

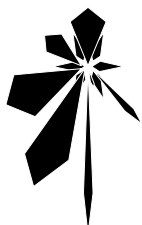
Introduction to Database Programming using PHP and MySQL

Revision 1.1

June 14, 2010

by

Aloysius Indrayanto



(C) 2010 AnemoneSoft.com

This document is multi-licensed under the Creative Commons Attribution Share-Alike (CC-BY-SA) license version 3.0 and the GNU Free Documentation License (GNU FDL) version 1.3 or later.

1. Introduction

LAMP (an acronym for Linux, Apache, MySQL, and PHP) is basically the most popular *solution stack* to deliver a general-purpose web and application server. A *solution stack* is a full set of software components bundled together to provide a specific solution. Basically, the exact combination of software included in LAMP can be modified, depending on the need of the developer. For example, PHP can be replaced with Python, Perl, or other scripting language. In MS Windows, the package is called WAMP. In Mac OS, it is called MAMP. In Solaris, it is called SAMP. In OpenBSD it is called OpAMP, etc.

PHP: Hypertext Preprocessor was invented by Rasmus Lerdorf in 1995. PHP is both a general-purpose programming language and a scripting language. PHP can be used to develop general-purpose applications (with and without graphical interface) as well as dynamic web pages or web applications. PHP is an interpreted language. Currently, PHP is maintained by The PHP Group. PHP is free and open source.

MySQL Community Server is a free and open source RDBMS (Relational Database Management System) that runs as a server to provide a multi-user database system. The original MySQL was developed by Michael Widenius. Currently, MySQL is owned and sponsored by Sun Microsystems, a subsidiary of Oracle Corporation. MySQL has been used in large-scale products, such as Wikipedia, Google, and Facebook.

2. Requirements

A basic knowledge in programming using PHP will be needed to understand the topic discussed in this tutorial. A system with an installation of Apache Web Server (HTTPD) version 2.2.x, PHP: Hypertext Preprocessor version 5.x, MySQL Community Server version 5.x, and a recent enough browser will be needed to run the code snippets.

A test database will be also needed to run the code snippets. Please login to a MySQL console as the *root* user and type these commands:

```
CREATE DATABASE MyTestDB;  
GRANT ALL PRIVILEGES ON MyTestDB.* TO 'mytestuser'@'localhost' IDENTIFIED BY 'mypassword';
```

The above commands will create a new database named 'MyTestDB' and a new user named 'mytestuser' with password 'mypassword'.

3. Using PHP's MySQL Extension

This extension allows a developer to develop PHP applications that access a MySQL database. This extension provides a procedural interface with access to all MySQL features up to MySQL server version 4.1.3. This extension is forward compatible to all newer versions of MySQL, even though features from newer MySQL servers will not be accessible using this extension.

Before connecting to a database, it is a good idea to store all the needed information in constants. Hence, it would be easier to modify the code if later those information changed.

```
// Access information
define('DB_HOST',      'localhost' );
define('DB_USER',      'mytestuser');
define('DB_PASSWORD', 'mypassword');
define('DB_DATABASE', 'MyTestDB' );
```

To connect to a MySQL server using the information above and directly select a database, one can use:

```
// Connect to a MySQL server
$mysql = @mysql_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_DATABASE);
if(!$mysql) die(mysql_error());
```

The at '@' sign in the front of the function call above is put there so that PHP will not print error message in case the connection failed. If the connection failed, the function `mysql_connect()` will return `false`, and the function `mysql_error()` will return the error message. To select another database, one can use the code:

```
// Select a database
if(!mysql_select_db(DB_DATABASE, $mysql)) die(mysql_error());
```

Finally, to close a connection, this code can be used:

```
// Close the connection
mysql_close($mysql);
```

3.1. Performing Queries that Do Not Return Result

To perform this type of queries, the function `mysql_query()` can be used. This function will return `true` on success and `false` on error. The code below will perform queries that will delete the table `TestTable1` if it is exist and (re-)create it. In case of error, the function `mysql_error()` will be used the obtain the error message.

```
// Delete a table (if exist)
$sql = 'DROP Table IF EXISTS TestTable1';
if(!mysql_query($sql, $mysql)) die(mysql_error());

// Create a new table
$sql = <<<EOT
    Create Table TestTable1 (ID    INT                NOT NULL PRIMARY KEY AUTO_INCREMENT,
                           Name  VARCHAR(255) NOT NULL,
                           Age   TINYINT         NOT NULL) ENGINE=InnoDB
EOT;
if(!mysql_query($sql, $mysql)) die(mysql_error());
```

