26. PHP – PHP and MySQL

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

What you should already have?

- You have gone through MySQL tutorial to understand MySQL Basics.
- Downloaded and installed a latest version of MySQL.
- Created database user **guest** with password **guest123**.
- If you have not created a database, then you would need root user and its password to create a database.

We have divided this chapter in the following sections:

- **Connecting to MySQL database** - Learn how to use PHP to open and close a MySQL database connection.
- **Create MySQL Database Using PHP** - This part explains how to create MySQL database and tables using PHP.
- **Delete MySQL Database Using PHP** - This part explains how to delete MySQL database and tables using PHP.
- **Insert Data To MySQL Database** - Once you have created your database and tables, then you would like to insert your data into created tables. This session will take you through real example on data insert.
- **Retrieving Data From MySQL Database** - Learn how to fetch records from MySQL database using PHP.
- **Using Paging through PHP** - This one explains how to show your query result into multiple pages and how to create the navigation link.
- **Updating Data Into MySQL Database** - This part explains how to update existing records into MySQL database using PHP.
- **Deleting Data From MySQL Database** - This part explains how to delete or purge existing records from MySQL database using PHP.
- **Using PHP To Backup MySQL Database** - Learn different ways to take backup of your MySQL database for safety purpose.

Connecting to MySQL Database

Opening a Database Connection

PHP provides **mysql_connect** function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or FALSE on failure.

Syntax

```
connection mysql_connect(server,user,passwd,new_link,client_flag);
```

Sr.No	Parameter & Description
1	**server** Optional – The host name running database server. If not specified, then default value is **localhost:3306**.
2	**user** Optional – The username accessing the database. If not specified, then default is the name of the user that owns the server process.
3	**passwd** Optional – The password of the user accessing the database. If not specified then default is an empty password.
4	**new_link** Optional – If a second call is made to mysql_connect() with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned.
5	**client_flags** Optional – A combination of the following constants – • **MYSQL_CLIENT_SSL** – Use SSL encryption • **MYSQL_CLIENT_COMPRESS** – Use compression protocol

	• **MYSQL_CLIENT_IGNORE_SPACE** – Allow space after function names • **MYSQL_CLIENT_INTERACTIVE** – Allow interactive timeout seconds of inactivity before closing the connection

NOTE – You can specify server, user, passwd in **php.ini** file instead of using them again and again in your every PHP scripts. Check php.ini fileconfiguration.

Closing Database Connection

Its simplest function **mysql_close** PHP provides to close a database connection. This function takes connection resource returned by mysql_connect function. It returns TRUE on success or FALSE on failure.

Syntax

```
bool mysql_close ( resource $link_identifier );
```

If a resource is not specified, then the last opened database is closed.

Example

Try the following example to open and close a database connection –

```
<?php

   $dbhost = 'localhost:3036';
   $dbuser = 'guest';
   $dbpass = 'guest123';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   echo 'Connected successfully';
   mysql_close($conn);
?>
```

Create MySQL Database Using PHP

Creating a Database

To create and delete a database, you should have admin privilege. It's very easy to create a new MySQL database. PHP uses **mysql_query** function to create a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax

```
bool mysql_query( sql, connection );
```

Sr.No	Parameter & Description
1	**sql** Required - SQL query to create a database
2	**connection** Optional - if not specified, then the last opened connection by mysql_connect will be used.

Example

Try the following example to create a database –

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   echo 'Connected successfully';

   $sql = 'CREATE Database test_db';
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not create database: ' . mysql_error());
   }


   echo "Database test_db created successfully\n";
   mysql_close($conn);
?>
```

Selecting a Database

Once you establish a connection with a database server, then it is required to select a particular database with which all your tables are associated.

This is required because there may be multiple databases residing on a single server and you can do work with a single database at a time.

PHP provides function **mysql_select_db** to select a database. It returns TRUE on success or FALSE on failure.

Syntax

```
bool mysql_select_db( db_name, connection );
```

Sr.No	Parameter & Description
1	**db_name** Required - Database name to be selected
2	**connection** Optional - if not specified, then the last opened connection by mysql_connect will be used.

Example

Here is the example showing you how to select a database.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'guest';
   $dbpass = 'guest123';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   echo 'Connected successfully';

   mysql_select_db( 'test_db' );
   mysql_close($conn);

?>
```

Creating Database Tables

To create tables in the new database, you need to do the same thing as creating the database. First create the SQL query to create the tables, then execute the query using mysql_query() function.

