CS 85:
PHP PROGRAMMING

Module 9: MySQL & PHP

Using PHP to connect to MySQL Database

Create, Modify Delete Databases and Tables

Using PHP for Forms to add records

Santa Monica College
Computer Science &
Information Systems Dept.

In this module we will go over how to use PHP to connect and manipulate MySQL databases and tables. PHP ability to access and manipulate databases is one of the reason PHP is such a widely used backend scripting language. PHP is able to run SQL statements dynamically generated by PHP to add and delete records from MySQL tables. All the SQL statement you learned in the last module can now be used in PHP database query statement to directly control a MySQL database.

As you work through this module, remember that everything you learned about MySQL and SQL in the preceding module is valid here. The current version of PHP recommends using the new and improved packaged called mysqli (MySQL Improved).

MySQL Connection

Before PHP can perform any SQL statement to a SQL server, yourPHP code must connect to the MySQL server using the mysqli_connection() built in PHP function. The mysqli_connect() function opens a new connection to the MySQL server.

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

Parameter	Description
host	Optional. Specifies a host name or an IP address
username	Optional. Specifies the MySQL username
password	Optional. Specifies the MySQL password
dbname	Optional. Specifies the default database to be used
port	Optional. Specifies the port number to attempt to connect to the MySQL server
socket	Optional. Specifies the socket or named pipe to be used

Sample Code:

```php
<?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();
}

?>
```

The first argument in the mysqli_connection() is the hostname of the server running a MySQL server. The string “localhost” is commonly used for when the MySQL server is running on the same server as the Apache or Ngnix Web Server. If XAMPP is installed locally on your system, then Apache and MySQL can both be started locally on your system and used for testing your code.

When any mysqli_connection() connection is open, it is good practice to use mysqli_close() to close the connection before you script ends, thereby release system memory used to maintain the connection.

```php
<?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();
}

mysqli_close($con);

?>
```

MySQL Errors

When connecting to a MySQL server via PHP, it is common practice to check for errors or connection failures, by using the mysqli_error() or mysqi_connect_error functions. The function mysqli_error() will return the last error description for the most recent function called.

```php
<?php
$con=mysqli_connect("localhost","my_user","my_password","my_db");
// Check connection errors
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Perform a query, check for query errors
if (!mysqli_query($con,"INSERT INTO Persons (FirstName) VALUES
('Glenn')"))
{
    echo("Error description: " . mysqli_error($con));
}

mysqli_close($con);
?>
```

Hiding Errors

In general developer use error messages to debug and troubleshoot their code, but displaying error messages on a publicly accessible website can cause security issues. Users should not be aware of any server side issues. By using the error control operator (@) in front of any built in PHP function, can suppress the error or warning messages from displaying.

```php
<?php $DBName = "catalog";
//Notice the @ symbol in front of the mysqli_connect() function
$DBConnect = @mysqli_connect("hostname", "user", "password");
if ($DBConnect === FALSE) echo "<p>Connection error: " .
mysqli_error() . "</p>\n"; else {
    if (@mysqli_select_db($DBName, $DBConnect) === FALSE) {
        echo "<p>Could not select the \"$DBName\" " . "database: " .
mysqli_error($DBConnect) . "</p>\n";
        mysqli_close($DBConnect);
        $DBConnect = FALSE;
    }
}
?>
```

MySQL Create Databases via PHP

Depending on the permissions granted to your MySQL user, will determine which SQL query your MySQL user able to perform. For example, generally only the root user is able to CREATE DATABASE. If you script requires to initially create a database, then your user must have the privilege level to run that SQL command.

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

// Create connection did not specify database
$conn = mysqli_connect($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);
?>
```

Selecting a Database

Once you have either created your database via PHP mysqli_query() statement or the database already exist. To now create tables or add records to an exiting tables, your script needs to select the database to use similarly seen on mysql command line interface with the USE database statement.

If that database already exist then use the database argument in the mysqli_connect() function as seen in the example below.

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

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

mysqli_close($conn);
?>
```

Alternative, the mysqli_select_db() function can be called to select the database to use after the initial mysql connect has been made or the change the database to use mid script.

```php
<?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();
}

// ...some PHP code for database "my_db"...

// Change database to "test"
mysqli_select_db($con,"test");

// ...some PHP code for database "test"...

mysqli_close($con);
?>
```

Security Concern

Storing the username and password to your database in a publicly access PHP file is concerned a security risk. By using the include(), the username and password can be stored in a PHP file that not publicly accessible but accessible to the any PHP script running on the web server. The file should be outside the public web directory on the server.