Note that the clause `IF EXISTS` is MySQL extension and may not be supported by other database vendors. The next code will execute queries that will insert records to the newly created table.

```
// Insert records
$sql = <<<EOT
    INSERT INTO TestTable1 (Name, Age)
    VALUES
        ("Mr. One", 25), ("Mr. Two", 27)
EOT;
if(!mysql_query($sql, $mysql)) die(mysql_error());

echo 'Last inserted ID = ' . mysql_insert_id($mysql) . '<br/>';
echo '<br/>';
```

Inserting multiple records using one `INSERT` statement is MySQL extension and may not be supported by other database vendors. The `ID` column is an `AUTO_INCREMENT` field; to get the last inserted ID for this type of field, the function `mysql_insert_id()` can be used. In case of inserting multiple records using one `INSERT` statement, the function will return the ID of the first inserted record.

3.2. Performing Queries that Return Results

To perform this type of queries, the same function, `mysql_query()` can be used. This function will return a resource on success and `false` on error. The code below will perform a `SELECT` query on the above table and the print the number of resulting rows and the number of columns (fields) in each row using the `mysql_num_rows()` and `mysql_num_fields()` functions.

```
// Select records
$sql = 'SELECT * FROM TestTable1'
$result = mysql_query($sql, $mysql);
if(!$result) die(mysql_error());

echo 'The number of resulting rows = ' . mysql_num_rows($result) . '<br/>';
echo 'The number of fields in each rows = ' . mysql_num_fields($result) . '<br/>';
echo '<br/>';
```

The function `mysql_fetch_field()` will return an object containing some information about the result. One of the information is the name of each column of the result. The code construct below

can be used to print all the column names from a result:

```
// Print the column names
while($field = mysql_fetch_field($result)) {
    echo $field->name . '&nbsp;&nbsp;&nbsp;';
}
echo '<br/>';
```

Next, the function `mysql_fetch_row()` can be used to get all the records (rows) one by one. The code construct below can be used to print all the column contents of all rows:

```
// Print the contents of the rows
while($row = mysql_fetch_row($result)) {
    for($i = 0; $i < mysql_num_fields($result); ++$i) {
        echo $row[$i] . '&nbsp;&nbsp;&nbsp;';
    }
    echo '<br/>';
}
echo '<br/>';
```

Finally, it is a good idea to free the result object by calling:

```
// Free result
mysql_free_result($result);
```

The function call above does not always necessary, however, it is the simplest solution to prevent the “Commands out of sync; you can't run this command now” error that may occur if one tries to execute another query while the result of the current query has not yet fully used (not all the rows have been fetched).

Sometime, it is necessary to get the number of rows that are affected by a query. The function `mysql_affected_rows()` can be used for this purpose. The code below will delete all records from the above table and then print the number of deleted records.

```
// Delete records
$sql = 'DELETE FROM TestTable1';
if(!mysql_query($sql, $mysql)) die(mysql_error());

echo 'The number of deleted rows = ' . mysql_affected_rows($mysql) . '<br/>';
echo '<br/>';
```

