PHP and MySQL CRUD Tutorial for Beginners – Step By Step Guide!

Last Update: July 4, 2016 • Date Posted: December 31, 2011 • by Mike Dalisay

Do you want a simple reference on PHP CRUD Tutorial, or how to do CRUD operations with PHP and MySQL?

Or, are you a beginner in this kind of PHP web programming? If you say YES, this PHP CRUD tutorial is for you!

For those who are not yet familiar, CRUD is an acronym. It stands for **C**reate, **R**ead, **U**pdate and **D**elete database records.

Coding CRUD with PHP and MySQL is one of the basics. PHP web programmers must be able to code it with less effort. We can perform this task using any of the three PHP Database extensions:

1. Using the MySQL extension.
2. Using the MySQLi extension.
3. Using the PDO extension.

PHP 5.5 deprecated the MySQL extension. It is not recommended to use these days.

If you are programming with PHP, you'll have to use either MySQLi (i means improved) or PDO extension.

With that in mind, we will use the PDO extension in this simple PHP CRUD tutorial. It is the most recent way of programming these CRUD operations.

We will cover the following topics or contents:

1.0 PHP CRUD TUTORIAL PROGRAM OUTPUT

We have three LEVELS of source code output. But WHY? Because I believe in "Learning Progression" to ensure efficient learning.

Accoding to Dr. W. James Popham, an Emeritus Professor in the UCLA Graduate School of Education and Information Studies:

"Learning Progression is a sequenced set of skills and knowledge we believe students must learn on the way to mastering a more distant curricular outcome."

You can read more about the subject here, here and here.

So if you really want to learn, and serious about learning from this programming tutorial: I highly recommend studying the LEVEL 1 source code first, then the LEVEL 2, then the LEVEL 3 source code.