Example

Try the following example to create a table –

```
<?php

   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   echo 'Connected successfully';

   $sql = 'CREATE TABLE employee( '.
      'emp_id INT NOT NULL AUTO_INCREMENT, '.
      'emp_name VARCHAR(20) NOT NULL, '.
      'emp_address  VARCHAR(20) NOT NULL, '.
      'emp_salary   INT NOT NULL, '.
      'join_date    timestamp(14) NOT NULL, '.
      'primary key ( emp_id ))';
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not create table: ' . mysql_error());
   }

   echo "Table employee created successfully\n";

   mysql_close($conn);
?>
```

In case you need to create many tables, then it’s better to create a text file first and put all the SQL commands in that text file and then load that file into $sql variable and execute those commands.

Consider the following content in sql_query.txt file

```
CREATE TABLE employee(
   emp_id INT NOT NULL AUTO_INCREMENT,
   emp_name VARCHAR(20) NOT NULL,
   emp_address  VARCHAR(20) NOT NULL,
   emp_salary   INT NOT NULL,
   join_date    timestamp(14) NOT NULL,
   primary key ( emp_id ));
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $query_file = 'sql_query.txt';

   $fp = fopen($query_file, 'r');
   $sql = fread($fp, filesize($query_file));
   fclose($fp);
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not create table: ' . mysql_error());
   }

   echo "Table employee created successfully\n";
   mysql_close($conn);
?>
```

Delete MySQL Database Using PHP

Deleting a Database

If a database is no longer required, then it can be deleted forever. You can use pass an SQL command to **mysql_query** to delete a database.

Example

Try the following example to drop a database.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'DROP DATABASE test_db';
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not delete database db_test: ' . mysql_error());
   }

   echo "Database deleted successfully\n";
   mysql_close($conn);
?>
```

WARNING – It’s very dangerous to delete a database and any table. So before deleting any table or database you should make sure you are doing everything intentionally.

Deleting a Table

It’s again a matter of issuing one SQL command through **mysql_query** function to delete any database table. But be very careful while using this command because by doing so you can delete some important information you have in your table.

Example

Try the following example to drop a table –

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'DROP TABLE employee';
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not delete table employee: ' . mysql_error());
   }

   echo "Table deleted successfully\n";

   mysql_close($conn);
?>
```

Insert Data to MySQL Database

Data can be entered into MySQL tables by executing SQL INSERT statement through PHP function **mysql_query**. Below a simple example to insert a record into **employee** table.

Example

Try the following example to insert record into employee table.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'INSERT INTO employee '.
      '(emp_name,emp_address, emp_salary, join_date) '.
      'VALUES ( "guest", "XYZ", 2000, NOW() )';

   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not enter data: ' . mysql_error());
   }

   echo "Entered data successfully\n";

   mysql_close($conn);
?>
```

In real application, all the values will be taken using HTML form and then those values will be captured using PHP script and finally they will be inserted into MySQL tables.

While doing data insert its best practice to use function **get_magic_quotes_gpc()** to check if current configuration for magic quote is set or not. If this function returns false, then use function **addslashes()** to add slashes before quotes.

Example

Try this example by putting this code into add_employee.php, this will take input using HTML Form and then it will create records into database.