3.3. The Full Code Snippet

The code snippet below demonstrate all the features that have been explained above. Try to experiment with the code.

```
<?php
/*
    Copyright (C) 2010 AnemoneSoft.com
    Aloysius Indrayanto
*/

// Access information
define('DB_HOST',      'localhost' );
define('DB_USER',      'mytestuser');
define('DB_PASSWORD', 'mypassword');
define('DB_DATABASE', 'MyTestDB' );

// Connect to a MySQL server
$mysql = @mysql_connect(DB_HOST, DB_USER, DB_PASSWORD, '');
if(!$mysql) die(mysql_error());

// Select a database
if(!mysql_select_db(DB_DATABASE, $mysql)) die(mysql_error());

// Delete a table (if exist)
$sql = 'DROP Table IF EXISTS TestTable1';
if(!mysql_query($sql, $mysql)) die(mysql_error());

// Create a new table
$sql = <<<EOT
    Create Table TestTable1 (ID      INT          NOT NULL PRIMARY KEY AUTO_INCREMENT,
                             Name  VARCHAR(255) NOT NULL,
                             Age   TINYINT     NOT NULL) ENGINE=InnoDB
EOT;
if(!mysql_query($sql, $mysql)) die(mysql_error());

// Insert records
$sql = <<<EOT
    INSERT INTO TestTable1 (Name, Age)
    VALUES
        ("Mr. One", 25), ("Mr. Two", 27)
EOT;
if(!mysql_query($sql, $mysql)) die(mysql_error());

echo 'Last inserted ID = ' . mysql_insert_id($mysql) . '<br/>';
echo '<br/>';

// Select records
$sql = 'SELECT * FROM TestTable1';
$result = mysql_query($sql, $mysql);
if(!$result) die(mysql_error());

echo 'The number of resulting rows = ' . mysql_num_rows($result) . '<br/>';
echo 'The number of fields in each rows = ' . mysql_num_fields($result) . '<br/>';
echo '<br/>';

// Print the column names
while($field = mysql_fetch_field($result)) {
    echo $field->name . '&nbsp;&nbsp;&nbsp;';
}
echo '<br/>';

// Print the contents of the rows
while($row = mysql_fetch_row($result)) {
    for($i = 0; $i < mysql_num_fields($result); ++$i) {
        echo $row[$i] . '&nbsp;&nbsp;&nbsp;';
    }
    echo '<br/>';
}
echo '<br/>';

// Free result
mysql_free_result($result);

// Delete records
$sql = 'DELETE FROM TestTable1';
if(!mysql_query($sql, $mysql)) die(mysql_error());
```

```
echo 'The number of deleted rows = ' . mysql_affected_rows($mysql) . '<br/>';
echo '<br/>';

// Close the connection
mysql_close($mysql);
?>
```

tut01.php: Using PHP's MySQL Extension.

4. Using PHP's MySQLi Extension

This extension allows a developer to develop PHP applications that access a MySQL database. This extension provides both procedural (deprecated) and object-oriented interfaces. This extension provides access to newer MySQL features, such as multi-statements, prepared statements, etc. Note that these advanced features of MySQL will not be discussed in this tutorial.

Before connecting to a database, it is a good idea to store all the needed information in constants. Hence, it would be easier to modify the code if later those information changed. To connect to a MySQL server and select a database, one can use:

```
// Access information
define('DB_HOST',      'localhost' );
define('DB_USER',      'mytestuser');
define('DB_PASSWORD', 'mypassword');
define('DB_DATABASE', 'MyTestDB' );

// Connect to a MySQL server
$mysqli = @new MySQLi(DB_HOST, DB_USER, DB_PASSWORD, '');
if($mysqli->connect_errno) die($mysqli->connect_error);

// Select a database
if(!$mysqli->select_db(DB_DATABASE)) die($mysqli->error);
```

In this tutorial, the object-oriented interface of MySQLi is used. The '@' sign in the front of the object creation is put there so that PHP will not print error message in case the connection failed. If the connection failed, the property `connect_errno` of the returned MySQLi object will contain a non zero value while the property `connect_error` will contain the error message. For other kind of errors, the error message will be in the `error` property of the MySQLi object. Finally, to close a connection, simply setting the MySQLi object to `null` is enough.

```
// Close the connection
$mysqli = null;
```

4.1. Performing Queries that Do Not Return Result

To perform this type of queries, the method `query()` can be used. This method will return `true` on success and `false` on error. The code below will perform queries that will delete table `TestTable1` if it is exist and (re-)create it.

```
// Delete a table (if exist)
$sql = 'DROP Table IF EXISTS TestTable1';
mysqli->query($sql);
if(mysqli->errno) die(mysqli->error);

// Create a new table
$sql = <<<EOT
    Create Table TestTable1 (ID    INT                NOT NULL PRIMARY KEY AUTO_INCREMENT,
                           Name VARCHAR(255) NOT NULL,
                           Age  TINYINT           NOT NULL) ENGINE=InnoDB
EOT;
mysqli->query($sql);
if(mysqli->errno) die(mysqli->error);
```

