PHP Connect to MySQL Server

In this tutorial you will learn how to connect to the MySQL server using PHP.

Ways of Connecting to MySQL through PHP

In order to store or access the data inside a MySQL database, you first need to connect to the MySQL database server. PHP offers two different ways to connect to MySQL server: **MySQLi**(Improved MySQL) and **PDO** (PHP Data Objects) extensions.

While the PDO extension is more portable and supports more than twelve different databases, MySQLi extension as the name suggests supports MySQL database only. MySQLi extension however provides an easier way to connect to, and execute queries on, a MySQL database server. Both PDO and MySQLi offer an object-oriented API, but MySQLi also offers a procedural API which is relatively easy for beginners to understand.

Tip: The PHP's MySQLi extension provides both speed and feature benefits over the PDO extension, so it could be a better choice for MySQL-specific projects.

Connecting to MySQL Database Server

In PHP you can easily do this using the `mysqli_connect()` function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:

Syntax: MySQLi, Procedural way

```
$link = mysqli_connect("hostname", "username", "password", "database");
```

Syntax: MySQLi, Object Oriented way

```
$mysqli = new mysqli("hostname", "username", "password", "database");
```

Syntax: PHP Data Objects (PDO) way

```
$pdo = new PDO("mysql:host=hostname;dbname=database", "username",
"password");
```

The *hostname* parameter in the above syntax specify the host name (e.g. `localhost`), or IP address of the MySQL server, whereas the *username* and *password* parameters specifies the credentials to access MySQL server, and the *database* parameter, if provided will specify the default MySQL database to be used when performing queries.

The following example shows how to connect to MySQL database server using MySQLi (both *procedural* and *object oriented* way) and PDO extension.

Example **Procedural** Object Oriented PDO **Download**

```php
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Print host information
echo "Connect Successfully. Host info: " .
mysqli_get_host_info($link);
?>
```

Note: The default username for MySQL database server is `root` and there is no password. However to prevent your databases from intrusion and unauthorized access you should set password for MySQL accounts.

Tip: Setting the `PDO::ATTR_ERRMODE` attribute to `PDO::ERRMODE_EXCEPTION` tells PDO to throw exceptions whenever a database error occurs.

Closing the MySQL Database Server Connection

The connection to the MySQL database server will be closed automatically as soon as the execution of the script ends. However, if you want to close it earlier you can do this by simply calling the PHP `mysqli_close()` function.

Example **Procedural** Object Oriented PDO **Download**

```php
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL
```

```
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Print host information
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);

// Close connection
mysqli_close($link);
?>
```