Running a PHP Query

We have connected to the mysql server and then selected the database we want to use, now we can run an SQL query over the database to select information, do an insert, update or delete. To do this we use **mysqli_query**. This takes two arguments: the first is our link_identifier and the second is an SQL query string. If we are doing a select sql statement **mysqli_query** generates a resource or the Boolean false to say our query failed, and if we are doing a delete, insert or update it generates a Boolean, true or false, to say if that was successful or not.

The basic code for running a query is the php function "mysqli_query($cxn, $query)". The "$query" argument is a MySQL query. The database argument is a database connection (here, the connection represented by $conn). For example, to return the query " SELECT id, firstname, lastname FROM MyGuests ".

```
<?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);
?>
```

However, this straightforward method will quickly become ungainly due to the length of MySQL queries and the common need to repeat the query when handling the return. All (or almost all) queries are therefore made in two steps. First, the query is assigned a variable (conventionally, this variable is named "$query", “$sql”or "$sql_query" for purposes of uniformity and easy recognition), which allows the program to call simply "mysqli_query($conn, $sql_query)".

Secondly, to handle the information returned from the query, practical considerations require that the information returned also be assigned to a variable. Again by convention rather than necessity (i.e. you could name it anything you wanted), this information is often assigned to "$result", and the function is called by the assignment of the variable.

It is important to understand that this code **calls the function mysqli_query**, in addition to assigning the return to a variable "$result".

NOTE: The queries that ask for information -- SELECT, SHOW, DESCRIBE, and EXPLAIN -- return what is called a resource. Other types of queries, which manipulate the database, return TRUE if the operation is successful and FALSE if not, or if the user does not have permission to access the table referenced.

To catch an error, for debugging purposes, we can write:

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

If the function **mysqli_query** returns false, PHP will terminate the script and print an error report from MySQL such as "you have an error in your SQL syntax" and the query.

Thus, our final code would be, assuming that there was a database connection named $conn:

```
$sql = "SELECT * FROM customers ORDER BY customer_id ASC";
$result = mysqli_query ($conn, $sql_query) or die (mysqli_error () . "
The query was:" . $sql);
```

Putting it all together

In the previous sections we looked at three commands, but not at how to use them in conjunction with each other. So let's take a look at selecting information for a table in our mysql database called *MyTable*, which is stored in a mysql database called *MyDB*.

```
<?php
    //Connect to the mysql server and get back our link_identifier
    $link = mysqli_connect ("your_database_host", "your_user_name",
"your_password")or die('Could not connect: ' . mysqli_error());

    //Now, we select which database we would like to use
    mysqli_select_db ("MyDB", $link) or die('could not select
```

```
database');

   //Our SQL Query
   $sql_query = "Select * From MyTable";

   //Run our sql query
   $result = mysqli_query($sql_query)or die('query failed'.
mysqli_error());

    //Close Database Connection
    mysqli_close ($link);
 ?>
```

Creating and Deleting Tables

The process of creating the table is as easy as creating the database. We have to execute the create table query using the mysqli construct that will create a table with its own unique name and consists of columns and rows. The columns of the table are the fields that are defined with their own specific data types. Each table should have a primary key defined.

A mysqli_query() using the CREATE TABLE statement is the easiest way to create a tablet. Before executing the CREATE TABLE SQL statement, ensure you are in the correct database. By using the mysqli_select_db() function you can select your database before creating tables.

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

It is common in MySQL, and databases in general, is to create a numeric index that is used as a primary key identifier for each record. It could be a user id incrementing or a simple integer count. To identify a field as a primary key in MySQL, include the PRIMARY KEY keywords the tablet fields are defined with the CREATE TABLE statement. The AUTO_INCREMENT keyword is often used with a primary key to generate a unique ID for each new row in a table. For the first row inserted into a table, a field created with the AUTO_INCREMENT keyword is assigned a value of 1. The value of the field for each subsequently added row is incremented by 1 from the preceding row. Another keyword that is often used with primary keys is NOT NULL, which requires a field to include a value. Notice in the example below the id field has the AUTO_INCREMENT keywords.