Note that the clause `IF EXISTS` is MySQL extension and may not be supported by other database vendors. The next code will execute queries that will insert records to the newly created table.

```
// Insert records
$sql = <<<EOT
    INSERT INTO TestTable1 (Name, Age)
    VALUES ("Mr. One", 25), ("Mr. Two", 27)
EOT;
mysqli->query($sql);
if(mysqli->errno) die(mysqli->error);

echo 'Last inserted ID = ' . mysqli->insert_id . '<br/>';
echo '<br/>';
```

Inserting multiple records using one `INSERT` statement is MySQL extension and may not be supported by other database vendors. The `ID` column is an `AUTO_INCREMENT` field; to get the last inserted ID for this type of field, the property `insert_id` of the `MySQLi` object can be read. In case of inserting multiple records using one `INSERT` statement, the function will return the ID of the first inserted record.

4.2. Performing Queries that Return Results

To perform this type of queries, the same method, `query()` can be used. This function will return a result object on success and `false` on error. The code below will perform a `SELECT` query on the above table and the print the number of resulting rows and the number of columns (fields) in each row by reading the `num_rows` and `field_count` properties of the result object.

```
// Select records
$sql = 'SELECT * FROM TestTable1';
mysqli->query($sql);
if(mysqli->errno) die(mysqli->error);

$result = mysqli->query('SELECT * FROM TestTable1', MYSQLI_STORE_RESULT);
if(mysqli->errno) die(mysqli->error);

echo 'The number of resulting rows = ' . $result->num_rows . '<br/>';
echo 'The number of fields in each rows = ' . $result->field_count . '<br/>';
echo '<br/>';
```

The method `fetch_field()` will return an object containing some information about the result. One of the information is the name of each column of the result. The code construct below can be used to print all the column names from a result object:

```
// Print the column names
while($field = $result->fetch_field()) {
    echo $field->name . '&nbsp;&nbsp;&nbsp;';
}
echo '<br/>';
```

Next, the function `fetch_row()` can be used to get all the records (rows) one by one. The code construct below can be used to print all the column contents of all rows:

```
// Print the contents of the rows
while($row = $result->fetch_row()) {
    for($i = 0; $i < $result->field_count; ++$i) {
        echo $row[$i] . '&nbsp;&nbsp;&nbsp;';
    }
    echo '<br/>';
}
echo '<br/>';
```

Finally, it is a good idea to free the result object by calling:

```
// Free result
$result->free();
```

The method call above does not always necessary, however, it is the simplest solution to prevent the “Commands out of sync; you can't run this command now” error that may occur if one tries to execute another query while the result of the current query has not yet fully used (not all the rows have been fetched).

This out of sync problem will most likely to occur if one has used `MYSQLI_USE_RESULT` instead of `MYSQLI_STORE_RESULT` in the call of `query()`. Basically, if the number of returned records is expected to be large and there will be only small (or no) processing in the PHP code, it may be better to use `MYSQLI_USE_RESULT` rather than `MYSQLI_STORE_RESULT` to reduce the memory usage of the PHP script. However, using `MYSQLI_USE_RESULT` together with table locks or transaction may cause other clients to be locked-up from using the same tables until the

current query has fully completed (all the rows have been fetched). Therefore, the default is to use `MYSQLI_STORE_RESULT` (hence, if the second parameter in the call of `query()` is omitted, it will resolve to `MYSQLI_STORE_RESULT`).

Sometime, it is necessary to get the number of rows affected by a query. The property `affected_rows` from the `MySQLi` object can be read for this purpose. The code below will delete all records from the above table and then print the number of deleted records.

```
// Delete records
$sql = 'DELETE FROM TestTable1';
if($mysqli->errno) die($mysqli->error);

echo 'The number of deleted rows = ' . $mysqli->affected_rows . '<br/>';
echo '<br/>';
```