```
<html>
```

```
   <head>
      <title>Add New Record in MySQL Database</title>
   </head>

   <body>
      <?php
         if(isset($_POST['add'])) {
            $dbhost = 'localhost:3036';
            $dbuser = 'root';
            $dbpass = 'rootpassword';
            $conn = mysql_connect($dbhost, $dbuser, $dbpass);

            if(! $conn ) {
               die('Could not connect: ' . mysql_error());
            }

            if(! get_magic_quotes_gpc() ) {
               $emp_name = addslashes ($_POST['emp_name']);
               $emp_address = addslashes ($_POST['emp_address']);
            }else {
               $emp_name = $_POST['emp_name'];
               $emp_address = $_POST['emp_address'];
            }

            $emp_salary = $_POST['emp_salary'];

            $sql = "INSERT INTO employee ". "(emp_name,emp_address, emp_salary,
               join_date) ". "VALUES('$emp_name','$emp_address',$emp_salary, NOW())";

            mysql_select_db('test_db');
            $retval = mysql_query( $sql, $conn );

            if(! $retval ) {
               die('Could not enter data: ' . mysql_error());
            }

            echo "Entered data successfully\n";

            mysql_close($conn);
         }else {
```

```
         ?>

            <form method = "post" action = "<?php $_PHP_SELF ?>">
               <table width = "400" border = "0" cellspacing = "1"
                  cellpadding = "2">

                  <tr>
                     <td width = "100">Employee Name</td>
                     <td><input name = "emp_name" type = "text"
                        id = "emp_name"></td>
                  </tr>

                  <tr>
                     <td width = "100">Employee Address</td>
                     <td><input name = "emp_address" type = "text"
                        id = "emp_address"></td>
                  </tr>

                  <tr>
                     <td width = "100">Employee Salary</td>
                     <td><input name = "emp_salary" type = "text"
                        id = "emp_salary"></td>
                  </tr>

                  <tr>
                     <td width = "100"> </td>
                     <td> </td>
                  </tr>

                  <tr>
                     <td width = "100"> </td>
                     <td>
                        <input name = "add" type = "submit" id = "add"
                           value = "Add Employee">
                     </td>
                  </tr>

               </table>
            </form>
            <?php
         }
      ?>

   </body>
</html>
```

Retrieving Data from MySQL Database

Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function mysql_query. You have several options to fetch data from MySQL.

The most frequently used option is to use function **mysql_fetch_array()**. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

Following is a simple example to fetch records from **employee** table.

Example

Try the following example to display all the records from employee table.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "EMP ID :{$row['emp_id']}  <br> ".
         "EMP NAME : {$row['emp_name']} <br> ".
         "EMP SALARY : {$row['emp_salary']} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";

   mysql_close($conn);
?>
```

The content of the rows are assigned to the variable $row and the values in row are then printed.

NOTE – Always remember to put curly brackets when you want to insert an array value directly into a string.

In the above example, the constant **MYSQL_ASSOC** is used as the second argument to mysql_fetch_array(), so that it returns the row as an associative array. With an associative array, you can access the field by using their name instead of using the index.

PHP provides another function called **mysql_fetch_assoc()** which also returns the row as an associative array.

Example

Try the following example to display all the records from employee table using mysql_fetch_assoc() function.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );
   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_assoc($retval)) {
      echo "EMP ID :{$row['emp_id']}  <br> ".
         "EMP NAME : {$row['emp_name']} <br> ".
         "EMP SALARY : {$row['emp_salary']} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";

   mysql_close($conn);
?>
```

You can also use the constant **MYSQL_NUM**, as the second argument to mysql_fetch_array(). This will cause the function to return an array with numeric index.

Example

Try the following example to display all the records from employee table using MYSQL_NUM argument.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );
   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_NUM)) {
      echo "EMP ID :{$row[0]}  <br> ".
         "EMP NAME : {$row[1]} <br> ".
         "EMP SALARY : {$row[2]} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";

   mysql_close($conn);
?>
```

All the above three examples will produce the same result.

Releasing Memory

It is a good practice to release cursor memory at the end of each SELECT statement. This can be done by using PHP function **mysql_free_result()**. Below is the example to show how it has to be used.

Example

Try the following example.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_NUM)) {
      echo "EMP ID :{$row[0]}  <br> ".
         "EMP NAME : {$row[1]} <br> ".
         "EMP SALARY : {$row[2]} <br> ".
         "--------------------------------<br>";
   }

   mysql_free_result($retval);
   echo "Fetched data successfully\n";

   mysql_close($conn);
?>
```

While fetching data, you can write as complex SQL as you like. The procedure will remain the same as mentioned above.

Using Paging through PHP

It's always possible that your SQL SELECT statement query may result into thousands of records. But it is not good idea to display all the results on one page. So we can divide this result into many pages as per requirement.

Paging means showing your query result in multiple pages instead of just put them all in one long page.

MySQL helps to generate paging by using **LIMIT** clause which will take two arguments. First argument as OFFSET and second argument how many records should be returned from the database.

Following is a simple example to fetch records using **LIMIT** clause to generate paging.

Example

Try the following example to display 10 records per page.