To delete a table, you use the DROP TABLE statement with the mysqli_query() function.

Example #1

```php
<?php
$cn = mysqli_connect("localhost", "my_username", "my_password",
"MyDatabase");

if (mysqli_connect_errno()) {
    echo "Connection failed : " . mysqli_connect_error();
}
$sql_query = "CREATE TABLE  MyTable(firstName VARCHAR(18), lastName
VARCHAR(18), salary DECIMAL(5,4) )";

if (mysqli_query($cn, $sql_query)) {
    echo "Table created successfully";
} else {
    echo "Error encountered  while creating the table : " .
mysqli_error($cn);
}
?>
```

Example #2

```php
<?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);
?>
```

Select Query

The select query will select out the information from a database which is stored in a variable commonly named $result which becomes our resource identifier. A resource is a special type of PHP variable, but let's look at how to access information in this resource.

We can use a function called **mysqli_fetch_assoc()** it takes one parameter, our resource identifier *$result*, and generates an associative array corresponding to the fetched row. Each column in the table corresponds to an index with the same name. We can now extract out information and print it like so:

```
<?php
//Connect to the mysql server and get back our link identifier
$link = mysqli_connect("localhost", "your_user_name", "your_password")
or die('Could not connect: ' . mysqli_error());

//Now, we select which database we would like to use
mysqli_select_db("MyDB") or die('could not select database');

//Our SQL Query
$sql_query = "Select * From MyTable";

//Run our sql query
$result = mysqli_query($sql_query)or die('query failed'.
mysqli_error());

//iterate through result
while($row = mysqli_fetch_assoc($result))
{
    //Prints out information of that row
    print_r($row);
    echo $row['foo'];
    //Prints only the column foo.
}

// Free resultset (optional)
mysqli_free_result($result);

//Close the MySQL Link
mysqli_close($link);
?>
```

Inserting Records

After a database and a table have been created, we can start adding data. Here are some syntax rules to follow. To add records to your tables use the INSERT and VALUE SQL statements with the mysqli_query() function. When inserting data, you follow the format of specifying the column and then the value to add

to the tablet column. Any field with the AUTO_INCREMENT attribute can be skipped, it will auto increment and assign a value. A few rules when inserting records, the SQL query must be quoted in PHP, string values inside the SQL query must be quoted and numeric values must not be quoted. To insert a large number of records to a tables, use the LOAD DATA INFILE SQL statement.

Example #1

```
<?php
$cn=mysqli_connect("localhost","your_username","your_password","MyDB")
;
if (mysqli_connect_errno())
{
    echo "Connection failed : " .
        mysqli_connect_error();
}

mysqli_query($cn,"INSERT INTO MyTable(firstName, lastName, salary)
VALUES ('George','Smith' ,55000)");

mysqli_close($cn);
?>
```

Example #2

```
<?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);
?>
```

```
//Load records from file
```

```
<?php
// http://www.phpfreaks.com/forums/index.php?topic=290985.0

//create table in database
$sql1 = "
CREATE TABLE IF NOT EXISTS `test` (
  `id` int(11) NOT NULL auto_increment,
  `type` text NOT NULL,
  `command` text NOT NULL,
  `value` text NOT NULL,
  UNIQUE KEY `id` (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1";

$maketable = mysqli_query($sql1);
if (!$maketable) {
    die('Could not create table: ' . mysqli_error());
}
//load data into table
$sql2 = "LOAD DATA LOCAL INFILE 'temp.txt' INTO TABLE `test` FIELDS
TERMINATED BY ' ' ENCLOSED BY '\"' LINES TERMINATED BY '\\r\\n'
(`type` , `command` , `value`)";

$loaddata = mysqli_query($sql2);
if (!$loaddata) {
    die('Could not load data from file into table: ' .mysqli_error());
}
```

Updating Records

Use the UPDATE statement with the mysqli_query() function to update the record values. The UPDATE keyword specifies the name of the table to update and the SET keyword specifies the value to assign to the fields in the records that match the condition in the WHERE keyword.

```
<?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);
?>
```

