PHP MySqli Basic usage (select, insert & update)

-  Saran Chamling
- 33 Comments
-

Share

http://twitter.com/home?status=PHP%20MySqli%20Basic%20usage%20(select,%20insert%20&%20update)+http%3A%2F%2Fj.mp%2F28NHHZGhttps://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.sanwebe.com%2F2013%2F03%2Fbasic-php-mysqli-usagehttps://plus.google.com/share?url=https%3A%2F%2Fwww.sanwebe.com%2F2013%2F03%2Fbasic-php-mysqli-usage

-

After the deprecation of PHP MySQL extension in 2011, alternative extension MySqli and PDO are available in PHP. MySqli and PDO are improved version and offer an object-oriented API and number of enhancement over the regular MySql extension. These extensions are much faster, efficient and totally secure against SQL injections.
Today I'd like to show you the basic usage of MySqli, such as connect, select, insert, update and delete records. I hope this list will come in handy for you.

Installing MySqli

If you are running PHP version 5.3.0 +, MySqli should be available to use it right away, but in old PHP 5.0, 5.1, 5.2, extension is not enabled by default on

Windows Systems, you must enable *php_mysqli.dll* DLL inside of *php.ini*. To enable the extension you need to edit your *php.ini* and remove comment (semi-colon) from the start of the line extension=php_mysqli.dll. In linux too when you install php5 mysql package, MySQLi automatically gets installed, more details about installation in Linux and Windows can be found here.

Connect to Database

MySqli offers two ways to connect to the database, procedural and object oriented, the recommended way to open a database connection is object oriented way, because it is secure, faster and efficient. The procedural style is much similar to old MySql and it may be helpful to users who are just switching to MySqli, but should keep away altogether.

```PHP

1

//procedural style
2 $mysqli =  mysqli_connect('host','username','password','datab
ase_name');
3
//object oriented style (recommended)
4 $mysqli = new mysqli('host','username','password','database_n
ame');
5
```

Here's how you open a database connection "object-oriented" style, which is a recommended way and we will only be using this style in all the examples below.

```PHP

<?php
1 //Open a new connection to the MySQL server
2 $mysqli = new mysqli('host','username','password','database_
3 name');
4
5 //Output any connection error
6 if ($mysqli->connect_error) {
7     die('Error : ('. $mysqli->connect_errno .') '. $mysqli-
8 >connect_error);
9 }
1
0
?>
```

SELECT Multiple Records as Associative array

mysqli_fetch_assoc() : Below is the code to fetch multiple records as an associative array. The returned array holds the strings fetched from database, where the column names will be the key used to access the internal data. As you can see below, data is displayed in an HTML table.

PHP

```php
<?php
//Open a new connection to the MySQL server
$mysqli = new mysqli('host','username','password','database_name');

//Output any connection error
if ($mysqli->connect_error) {
    die('Error : ('. $mysqli->connect_errno .') '. $mysqli->connect_error);
}

//MySqli Select Query
$results = $mysqli->query("SELECT id, product_code, product_desc, price FROM products");

print '<table border="1">';
while($row = $results->fetch_assoc()) {
    print '<tr>';
    print '<td>'.$row["id"].'</td>';
    print '<td>'.$row["product_code"].'</td>';
    print '<td>'.$row["product_name"].'</td>';
    print '<td>'.$row["product_desc"].'</td>';
    print '<td>'.$row["price"].'</td>';
    print '</tr>';
}
print '</table>';

// Frees the memory associated with a result
$results->free();

// close connection
$mysqli->close();
?>
```

SELECT Multiple Records as Array

fetch_array() : Function returns an array of
both mysqli_fetch_row and mysqli_fetch assoc merged together, it is an extended version of the mysqli_fetch_row() function and both numeric and string can be used as keys to access the data.

PHP

```php
<?php
//Open a new connection to the MySQL server
$mysqli = new mysqli('host','username','password','database_name');

//Output any connection error
if ($mysqli->connect_error) {
    die('Error : ('. $mysqli->connect_errno .') '. $mysqli->connect_error);
}

//MySqli Select Query
$results = $mysqli->query("SELECT id, product_code, product_desc, price FROM products");

print '<table border="1"';
while($row = $results->fetch_array()) {
    print '<tr>';
    print '<td>'.$row["id"].'</td>';
    print '<td>'.$row["product_code"].'</td>';
    print '<td>'.$row["product_name"].'</td>';
    print '<td>'.$row["product_desc"].'</td>';
    print '<td>'.$row["price"].'</td>';
    print '</tr>';

}
print '</table>';

// Frees the memory associated with a result
$results->free();
```

```
// close connection
$mysqli->close();
?>
```

SELECT Multiple Records as Objects

fetch_object() : To fetch database result set as an objects, just use MySqli fetch_object(). The attributes of the object represent the names of the fields found within the result set.