```
<html>

   <head>
      <title>Paging Using PHP</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';

         $rec_limit = 10;
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
         }
         mysql_select_db('test_db');

         /* Get total number of records */
         $sql = "SELECT count(emp_id) FROM employee ";
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not get data: ' . mysql_error());
         }
         $row = mysql_fetch_array($retval, MYSQL_NUM );
         $rec_count = $row[0];

         if( isset($_GET{'page'} ) ) {
            $page = $_GET{'page'} + 1;
            $offset = $rec_limit * $page ;
         }else {
            $page = 0;
            $offset = 0;
         }

         $left_rec = $rec_count - ($page * $rec_limit);
         $sql = "SELECT emp_id, emp_name, emp_salary ".
            "FROM employee ".
            "LIMIT $offset, $rec_limit";
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not get data: ' . mysql_error());
         }

         while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
            echo "EMP ID :{$row['emp_id']}  <br> ".
               "EMP NAME : {$row['emp_name']} <br> ".
               EMP SALARY : {$row['emp_salary']} <br> ".
               "--------------------------------<br>";
         }

         if( $page > 0 ) {
            $last = $page - 2;
            echo "<a href = \"$_PHP_SELF?page = $last\">Last 10 Records</a> |";
            echo "<a href = \"$_PHP_SELF?page = $page\">Next 10 Records</a>";
         }else if( $page == 0 ) {
            echo "<a href = \"$_PHP_SELF?page = $page\">Next 10 Records</a>";
         }else if( $left_rec < $rec_limit ) {
            $last = $page - 2;
            echo "<a href = \"$_PHP_SELF?page = $last\">Last 10 Records</a>";
         }

         mysql_close($conn);
      ?>

   </body>
</html>
```

Updating Data into MySQL Database

Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function **mysql_query**.

Below is a simple example to update records into **employee** table. To update a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

Example

Try the following example to understand update operation. You need to provide an employee ID to update an employee salary.

```
<html>

   <head>
      <title>Update a Record in MySQL Database</title>
   </head>

   <body>
      <?php
         if(isset($_POST['update'])) {
            $dbhost = 'localhost:3036';
            $dbuser = 'root';
            $dbpass = 'rootpassword';

            $conn = mysql_connect($dbhost, $dbuser, $dbpass);

            if(! $conn ) {
               die('Could not connect: ' . mysql_error());
            }

            $emp_id = $_POST['emp_id'];
            $emp_salary = $_POST['emp_salary'];

            $sql = "UPDATE employee ". "SET emp_salary = $emp_salary ".
               "WHERE emp_id = $emp_id" ;
            mysql_select_db('test_db');
            $retval = mysql_query( $sql, $conn );

            if(! $retval ) {
               die('Could not update data: ' . mysql_error());
            }
            echo "Updated data successfully\n";

            mysql_close($conn);
         }else {
            ?>
```

```
                <form method = "post" action = "<?php $_PHP_SELF ?>">
                   <table width = "400" border =" 0" cellspacing = "1"
                      cellpadding = "2">

                      <tr>
                         <td width = "100">Employee ID</td>
                         <td><input name = "emp_id" type = "text"
                            id = "emp_id"></td>
                      </tr>

                      <tr>
                         <td width = "100">Employee Salary</td>
                         <td><input name = "emp_salary" type = "text"
                            id = "emp_salary"></td>
                      </tr>

                      <tr>
                         <td width = "100"> </td>
                         <td> </td>
                      </tr>

                      <tr>
                         <td width = "100"> </td>
                         <td>
                            <input name = "update" type = "submit"
                               id = "update" value = "Update">
                         </td>
                      </tr>

                   </table>
                </form>
             <?php
          }
       ?>

   </body>
</html>
```

Deleting Data from MySQL Database

Data can be deleted from MySQL tables by executing SQL DELETE statement through PHP function **mysql_query**.

Following is a simple example to delete records into **employee** table. To delete a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

Example

Try the following example to understand delete operation. You need to provide an employee ID to delete an employee record from employee table.

