQL

- Utilizar sentencias preparadas y vinculación de parámetros al ejecutar las consultas cuando los valores vienen de un formulario, en vez de la función query. Es la opción más **recomendable**.

```
$stmt = $conn->prepare("SELECT * FROM usuarios WHERE nombre =
  ?");
$stmt->bind_param("s", $nombre);

$nombre = "usuario";
$stmt->execute();
$result = $stmt->get_result();
```

5. PROBLEMAS DE SEGURIDAD. INYECCIONES DE SQL

La función bind_param se utiliza para vincular variables a los marcadores de posición en una sentencia SQL preparada. El primer argumento especifica los tipos de datos de los parámetros que se van a vincular. Estos especificadores de tipo pueden ser:

- "i": Entero
- "d": Doble (número de punto flotante)
- "s": Cadena
- "b": Blob (binario)

Obtener datos

```
//conexión a la base de datos
// Crear una sentencia preparada con marcadores de posición (?)
$stmt = $conn->prepare("SELECT nombre, edad FROM usuarios
WHERE id = ?");
$idUsuario = 1;
//Vincular el parámetro a la sentencia preparada
$stmt->bind_param("i", $idUsuario);
$stmt->execute();

//Vincular los resultados a variables usando bind_result
$stmt->bind_result($nombre, $edad);

// Recuperar resultados con fetch_assoc
$resultado = array();
while ($stmt->fetch()) {
    // Crear un array asociativo con los resultados usando fetch_assoc
    $resultado[] = array("nombre" => $nombre, "edad" => $edad);
}
// Cerrar la sentencia preparada y la conexión
$stmt->close();
$conn->close();
```