PHP

```
<?php
//Open a new connection to the MySQL server
$mysqli = new mysqli('host','username','password','database_
name');

//Output any connection error
if ($mysqli->connect_error) {
    die('Error : ('. $mysqli->connect_errno .') '. $mysqli-
>connect_error);
}

//MySqli Select Query
$results = $mysqli->query("SELECT id, product_code,
product_desc, price FROM products");

print '<table border="1">';
while($row = $results->fetch_object()) {
    print '<tr>';
```

```
    print '<td>'.$row->id.'</td>';
    print '<td>'.$row->product_code.'</td>';
    print '<td>'.$row->product_name.'</td>';
    print '<td>'.$row->product_desc.'</td>';
    print '<td>'.$row->price.'</td>';
    print '</tr>';
}

print '</table>';

// close connection
$mysqli->close();
?>
```

SELECT Single value

How about getting a single value from database using fetch_object (Cameron Spear style).

PHP

```
<?php
//Open a new connection to the MySQL server
$mysqli = new mysqli('host','username','password','database_
name');

//Output any connection error
if ($mysqli->connect_error) {
    die('Error : ('. $mysqli->connect_errno .') '. $mysqli-
>connect_error);
}
```

```
//chained PHP functions
$product_name = $mysqli->query("SELECT product_name FROM products WHERE id = 1")->fetch_object()->product_name;
print $product_name; //output value

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

Above code will return error if there's no result, so here's safer way to retrieve the value.

PHP

```
$result = $mysqli->query("SELECT product_name FROM products WHERE id = 1");
if($result->num_rows > 0){
    echo $result->fetch_object()->product_name;
}
```

SELECT COUNT Total records of a table

Sometimes you may want to know total records of a table, especially for a pagination.

PHP

```

<?php
//Open a new connection to the MySQL server
$mysqli = new mysqli('host','username','password','database_name');

//Output any connection error
if ($mysqli->connect_error) {
    die('Error : ('. $mysqli->connect_errno .') '. $mysqli->connect_error);
}

//get total number of records
$results = $mysqli->query("SELECT COUNT(*) FROM users");
$get_total_rows = $results->fetch_row(); //hold total records in variable

$mysqli->close();
?>

```

SELECT Using Prepared Statements

Another important feature of MySqli is the Prepared Statements, it allows us to write query just once and then it can be executed repeatedly with different parameters. Prepared Statements significantly improves performance on larger table and more complex queries. The queries are parsed separately by the server, making it resilient to malicious code injection.

The code below uses Prepared statement to fetch records from the database. `?`placeholder in the SQL query acts like marker and will be replaced by a parameter, which could be string, integer, double or blob. In our case it's a string *$search_product*.

PHP

```php
$search_product = "PD1001"; //product id

//create a prepared statement
$query = "SELECT id, product_code, product_desc, price FROM
products WHERE product_code=?";
$statement = $mysqli->prepare($query);

//bind parameters for markers, where (s = string, i =
integer, d = double,  b = blob)
$statement->bind_param('s', $search_product);

//execute query
$statement->execute();

//bind result variables
$statement-
>bind_result($id, $product_code, $product_desc, $price);

print '<table border="1">';

//fetch records
while($statement->fetch()) {
    print '<tr>';
    print '<td>'.$id.'</td>';
    print '<td>'.$product_code.'</td>';
    print '<td>'.$product_desc.'</td>';
    print '<td>'.$price.'</td>';
    print '</tr>';

}
print '</table>';
```

```
//close connection
$statement->close();
```

Same query with multiple parameters:

PHP

```
$search_ID = 1;
$search_product = "PD1001";

$query = "SELECT id, product_code, product_desc, price FROM
products WHERE ID=? AND product_code=?";
$statement = $mysqli->prepare($query);
$statement->bind_param('is', $search_ID, $search_product);
$statement->execute();
$statement-
>bind_result($id, $product_code, $product_desc, $price);

print '<table border="1">';
while($statement->fetch()) {
    print '<tr>';
    print '<td>'.$id.'</td>';
    print '<td>'.$product_code.'</td>';
    print '<td>'.$product_desc.'</td>';
    print '<td>'.$price.'</td>';
    print '</tr>';

}
print '</table>';

//close connection
$statement->close();
```

INSERT a Record

Following MySQLi statement inserts a new row in the table.

PHP

```
<?php
//values to be inserted in database table
$product_code = '"'.$mysqli-
>real_escape_string('P1234').'"';
$product_name = '"'.$mysqli->real_escape_string('42 inch
TV').'"';
$product_price = '"'.$mysqli->real_escape_string('600').'"';

//MySqli Insert Query
```

```php
$insert_row = $mysqli->query("INSERT INTO products
(product_code, product_name, price)
VALUES($product_code, $product_name, $product_price)");

if($insert_row){
    print 'Success! ID of last inserted record is :
' .$mysqli->insert_id .'<br />';
}else{
    die('Error : ('. $mysqli->errno .') '. $mysqli->error);
}

?>
```

INSERT a Record (Prepared Statement)

Snippet below inserts same values using Prepared Statement. As discussed earlier the Prepared statements are very effective against SQL injection, you should always use prepared statement in any given situations.

PHP

```php


//values to be inserted in database table
$product_code = 'P1234';
$product_name = '42 inch TV';
$product_price = '600';

$query = "INSERT INTO products (product_code, product_name,
price) VALUES(?, ?, ?)";
$statement = $mysqli->prepare($query);

//bind parameters for markers, where (s = string, i =
integer, d = double,  b = blob)
$statement-
>bind_param('sss', $product_code, $product_name, $product_pr
ice);

if($statement->execute()){
    print 'Success! ID of last inserted record is :
' .$statement->insert_id .'<br />';
}else{
    die('Error : ('. $mysqli->errno .') '. $mysqli->error);
}
$statement->close();

```

Insert Multiple Records

To insert multiple rows at once, include multiple lists of column values, each enclosed within parentheses and separated by commas. Sometimes you want to know how many records have been inserted, updated or deleted, you can use `mysqli_affected_rows` for that occasion.