```
<html>

   <head>
      <title>Delete a Record from MySQL Database</title>
   </head>

   <body>
      <?php
         if(isset($_POST['delete'])) {
            $dbhost = 'localhost:3036';
            $dbuser = 'root';
            $dbpass = 'rootpassword';
            $conn = mysql_connect($dbhost, $dbuser, $dbpass);

            if(! $conn ) {
               die('Could not connect: ' . mysql_error());
            }

            $emp_id = $_POST['emp_id'];

            $sql = "DELETE employee ". "WHERE emp_id = $emp_id" ;
            mysql_select_db('test_db');
            $retval = mysql_query( $sql, $conn );

            if(! $retval ) {
               die('Could not delete data: ' . mysql_error());
            }
            echo "Deleted data successfully\n";

            mysql_close($conn);
         }else {
            ?>
               <form method = "post" action = "<?php $_PHP_SELF ?>">
                  <table width = "400" border = "0" cellspacing = "1"
                     cellpadding = "2">

                     <tr>
                        <td width = "100">Employee ID</td>
                        <td><input name = "emp_id" type = "text"
                           id = "emp_id"></td>
                     </tr>

                     <tr>
                        <td width = "100"> </td>
                        <td> </td>
                     </tr>

                     <tr>
                        <td width = "100"> </td>
                        <td>
                           <input name = "delete" type = "submit"
                              id = "delete" value = "Delete">
                        </td>
                     </tr>

                  </table>
               </form>
            <?php
         }
      ?>

   </body>
</html>
```

Using PHP to Backup MySQL Database

It is always a good practice to take a regular backup of your database. There are three ways you can use to take backup of your MySQL database.

- Using SQL Command through PHP.
- Using MySQL binary mysqldump through PHP.
- Using phpMyAdmin user interface.

Using SQL Command through PHP

You can execute SQL SELECT command to take a backup of any table. To take a complete database dump you will need to write separate query for separate table. Each table will be stored into separate text file.

Example

Try the following example of using SELECT INTO OUTFILE query for creating table backup −

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $table_name = "employee";
   $backup_file  = "/tmp/employee.sql";
   $sql = "SELECT * INTO OUTFILE '$backup_file' FROM $table_name";

   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not take data backup: ' . mysql_error());
   }
   echo "Backedup  data successfully\n";

   mysql_close($conn);
?>
```

There may be instances when you would need to restore data which you have backed up some time ago. To restore the backup, you just need to run LOAD DATA INFILE query like this −

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $table_name = "employee";
   $backup_file  = "/tmp/employee.sql";
   $sql = "LOAD DATA INFILE '$backup_file' INTO TABLE $table_name";

   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not load data : ' . mysql_error());
   }
   echo "Loaded  data successfully\n";

   mysql_close($conn);
?>
```

Using MySQL binary mysqldump through PHP

MySQL provides one utility **mysqldump** to perform database backup. Using this binary you can take complete database dump in a single command.

Example

Try the following example to take your complete database dump –

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';

   $backup_file = $dbname . date("Y-m-d-H-i-s") . '.gz';
   $command = "mysqldump --opt -h $dbhost -u $dbuser -p $dbpass ". "test_db |
gzip > $backup_file";

   system($command);
?>
```

Using phpMyAdmin user interface

If you have **phpMyAdmin** user interface available, then it is very easy for you to take backup of your database.

To back up your MySQL database using phpMyAdmin click on the "export" link on phpMyAdmin main page. Choose the database you wish to backup, check the appropriate SQL options and enter the name for the backup file.

27. PHP – PHP and AJAX

What is AJAX ?

- AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.

- Conventional web application transmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.

- With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

For complete learning on AJAX, please refer our AJAX Tutorial.