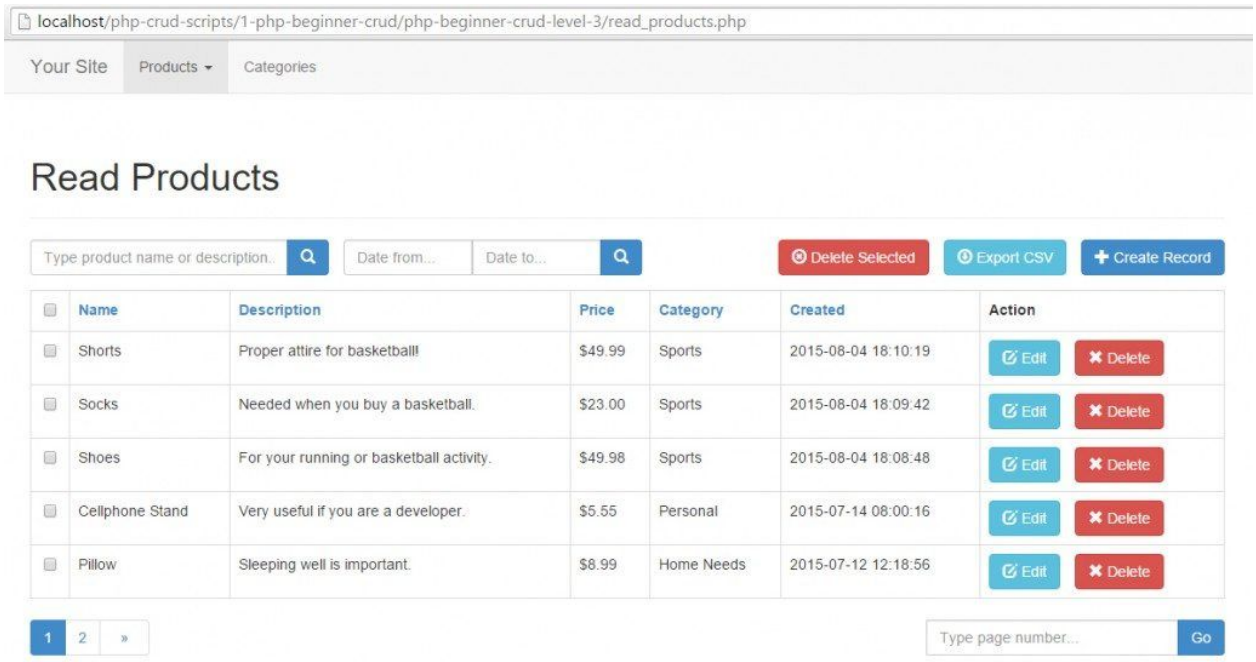
1.1 LEVEL 1 Source Code Output

VIDEO: https://www.youtube.com/watch?v=YbZOfPqsE70

1.2 LEVEL 2 Source Code Output

VIDEO: https://www.youtube.com/watch?v=0qqs7rH5M7k

1.3 LEVEL 3 Source Code Output



You can see the complete list of features on section 9.0 below.

For now, let us continue to learn how the LEVEL 1 source code was made.

2.0 PROJECT FILE STRUCTURE

Our PHP CRUD tutorial will contain the following main files.

- dev/**products.sql** – contains the database table structure and sample data used in this project. Once you created your database in PhpMyAdmin, you can import this file.

- **libs/** – is where our Bootstrap CSS framework and jQuery library is located.
- config/**database.php** – used for database connection and configuration.
- **create.php** – used for creating a new record. It contains an HTML form where the user can enter details for a new record.
- **read.php** – used for reading records from the database. It uses an HTML table to display the data retrieved from the MySQL database.
- **read_one.php** – used for reading one or single record from database. It uses an HTML table to display the data retrieved from the MySQL database.
- **update.php** – used for updating a record. It uses an HTML form which will be filled out with data based on the given "id" parameter.
- **delete.php** – used for deleting a record. It accepts an "id" parameter and deletes the record with it. Once it execute the delete query, it will redirect the user to the read.php page.

3.0 PREPARE THE DATABASE

3.1 Create the Database

On your PhpMyAdmin, create a database named "1phpbeginnercrudlevel1".

If you're not sure how to do it, please take a look at the following example. Follow only the "create database" part.

VIDEO: https://www.youtube.com/watch?v=_ULMwIO3Gt0

3.2 Create the Database Table

Next, run the following SQL code. This is to create our "products" database table. If you're not sure how to do this, take a look at this resource.

```
--
-- Table structure for table `products`
--

CREATE TABLE IF NOT EXISTS `products` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(128) NOT NULL,
  `description` text NOT NULL,
  `price` double NOT NULL,
  `created` datetime NOT NULL,
  `modified` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;
```

3.3 Dump Sample Data On The Table

Again, run the following SQL code on your PhpMyAdmin. This will insert the sample data or record on our ‘products’ database table.

```
--
-- Dumping data for table `products`
--

INSERT INTO `products` (`id`, `name`, `description`, `price`,
`created`, `modified`) VALUES
(1, 'Basketball', 'A ball used in the NBA.', 49.99, '2015-08-02
12:04:03', '2015-08-06 06:59:18'),
```

```
(3, 'Gatorade', 'This is a very good drink for athletes.', 1.99,
'2015-08-02 12:14:29', '2015-08-06 06:59:18'),
(4, 'Eye Glasses', 'It will make you read better.', 6, '2015-08-02
12:15:04', '2015-08-06 06:59:18'),
(5, 'Trash Can', 'It will help you maintain cleanliness.', 3.95,
'2015-08-02 12:16:08', '2015-08-06 06:59:18'),
(6, 'Mouse', 'Very useful if you love your computer.', 11.35,
'2015-08-02 12:17:58', '2015-08-06 06:59:18'),
(7, 'Earphone', 'You need this one if you love music.', 7,
'2015-08-02 12:18:21', '2015-08-06 06:59:18'),
(8, 'Pillow', 'Sleeping well is important.', 8.99, '2015-08-02
12:18:56', '2015-08-06 06:59:18');
```

As you may have noticed, steps 1 and 2 are both SQL queries. Yes, they can run at the same time. But I wanted it to be on separate steps to emphasize those SQL queries' purpose.