```php
//product 1
$product_code1 = '"'.$mysqli->real_escape_string('P1').'"';
$product_name1 = '"'.$mysqli->real_escape_string('Google Nexus').'"';
$product_price1 = '"'.$mysqli->real_escape_string('149').'"';

//product 2
$product_code2 = '"'.$mysqli->real_escape_string('P2').'"';
$product_name2 = '"'.$mysqli->real_escape_string('Apple iPad 2').'"';
$product_price2 = '"'.$mysqli->real_escape_string('217').'"';

//product 3
$product_code3 = '"'.$mysqli->real_escape_string('P3').'"';
$product_name3 = '"'.$mysqli->real_escape_string('Samsung Galaxy Note').'"';
$product_price3 = '"'.$mysqli->real_escape_string('259').'"';

//Insert multiple rows
$insert = $mysqli->query("INSERT INTO products(product_code, product_name, price) VALUES
($product_code1, $product_name1, $product_price1),
($product_code2, $product_name2, $product_price2),
($product_code3, $product_name3, $product_price3)");

if($insert){
    //return total inserted records using mysqli_affected_rows
    print 'Success! Total ' .$mysqli->affected_rows .' rows added.<br />';
}else{
    die('Error : ('. $mysqli->errno .') '. $mysqli->error);
}
```

Update/Delete a Records

Updating and deleting records works similar way, just change to query string to MySql Update or delete.

```php
//MySqli Update Query
$results = $mysqli->query("UPDATE products SET
product_name='52 inch TV', product_code='323343' WHERE
ID=24");

//MySqli Delete Query
//$results = $mysqli->query("DELETE FROM products WHERE
ID=24");

if($results){
    print 'Success! record updated / deleted';
}else{
    print 'Error : ('. $mysqli->errno .') '. $mysqli->error;
}
```

Update using Prepared Statement

Here's how you update record using Prepared Statement.

```php
$product_name = '52 inch TV';
$product_code = '9879798';
$find_id = 1;

$statement = $mysqli->prepare("UPDATE products SET
product_name=?, product_code=? WHERE ID=?");

//bind parameters for markers, where (s = string, i =
integer, d = double,  b = blob)
$statement-
>bind_param('ssi', $product_name, $product_code, $find_id);
$results =  $statement->execute();
if($results){
    print 'Success! record updated';
}else{
    print 'Error : ('. $mysqli->errno .') '. $mysqli->error;
}
```

Delete Old Records

Delete all records that is 1 day old, or specify X days records you want to delete.

PHP

```php
//MySqli Delete Query
$results = $mysqli->query("DELETE FROM products WHERE
added_timestamp < (NOW() - INTERVAL 1 DAY)");

if($results){
    print 'Success! deleted one day old records';
}else{
    print 'Error : (' . $mysqli->errno . ') ' . $mysqli->error;
}
```

Insert or Update if record already exists

People often ask how to INSERT a new row or UPDATE if record already exists. The answer is simple, using `ON DUPLICATE KEY UPDATE` Syntax in MySql query. This clause simply looks for duplicate values in UNIQUE or PRIMARY key and performs INSERT or UPDATE statement. It is pretty useful when you want to INSERT a new record or UPDATE if record already exist.

PHP

```php
$id = 0;
$product_code = "P1234";
$product_name = "42 inch TV";
$product_desc = "42 inch TV is good enough for movies";
$price = "1000";

//MySql query using ON DUPLICATE KEY UPDATE
$query = "INSERT INTO products (id, product_code,
product_name, product_desc, price)
            VALUES (?, ?, ?, ?, ?) ON DUPLICATE KEY UPDATE
product_code=?, product_name=?, product_desc=?, price=?";

$statement = $mysqli->prepare($query); // Prepare query for
execution

//bind parameters for markers, where (s = string, i =
integer, d = double,  b = blob)
$statement-
>bind_param('dsssdsssd', $id, $product_code, $product_name,
$product_desc, $price, //insert vars

 $product_code, $product_name, $product_desc, $price); //upd
ate vars

$statement->execute();
```

Just pass the UNIQUE or PRIMARY value such as ID of a row to perform update, or *0* to insert new row.

Conclusion

MySqli is clearly a winner over the regular MySql extension in PHP, and the implementation is also not that different. I just hope this article will help you migrate/build your projects in future. I have also included some example files in download section, download it for the reference and happy coding!