Delete Records

A record can be delete from a table by using the DELETE statement with the mysqli_query() function. Using the DELETE SQL statement with the WHERE keyword to determine which records should be deleted. To delete all records in a table, do not use the WHERE keyword.

```
<?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);
?>
```

mysqli_affected_rows() Function

The mysqli_affected_rows() function returns the number of affected rows in the previous SELECT, INSERT, UPDATE, REPLACE, or DELETE query.

```
<?php
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Perform queries and print out affected rows
mysqli_query($con,"SELECT * FROM Persons");
echo "Affected rows: " . mysqli_affected_rows($con);
```

```
mysqli_query($con,"DELETE FROM Persons WHERE Age>32");
echo "Affected rows: " . mysqli_affected_rows($con);

mysqli_close($con);
?>
```

mysqli_num_rows() Function

The mysqli_num_rows() function returns the number of rows in a result set.

```
<?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();
}

$sql="SELECT Lastname,Age FROM Persons ORDER BY Lastname";

if ($result=mysqli_query($con,$sql))
{
    // Return the number of rows in result set
    $rowcount=mysqli_num_rows($result);
    printf("Result set has %d rows.\n",$rowcount);
    // Free result set
    mysqli_free_result($result);
}

mysqli_close($con);
?>
```

Example #2

```
<?php
$SQLstring = "DELETE FROM my_cars WHERE license='232FGSA'";
$QueryResult = mysqli_query($SQLstring, $DBConnect);
if ($QueryResult === FALSE) {
    echo "<p>Unable to execute the query.</p>" . "<p>Error code " .
mysqli_errno($DBConnect) . ": " . mysqli_error($DBConnect) . "</p>";
}
else {
    echo "<p>Successfully deleted " . mysqli_affected_rows($DBConnect)
. " record(s).</p>";
}
?>
```

Working with Query Results

When the mysqli_query() function is called, it returns a result pointer that represents the query results. It is common to assign the result pointer to a variable. Then by using functions like mysqli_fetch_row() with a loop to retrieve records one by one and add it into an index array.

Retrieving Records into an Indexed Array

The mysqli_fetch_row() function fetches one row from a result-set and returns it as an enumerated array. You can then use the array to access the individual fields in the row.

```
<?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();
}

$sql="SELECT Lastname,Age FROM Persons ORDER BY Lastname";

if ($result=mysqli_query($con,$sql))
{
    // Fetch one and one row
    while ($row=mysqli_fetch_row($result))
    {
        printf ("%s (%s)\n",$row[0],$row[1]);
    }
    // Free result set
    mysqli_free_result($result);
}

mysqli_close($con);
?>
```

Retrieving Records into an Associative Array

The mysqli_fetch_assoc() function fetches a result row as an associative array. The result is a required argument. It specifies a result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result(). The primary difference between the mysql_fetch_assoc() function and the mysql_fetch_row() function is that instead of returning the fields into an indexed array, the mysql_fetch_assoc() returns the fields into an associative array and uses each field name as the array key.

Note: Fieldnames returned from this function are case-sensitive.

```
<?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();
}
```

```
$sql="SELECT Lastname,Age FROM Persons ORDER BY Lastname";
$result=mysqli_query($con,$sql);

// Associative array
$row=mysqli_fetch_assoc($result);
printf ("%s (%s)\n",$row["Lastname"],$row["Age"]);

// Free result set
mysqli_free_result($result);

mysqli_close($con);
?>
```

Closing Query Results

The mysqli_free_result() function frees the memory associated with the result. The result is a required argument. It specifies a result set identifier returned by mysqli_query(), mysqli_store_result() or mysqli_use_result(). If you do not call mysqli_free_result(), then the memory used for the query is not released till the script ends.

```
<?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();
}

$sql="SELECT Lastname,Age FROM Persons ORDER BY Lastname";

if ($result=mysqli_query($con,$sql))
{
    // Fetch one and one row
    while ($row=mysqli_fetch_row($result))
    {
        printf ("%s (%s)\n",$row[0],$row[1]);
    }
    // Free result set from memory
    mysqli_free_result($result);
}

mysqli_close($con);
?>
```