3.4 Create Database Connection PHP File

Create database.php file and put the following code inside it. It answers the question: how to connect to MySQL database with PDO?

```
<?php
// used to connect to the database
$host = "localhost";
$db_name = "1phpbeginnercrudlevel1";
$username = "root";
$password = "";
```

```
try {
    $con = new PDO("mysql:host={$host};dbname={$db_name}", $username,
$password);
}

// show error
catch(PDOException $exception){
    echo "Connection error: " . $exception->getMessage();
}
?>
```

4.0 CREATE RECORD IN PHP

4.1 Basic HTML Code For create.php

Create the create.php file. We will use it to create a new record to the database. Put the code following code inside the create.php file.

We use Bootstrap user interface for this project. Make sure you downloaded a copy and put it inside the “libs” folder.

If you are not familiar with Bootstrap, please learn our Bootstrap Tutorial for Beginners real quick.

```
<!DOCTYPE HTML>
<html>
<head>
    <title>PDO - Create a Record - PHP CRUD Tutorial</title>
```

```
    <!-- Bootstrap -->
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="libs/bootstrap-3.3.6/css/bootstrap.min.css" />

    <!-- HTML5 Shiv and Respond.js IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
    <!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></scri
pt>
    <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></s
cript>
    <![endif]-->

</head>
<body>

    <!-- container -->
    <div class="container">

        <div class="page-header">
            <h1>Create Product</h1>
        </div>
```

```
    <!-- dynamic content will be here -->

    </div> <!-- end .container -->

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="libs/jquery-3.0.0.min.js"></script>

<!-- Include all compiled plugins (below), or include individual
files as needed -->
<script src="libs/bootstrap-3.3.6/js/bootstrap.min.js"></script>

</body>
</html>
```

4.2 HTML Form To Input New Record Data

Now we are going to start answering the question: how to create a record with PDO?

The code below will create an HTML form with input fields that matches the fields in the database. Put it inside the “container” div of create.php of section 4.1 above.

```
<!-- html form here where the product information will be entered -->
<form action='create.php' method='post'>
    <table class='table table-hover table-responsive table-bordered'>
        <tr>
            <td>Name</td>
            <td><input type='text' name='name' class='form-control'
/></td>
        </tr>
```

```
        <tr>
            <td>Description</td>
            <td><textarea name='description'
class='form-control'></textarea></td>
        </tr>
        <tr>
            <td>Price</td>
            <td><input type='text' name='price' class='form-control'
/></td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input type='submit' value='Save' class='btn
btn-primary' />
                <a href='read.php' class='btn btn-danger'>Back to
read products</a>
            </td>
        </tr>
    </table>
</form>
```

4.3 Code To Create A New Record

We are still working in the create.php file. Once the user filled out the form and clicked the save button in section 4.2, the code below will save it to the MySQL database. Put it above the “form” tag of section 4.2 above.

```
<?php
```

```
if($_POST){

    // include database connection
    include 'config/database.php';

    try{

        // insert query
        $query = "INSERT INTO products SET name=:name,
description=:description, price=:price, created=:created";

        // prepare query for execution
        $stmt = $con->prepare($query);

        // posted values
        $name=htmlspecialchars(strip_tags($_POST['name']));

$description=htmlspecialchars(strip_tags($_POST['description']));
        $price=htmlspecialchars(strip_tags($_POST['price']));

        // bind the parameters
        $stmt->bindParam(':name', $name);
        $stmt->bindParam(':description', $description);
        $stmt->bindParam(':price', $price);

        // specify when this record was inserted to the database
        $created=date('Y-m-d H:i:s');
```

```
        $stmt->bindParam(':created', $created);

        // Execute the query
        if($stmt->execute()){
            echo "<div class='alert alert-success'>Record was
saved.</div>";
        }else{
            echo "<div class='alert alert-danger'>Unable to save
record.</div>";
        }

    }

    // show error
    catch(PDOException $exception){
        die('ERROR: ' . $exception->getMessage());
    }
}
?>
```

5.0 READ RECORDS IN PHP

5.1 Basic HTML Code For read.php

Create the read.php file. We prepare this to read records from the database. It answers the question: how to read records with PDO?