PHP and AJAX Example

To clearly illustrate how easy it is to access information from a database using Ajax and PHP, we are going to build MySQL queries on the fly and display the results on "ajax.html". But before we proceed, let's do a bit of ground work first. Create a table using the following command.

NOTE: We are assuming you have sufficient privilege to perform following MySQL operations

```
CREATE TABLE `ajax_example` (
  `name` varchar(50) NOT NULL,
  `age` int(11) NOT NULL,
  `sex` varchar(1) NOT NULL,
  `wpm` int(11) NOT NULL,
  PRIMARY KEY  (`name`)
)
```

Now dump the following data into this table using the following SQL statements:

```
INSERT INTO `ajax_example` VALUES ('Jerry', 120, 'm', 20);
INSERT INTO `ajax_example` VALUES ('Regis', 75, 'm', 44);
INSERT INTO `ajax_example` VALUES ('Frank', 45, 'm', 87);
INSERT INTO `ajax_example` VALUES ('Jill', 22, 'f', 72);
INSERT INTO `ajax_example` VALUES ('Tracy', 27, 'f', 0);
INSERT INTO `ajax_example` VALUES ('Julie', 35, 'f', 90);
```

Client Side HTML file

Now let’s have our client side HTML file which is ajax.html and it will have following code

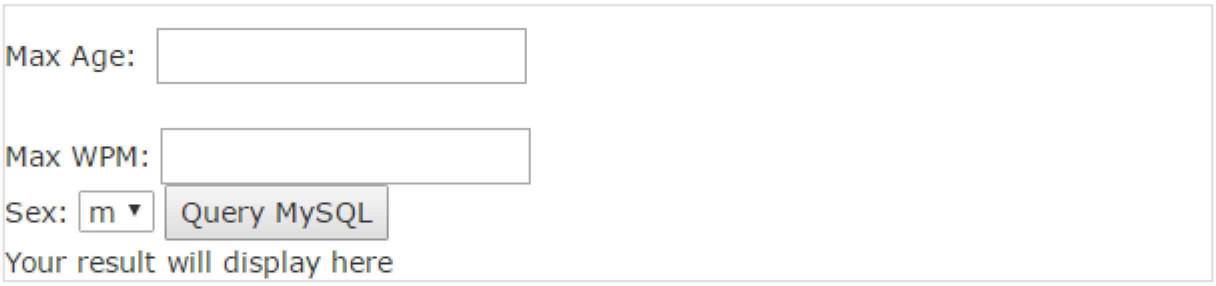
```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
//Browser Support Code
function ajaxFunction(){
 var ajaxRequest;  // The variable that makes Ajax possible!

 try{
   // Opera 8.0+, Firefox, Safari
   ajaxRequest = new XMLHttpRequest();
 }catch (e){
   // Internet Explorer Browsers
   try{
      ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
   }catch (e) {
      try{
         ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
      }catch (e){
         // Something went wrong
         alert("Your browser broke!");
         return false;
      }
   }
 }
 // Create a function that will receive data
 // sent from the server and will update
 // div section in the same page.
 ajaxRequest.onreadystatechange = function(){
   if(ajaxRequest.readyState == 4){
      var ajaxDisplay = document.getElementById('ajaxDiv');
      ajaxDisplay.innerHTML = ajaxRequest.responseText;
   }
 }
 // Now get the value from user and pass it to
 // server script.
 var age = document.getElementById('age').value;
 var wpm = document.getElementById('wpm').value;
 var sex = document.getElementById('sex').value;
 var queryString = "?age=" + age ;
 queryString +=  "&wpm=" + wpm + "&sex=" + sex;
 ajaxRequest.open("GET", "ajax-example.php" +
                              queryString, true);
 ajaxRequest.send(null);
}
//-->
</script>
<form name='myForm'>
Max Age: <input type='text' id='age' /> <br />
Max WPM: <input type='text' id='wpm' />
<br />
Sex: <select id='sex'>
<option value="m">m</option>
<option value="f">f</option>
</select>
<input type='button' onclick='ajaxFunction()'
                              value='Query MySQL'/>
</form>
<div id='ajaxDiv'>Your result will display here</div>
</body>
</html>
```

NOTE: The way of passing variables in the Query is according to HTTP standard and the have formA

```
URL?variable1=value1;&variable2=value2;
```

The above code will produce a screen as given below:

NOTE: This is dummy screen.