4.3. The Full Code Snippet

The code snippet below demonstrate all the features that have been explained above. Try to experiment with the code.

```
<?php
/*
    Copyright (C) 2010 AnemoneSoft.com
    Aloysius Indrayanto
*/

// Access information
define('DB_HOST', 'localhost');
define('DB_USER', 'mytestuser');
define('DB_PASSWORD', 'mypassword');
define('DB_DATABASE', 'MyTestDB');

// Connect to a MySQL server
$mysqli = @new MySQLi(DB_HOST, DB_USER, DB_PASSWORD, '');
if($mysqli->connect_errno) die($mysqli->connect_error);

// Select a database
if(!$mysqli->select_db(DB_DATABASE)) die($mysqli->error);

// Delete a table (if exist)
$sql = 'DROP Table IF EXISTS TestTable1';
$mysqli->query($sql);
if($mysqli->errno) die($mysqli->error);

// Create a new table
$sql = <<<EOT
    Create Table TestTable1 (ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
                            Name VARCHAR(255) NOT NULL,
                            Age TINYINT NOT NULL) ENGINE=InnoDB
EOT;
$mysqli->query($sql);
if($mysqli->errno) die($mysqli->error);

// Insert records
$sql = <<<EOT
    INSERT INTO TestTable1 (Name, Age)
    VALUES ("Mr. One", 25), ("Mr. Two", 27)
EOT;
$mysqli->query($sql);
if($mysqli->errno) die($mysqli->error);
```

```
echo 'Last inserted ID = ' . $mysqli->insert_id . '<br/>';
echo '<br/>';

// Select records
$sql = 'SELECT * FROM TestTable1';
$mysqli->query($sql);
if($mysqli->errno) die($mysqli->error);

$result = $mysqli->query('SELECT * FROM TestTable1', MYSQLI_STORE_RESULT);
if($mysqli->errno) die($mysqli->error);

echo 'The number of resulting rows = ' . $result->num_rows . '<br/>';
echo 'The number of fields in each rows = ' . $result->field_count . '<br/>';
echo '<br/>';

// Print the column names
while($field = $result->fetch_field()) {
    echo $field->name . '&nbsp;&nbsp;&nbsp;';
}
echo '<br/>';

// Print the contents of the rows
while($row = $result->fetch_row()) {
    for($i = 0; $i < $result->field_count; ++$i) {
        echo $row[$i] . '&nbsp;&nbsp;&nbsp;';
    }
    echo '<br/>';
}
echo '<br/>';

// Free result
$result->free();

// Delete records
$sql = 'DELETE FROM TestTable1';
if($mysqli->errno) die($mysqli->error);

echo 'The number of deleted rows = ' . $mysqli->affected_rows . '<br/>';
echo '<br/>';

// Close the connection
$mysqli = null;
?>
```

tut02.php: Using PHP's MySQLi Extension.

5. Preventing SQL-Injection Attack

SQL-injection is a technique that exploit a security vulnerability in a database-enabled application. This vulnerability may present if the application failed to filter user inputs (especially for string values) for malicious string-escape or code.

The simplest method to prevent this type of attack is to convert all numerical values (integers and floats) to PHP variables with the correct types and to escape all string values. When using the PHP's MySQL extension, this can be done by using the function `mysql_real_escape_string()`. This function call needs a valid connection to a MySQL server. The code snippet below demonstrate this feature:

```
echo mysql_real_escape_string('Mr. Q is "crazy"!', $mysql) . '<br/>';
```

When using the PHP's MySQLi extension, the `real_escape_string()` method from the MySQLi object can be used.

```
echo $mysqli->real_escape_string('Mr. Q is "crazy"!') . '<br/>';
```

Note that, when using MySQLi, the better way is to use prepared-statements rather than escaping strings. This topic (together with other more advanced topics) will not be discussed in this tutorial.

References

<http://httpd.apache.org>, June 09, 2010

<http://php.net/index.php>, June 09, 2010

<http://www.mysql.com>, June 09, 2010

<http://www.wampserver.com/en>, June 09, 2010

<http://en.wikipedia.org/wiki/Php>, June 09, 2010

<http://php.net/manual/en>, June 09, 2010

<http://en.wikipedia.org/wiki/MySQL>, June 09, 2010

[http://en.wikipedia.org/wiki/LAMP_\(software_bundle\)](http://en.wikipedia.org/wiki/LAMP_(software_bundle)), June 09, 2010

<http://en.wikipedia.org/wiki/WAMP>, June 09, 2010

<http://en.wikipedia.org/wiki/MAMP>, June 09, 2010

http://en.wikipedia.org/wiki/SQL_injection, June 12, 2010