Put the following code inside the read.php file.

```
<!DOCTYPE HTML>
<html>
<head>
    <title>PDO - Read Records - PHP CRUD Tutorial</title>

    <!-- Bootstrap -->
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="libs/bootstrap-3.3.6/css/bootstrap.min.css" />

    <!-- HTML5 Shiv and Respond.js IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
    <!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></scri
pt>
    <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></s
cript>
    <![endif]-->

    <!-- custom css -->
    <style>
    .m-r-1em{ margin-right:1em; }
    .m-b-1em{ margin-bottom:1em; }
```

```
    .m-l-1em{ margin-left:1em; }
    </style>

</head>
<body>

    <!-- container -->
    <div class="container">

        <div class="page-header">
            <h1>Read Products</h1>
        </div>

    <!-- dynamic content will be here -->

    </div> <!-- end .container -->

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="libs/jquery-3.0.0.min.js"></script>

<!-- Include all compiled plugins (below), or include individual
files as needed -->
<script src="libs/bootstrap-3.3.6/js/bootstrap.min.js"></script>

</body>
</html>
```

5.2 Read Records From The Database

This time we will read records from the database. Put the following code inside the "container" div tags in section 5.1 above.

```
<?php
// include database connection
include 'config/database.php';

// select all data
$query = "SELECT id, name, description, price FROM products ORDER BY
id DESC";
$stmt = $con->prepare($query);
$stmt->execute();

// this is how to get number of rows returned
$num = $stmt->rowCount();

// link to create record form
echo "<a href='create.php' class='btn btn-primary m-b-1em'>Create New
Product</a>";

//check if more than 0 record found
if($num>0){

    echo "<table class='table table-hover table-responsive
table-bordered'>";//start table
```

```
        //creating our table heading
        echo "<tr>";
            echo "<th>ID</th>";
            echo "<th>Name</th>";
            echo "<th>Description</th>";
            echo "<th>Price</th>";
            echo "<th>Action</th>";
        echo "</tr>";

        // retrieve our table contents
        // fetch() is faster than fetchAll()
        //
```
http://stackoverflow.com/questions/2770630/pdofetchall-vs-pdofetch-in-a-loop
```
        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            // extract row
            // this will make $row['firstname'] to
            // just $firstname only
            extract($row);

            // creating new table row per record
            echo "<tr>";
                echo "<td>{$id}</td>";
                echo "<td>{$name}</td>";
                echo "<td>{$description}</td>";
                echo "<td>&#36;{$price}</td>";
                echo "<td>";
```

```
                    // read one record
                    echo "<a href='read_one.php?id={$id}' class='btn
btn-info m-r-1em'>Read</a>";

                    // we will use this links on next part of this
post
                    echo "<a href='update.php?id={$id}' class='btn
btn-primary m-r-1em'>Edit</a>";

                    // we will use this links on next part of this
post
                    echo "<a href='#' onclick='delete_user({$id});'
class='btn btn-danger'>Delete</a>";
                echo "</td>";
            echo "</tr>";
        }

    // end table
    echo "</table>";

}

// if no records found
else{
    echo "<div>No records found.</div>";
}
?>
```

The code above shows:

- The inclusion of database.php file from section 3.4
- The SELECT SQL query.
- The HTML table where the retrieved data will be put.

6.0 READ ONE RECORD IN PHP

6.1 Basic HTML Code For read_one.php

Create a PHP file and name it read_one.php – this is where we will read and display the details of a single database record. Put the following basic HTML code.

```
<!DOCTYPE HTML>
<html>
<head>
    <title>PDO - Read One Record - PHP CRUD Tutorial</title>

    <!-- Bootstrap -->
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="libs/bootstrap-3.3.6/css/bootstrap.min.css" />

    <!-- HTML5 Shiv and Respond.js IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
    <!--[if lt IE 9]>
```

```
    <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></scri
pt>
    <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></s
cript>
    <![endif]-->

</head>
<body>



    <!-- container -->
    <div class="container">

        <div class="page-header">
            <h1>Read Product</h1>
        </div>

        <!-- dynamic content will be here -->

    </div> <!-- end .container -->

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="libs/jquery-3.0.0.min.js"></script>
```

```
<!-- Include all compiled plugins (below), or include individual
files as needed -->
<script src="libs/bootstrap-3.3.6/js/bootstrap.min.js"></script>

</body>
</html>
```