Your result will display here

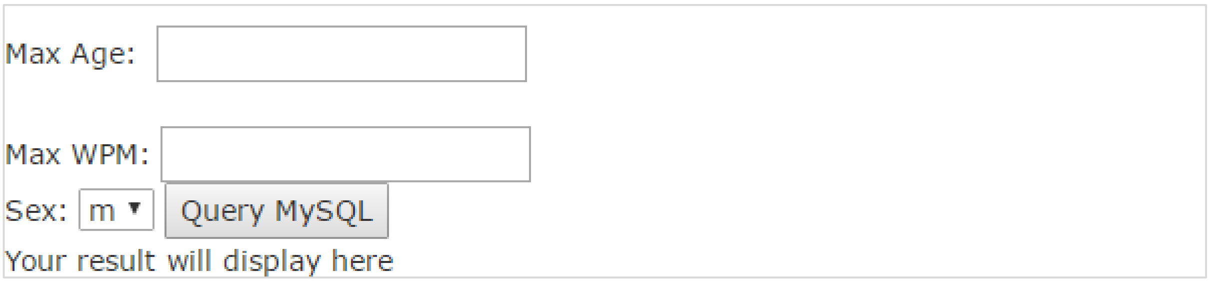
Server Side PHP file

So now your client side script is ready. Now we have to write our server side script which will fetch age, wpm and sex from the database and will send it back to the client. Put the following code into "ajax-example.php" file

```
<?php
$dbhost = "localhost";
$dbuser = "dbusername";
$dbpass = "dbpassword";
$dbname = "dbname";
     //Connect to MySQL Server
mysql_connect($dbhost, $dbuser, $dbpass);
     //Select Database
mysql_select_db($dbname) or die(mysql_error());
     // Retrieve data from Query String
$age = $_GET['age'];
$sex = $_GET['sex'];
$wpm = $_GET['wpm'];
     // Escape User Input to help prevent SQL Injection
$age = mysql_real_escape_string($age);
$sex = mysql_real_escape_string($sex);
$wpm = mysql_real_escape_string($wpm);
     //build query
$query = "SELECT * FROM ajax_example WHERE sex = '$sex'";
if(is_numeric($age))
     $query .= " AND age <= $age";
if(is_numeric($wpm))
     $query .= " AND wpm <= $wpm";
     //Execute query
```

```
$qry_result = mysql_query($query) or die(mysql_error());

     //Build Result String
$display_string = "<table>";
$display_string .= "<tr>";
$display_string .= "<th>Name</th>";
$display_string .= "<th>Age</th>";
$display_string .= "<th>Sex</th>";
$display_string .= "<th>WPM</th>";
$display_string .= "</tr>";

// Insert a new row in the table for each person returned
while($row = mysql_fetch_array($qry_result)){
     $display_string .= "<tr>";
     $display_string .= "<td>$row[name]</td>";
     $display_string .= "<td>$row[age]</td>";
     $display_string .= "<td>$row[sex]</td>";
     $display_string .= "<td>$row[wpm]</td>";
     $display_string .= "</tr>";

}
echo "Query: " . $query . "<br />";
$display_string .= "</table>";
echo $display_string;
?>
```

Now enter a valid value in "Max Age" or any other box and then click Query MySQL button.



If you have successfully completed this lesson, then you know how to use MySQL, PHP, HTML, and Javascript in tandem to write Ajax applications.

28. PHP – PHP and XML

XML is a markup language that looks a lot like HTML. An XML document is plain text and contains tags delimited by < and >. There are two big differences between XML and HTML:

- XML doesn't define a specific set of tags you must use.
- XML is extremely picky about document structure.

XML gives you a lot more freedom than HTML. HTML has a certain set of tags: the <a></a> tags surround a link, the <p> starts a paragraph and so on. An XML document, however, can use any tags you want. Put <rating></rating> tags around a movie rating, <height></height> tags around someone's height. Thus XML gives you option to device your own tags.

