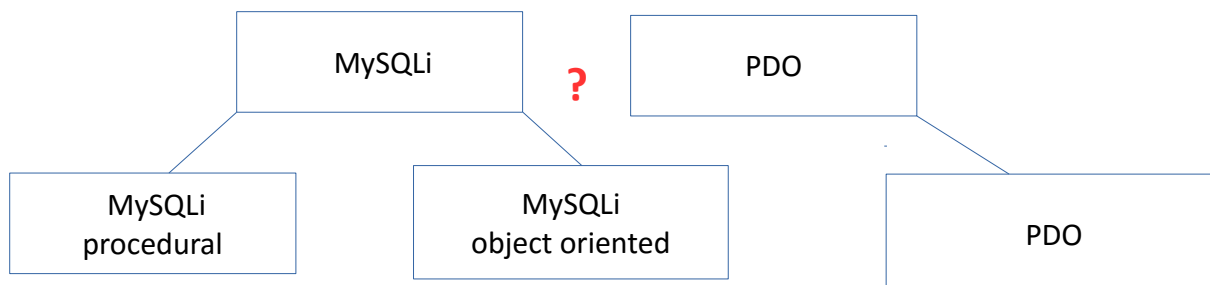


# Tres formas de manipular una MySQL BD



PHP 5 y posteriores pueden manipular bases de datos MySQL usando:

- la extensión MySQLi (la "i" significa mejora (improved) )
- PDO (PHP Data Objects)

Versiones anteriores de PHP usan extensión MySQL. Sin embargo, esta extensión caducó en el 2012.

## Cuál debo usar: MySQLi o PDO?

Realmente, la que prefieras. Tanto MySQLi como PDO tienen sus ventajas:

- PDO funciona con 12 sistemas de bases de datos diferentes, mientras que MySQLi únicamente funciona con las bases de datos de MySQL.
- Por tanto, si tienes que migrar tu proyecto a otra base de datos, PDO puede hacer más fácil el proceso. Sólo habría que cambiar la cadena de conexión y algunas queries. Mientras que si estás usando MySQLi, tendrías que reescribir el código entero, incluyendo las queries.
- Ambos modelos son object-oriented, pero MySQLi también ofrece procedimientos API.
- Ambos soportan instrucciones preparadas. Las sentencias preparadas protegen de la inyección de SQL, y son muy importantes para la seguridad de las aplicaciones web.

## Ejemplos de MySQL en MySQLi y PDO

Vamos a mostrar las tres formas de trabajar con PHP y MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

# MySQLi: Instalación

Para Linux y Windows: La extensión MySQLi se instala automáticamente en la mayoría de los casos, con la versión de php 5.

Si quieres ver detalles de la instalación ver: <http://php.net/manual/en/mysqli.installation.php>

---

# PDO: Instalación

Si quieres ver detalles de la instalación ver: <http://php.net/manual/en/pdo.installation.php>

## 1. Abrir una conexión a MySQL

Lo primero que debe hacerse para acceder a una base de datos en MySQL es conectarse al servidor:

### (a) Ejemplo (MySQLi Object-Oriented)

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

// Crea la conexión
$conn = new mysqli($servername, $username, $password);

// Comprueba conexión
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

*Note: on the object-oriented example above: \$connect\_error was broken until PHP 5.2.9 and 5.3.0. If you need to ensure compatibility with PHP versions prior to 5.2.9 and 5.3.0, use the following code instead:*

```
// Check connection
if (mysqli_connect_error()) {
    die("Database connection failed: " . mysqli_connect_error());
}
```

### (b) Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
```

```
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

### (c) Example (PDO)

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

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}
?>
```

**Nota:** A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

## 2. Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

### (a) Example (MySQLi Object-Oriented)

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

## (b) Example (MySQLi Procedural)

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

## (c) Example (PDO)

```
$conn = null;
```

# 3. Create a MySQL Database Using MySQLi and PDO

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

## (a) Example (MySQLi Object-oriented)

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

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

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

**Note:**

When you create a new database, you must only specify the first three arguments to the `mysqli` object (servername, username and password).

If you have to use a specific port, add an empty string for the database-name argument, like this: `new mysqli("localhost", "username", "password", "", port)`

## (b) Example (MySQLi Procedural)

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

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

## (c) Example (PDO)

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

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDBPDO";
```

```

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Database created successfully<br>";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

### 3. Create a MySQL Table Using MySQLi and PDO

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg\_date":

```

CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)

```

#### (a) Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

```

```

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>

```

## (b) Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

```

```

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

### (c) Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to create table
    $sql = "CREATE TABLE MyGuests (
        id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        firstname VARCHAR(30) NOT NULL,
        lastname VARCHAR(30) NOT NULL,
        email VARCHAR(50),
        reg_date TIMESTAMP
    )";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Table MyGuests created successfully";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```



## 4. Insert Data Into MySQL Using MySQLi and PDO

### (a) Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

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

### (b) Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
```

```

if(mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

## (c) Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "New record created successfully";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

## 5. Get ID of The Last Inserted Record

If we perform an INSERT or UPDATE on a table with an AUTO\_INCREMENT field, we can get the ID of the last inserted/updated record immediately.