6.2 Read One Record From The Database

The following code is how we retrieve a single database record. Put it inside the "container" div tags.

```
<?php
// get passed parameter value, in this case, the record ID
// isset() is a PHP function used to verify if a value is there or
not
$id=isset($_GET['id']) ? $_GET['id'] : die('ERROR: Record ID not
found.');

//include database connection
include 'config/database.php';

// read current record's data
try {
    // prepare select query
    $query = "SELECT id, name, description, price FROM products WHERE
id = ? LIMIT 0,1";
    $stmt = $con->prepare( $query );
```

```
    // this is the first question mark
    $stmt->bindParam(1, $id);

    // execute our query
    $stmt->execute();

    // store retrieved row to a variable
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    // values to fill up our form
    $name = $row['name'];
    $description = $row['description'];
    $price = $row['price'];
}

// show error
catch(PDOException $exception){
    die('ERROR: ' . $exception->getMessage());
}
?>
```

6.3 Display Record Details

The following HTML table will hold and display the details of a single database record.

Put the following code after the code in section 6.2 above.

```
<!--we have our html table here where new user information will be
displayed-->
<table class='table table-hover table-responsive table-bordered'>
```

```
    <tr>
        <td>Name</td>
        <td><?php echo htmlspecialchars($name, ENT_QUOTES);  ?></td>
    </tr>
    <tr>
        <td>Description</td>
        <td><?php echo htmlspecialchars($description, ENT_QUOTES);
?></td>
    </tr>
    <tr>
        <td>Price</td>
        <td><?php echo htmlspecialchars($price, ENT_QUOTES);  ?></td>
    </tr>
    <tr>
        <td></td>
        <td>
            <a href='read.php' class='btn btn-danger'>Back to read
products</a>
        </td>
    </tr>
</table>
```

7.0 UPDATE RECORD IN PHP

7.1 Basic HTML Code For udpate.php

Create the update.php file. We are preparing to update a selected record from the database. This will answer the question: how to update a record with PDO?

Put the following code inside the new update.php file.

```
<!DOCTYPE HTML>
<html>
<head>
    <title>PDO - Update a Record - PHP CRUD Tutorial</title>

    <!-- Bootstrap -->
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="libs/bootstrap-3.3.6/css/bootstrap.min.css" />

    <!-- HTML5 Shiv and Respond.js IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
    <!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></scri
pt>
    <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></s
cript>
    <![endif]-->

</head>
<body>
```

```
    <!-- container -->
    <div class="container">

        <div class="page-header">
            <h1>Update Product</h1>
        </div>

    <!-- dynamic content will be here -->

    </div> <!-- end .container -->

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="libs/jquery-3.0.0.min.js"></script>

<!-- Include all compiled plugins (below), or include individual
files as needed -->
<script src="libs/bootstrap-3.3.6/js/bootstrap.min.js"></script>

</body>
</html>
```

7.2 Read A Record By ID Parameter

We have to get the record ID and store it in the $id variable. We access the $_GET[‘id’] variable to do it.

What we are trying to do here is to get the related data based on the given record ID. This is a way to auto-fill the HTML form (on section 7.3) with existing row data.