XML is very strict when it comes to document structure. HTML lets you play fast and loose with some opening and closing tags. But this is not the case with XML.

HTML list that's not valid XML

```
<ul>
<li>Braised Sea Cucumber
<li>Baked Giblets with Salt
<li>Abalone with Marrow and Duck Feet
</ul>
```

This is not a valid XML document because there are no closing </li> tags to match up with the three opening <li> tags. Every opened tag in an XML document must be closed.

HTML list that is valid XML

```
<ul>
<li>Braised Sea Cucumber</li>
<li>Baked Giblets with Salt</li>
<li>Abalone with Marrow and Duck Feet</li>
</ul>
```

Parsing an XML Document

PHP 5's new **SimpleXML** module makes parsing an XML document, well, simple. It turns an XML document into an object that provides structured access to the XML.

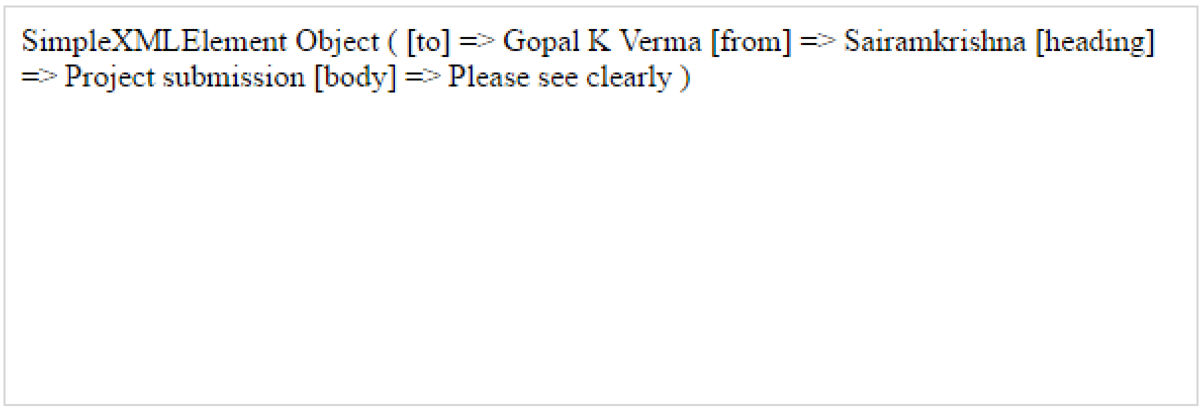
To create a SimpleXML object from an XML document stored in a string, pass the string to **simplexml_load_string( )**. It returns a SimpleXML object.

Example

Try out the following example:

```
<html>
   <body>

      <?php
         $note=<<<XML

         <note>
            <to>Gopal K Verma</to>
            <from>Sairamkrishna</from>
            <heading>Project submission</heading>
            <body>Please see clearly </body>
         </note>

         XML;
         $xml=simplexml_load_string($note);
         print_r($xml);
      ?>

   </body>
</html>
```

It will produce the following result:



```
SimpleXMLElement Object ( [to] => Gopal K Verma [from] => Sairamkrishna [heading]
=> Project submission [body] => Please see clearly )
```

NOTE: You can use function **simplexml_load_file(filename)** if you have XML content in a file.

For a complete detail of XML parsing function, check PHP Function Reference.

Generating an XML Document

SimpleXML is good for parsing existing XML documents, but you can't use it to create a new one from scratch.

The easiest way to generate an XML document is to build a PHP array whose structure mirrors that of the XML document and then to iterate through the array, printing each element with appropriate formatting.

Example

Try out the following example:

```
<?php


$channel = array('title' => "What's For Dinner",
                 'link' => 'http://menu.example.com/',
                 'description' => 'Choose what to eat tonight.');
print "<channel>\n";
foreach ($channel as $element => $content) {
   print " <$element>";
   print htmlentities($content);
   print "</$element>\n";
}
print "</channel>";
?>
```

It will produce the following result:

```
<channel>
<title>What's For Dinner</title>
<link>http://menu.example.com/</link>
<description>Choose what to eat tonight.</description>
</channel></html>
```