In the table "MyGuests", the "id" column is an AUTO\_INCREMENT field:

```
CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg_date TIMESTAMP  
)
```

### (a) Example (MySQLi Object-oriented)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if ($conn->query($sql) === TRUE) {  
    $last_id = $conn->insert_id;  
    echo "New record created successfully. Last inserted ID is: " . $last_id;  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}  
  
$conn->close();  
?>
```

### (b) Example (MySQLi Procedural)

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

```

$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

### (c) Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
    // use exec() because no results are returned
    $conn->exec($sql);
    $last_id = $conn->lastInsertId();
    echo "New record created successfully. Last inserted ID is: " . $last_id;
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

```

```
$conn = null;  
?>
```

## 6. Insert Multiple Records Into MySQL Using MySQLi and PDO

Multiple SQL statements must be executed with the `mysqli_multi_query()` function.

### (a) Example (MySQLi Object-oriented)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('Mary', 'Moe', 'mary@example.com')";  
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('Julie', 'Dooley', 'julie@example.com')";  
  
if ($conn->multi_query($sql) === TRUE) {  
    echo "New records created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}  
  
$conn->close();  
?>
```

## (b) Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

The PDO way is a little bit different:

## (c) Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
```

```

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // begin the transaction
    $conn->beginTransaction();
    // our SQL statements
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')");
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')");
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')");

    // commit the transaction
    $conn->commit();
    echo "New records created successfully";
}
catch(PDOException $e)
{
    // roll back the transaction if something failed
    $conn->rollback();
    echo "Error: " . $e->getMessage();
}

$conn = null;
?>

```

## 7. Prepared Statements and Bound Parameters

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements basically work like this:

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Compared to executing SQL statements directly, prepared statements have two main advantages:

- Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

## (a) Example (MySQLi with Prepared Statements)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES
(?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";
```



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

Nota: In our SQL, we insert a question mark (?) where we want to substitute in an integer, string, double or blob value.

## (b) Example (PDO with Prepared Statements)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
```

```

    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

    echo "New records created successfully";
}
catch(PDOException $e)
{
    echo "Error: " . $e->getMessage();
}
$conn = null;
?>

```

## 8. Select Data From a MySQL Database

### (a) Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

```

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

## (b) Example (MySQLi Procedural)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli_connect_error());  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests";  
$result = mysqli_query($conn, $sql);  
  
if (mysqli_num_rows($result) > 0) {  
    // output data of each row  
    while($row = mysqli_fetch_assoc($result)) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .  
$row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
  
mysqli_close($conn);  
?>
```

## (c) Example (MySQLi Object-oriented)

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {
```

```

    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]." ".
$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>

```

## 9. Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table and displays it in an HTML table:

### (a) Example (PDO)

```

<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }
}

```

```

function endChildren() {
    echo "</tr>" . "\n";
}
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>

```

## 10. Delete Data From a MySQL Table Using MySQLi and PDO

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name WHERE some_column = some_value
```

### (a) Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {

```

```

    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>

```

## (b) Example (MySQLi Procedural)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

## (c) Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to delete a record

```

```

$sql = "DELETE FROM MyGuests WHERE id=3";

// use exec() because no results are returned
$conn->exec($sql);
echo "Record deleted successfully";
}
catch(PDOException $e)
{
echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>

```

## 11. Update Data In a MySQL Table Using MySQLi and PDO

### (a) Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>

```

### (b) Example (MySQLi Procedural)

```

<?php

```

```

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

### (c) Example (PDO)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

    // Prepare statement
    $stmt = $conn->prepare($sql);

    // execute the query
    $stmt->execute();

    // echo a message to say the UPDATE succeeded
    echo $stmt->rowCount() . " records UPDATED successfully";
}

```



```
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

=====

## PHP mysqli\_connect() Function

PHP MySQLi = PHP MySQL Improved!

The MySQLi functions allows you to access MySQL database servers.

Note: The MySQLi extension is designed to work with MySQL version 4.1.13 or newer.

The MySQLi extension was introduced with PHP version 5.0.0. The MySQL Native Driver was included in PHP version 5.3.0.

For installation details, go to: <http://php.net/manual/en/mysqli.installation.php>

For runtime configuration details, go to: <http://php.net/manual/en/mysqli.configuration.php>

Open a new connection to the MySQL server:

```
<?php
$con = mysqli_connect("localhost","my_user","my_password","my_db");

// Check connection
if (mysqli_connect_errno()) {

    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```

### Sintaxis

```
mysqli_connect(host, username, password, dbname, port, socket);
```

### Return Value

Returns an object representing the connection to the MySQL server