Put the following code inside the “container” div tag of update.php in section 7.1 above.

```
<?php
// get passed parameter value, in this case, the record ID
// isset() is a PHP function used to verify if a value is there or
not
$id=isset($_GET['id']) ? $_GET['id'] : die('ERROR: Record ID not
found.');

// read current record's data
try {
    // prepare select query
    $query = "SELECT id, name, description, price FROM products WHERE
id = ? LIMIT 0,1";
    $stmt = $con->prepare( $query );

    // this is the first question mark
    $stmt->bindParam(1, $id);

    // execute our query
    $stmt->execute();

    // store retrieved row to a variable
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    // values to fill up our form
    $name = $row['name'];
```

```
    $description = $row['description'];
    $price = $row['price'];
}

// show error
catch(PDOException $exception){
    die('ERROR: ' . $exception->getMessage());
}
?>
```

7.3 HTML Form To Update A Record

The following code will be under the code on section 7.2. This is no ordinary HTML form like the section 4.2 above. This form shows data retrieved from section 7.2 where we read a single record based on given ID parameter.

```
<!--we have our html form here where new user information will be
entered-->
<form action='update.php?id=<?php echo htmlspecialchars($id); ?>'
method='post' border='0'>
    <table class='table table-hover table-responsive table-bordered'>
        <tr>
            <td>Name</td>
            <td><input type='text' name='name' value="<?php echo
htmlspecialchars($name, ENT_QUOTES);  ?>" class='form-control'
/></td>
        </tr>
        <tr>
            <td>Description</td>
```

```
            <td><textarea name='description'
class='form-control'><?php echo htmlspecialchars($description,
ENT_QUOTES);  ?></textarea></td>
        </tr>
        <tr>
            <td>Price</td>
            <td><input type='text' name='price' value="<?php echo
htmlspecialchars($price, ENT_QUOTES);  ?>" class='form-control'
/></td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input type='submit' value='Save Changes' class='btn
btn-primary' />
                <a href='read.php' class='btn btn-danger'>Back to
read products</a>
            </td>
        </tr>
    </table>
</form>
```

7.4 Code To Update The Record

The following code will save the changes in the database.

That is if the user change some value on the form and hit the “Save Changes” button.

Put the following code before section 7.3’s code.

```
<?php
```

```
// get passed parameter value, in this case, the record ID
// isset() is a PHP function used to verify if a value is there or
not
$id=isset($_GET['id']) ? $_GET['id'] : die('ERROR: Record ID not
found.');

//include database connection
include 'config/database.php';

// check if form was submitted
if($_POST){

    try{

        // write update query
        // in this case, it seemed like we have so many fields to
pass and
        // it is better to label them and not use question marks
        $query = "UPDATE products
                    SET name=:name, description=:description,
price=:price
                    WHERE id = :id";

        // prepare query for excecution
        $stmt = $con->prepare($query);

        // posted values
```

```
        $name=htmlspecialchars(strip_tags($_POST['name']));

$description=htmlspecialchars(strip_tags($_POST['description']));
        $price=htmlspecialchars(strip_tags($_POST['price']));

        // bind the parameters
        $stmt->bindParam(':name', $name);
        $stmt->bindParam(':description', $description);
        $stmt->bindParam(':price', $price);
        $stmt->bindParam(':id', $id);

        // Execute the query
        if($stmt->execute()){
            echo "<div class='alert alert-success'>Record was
updated.</div>";
        }else{
            echo "<div class='alert alert-danger'>Unable to update
record. Please try again.</div>";
        }

    }

    // show errors
    catch(PDOException $exception){
        die('ERROR: ' . $exception->getMessage());
    }
}
```

```
?>
```

8.0 DELETE RECORD IN PHP

8.1 Tell The User If Record Was Deleted

Put the following code right after the code ***include 'config/database.php';*** in read.php file.

This will tell the user if there is a deleted record after clicking the delete button and "ok" in the pop up.

```
$action = isset($_GET['action']) ? $_GET['action'] : "";

// if it was redirected from delete.php
if($action=='deleted'){
    echo "<div class='alert alert-success'>Record was
deleted.</div>";
}
```

8.2 JavaScript To Verify Record Deletion

The user clicks on the "Delete" button in read.php. Next, he will verify the deletion by clicking "OK" on the pop up.

That user activity will execute the following JavaScript code. Put it before the end "body" tag in read.php file.

```
<script type='text/javascript'>
function delete_user( id ){

    var answer = confirm('Are you sure?');
```

```
    if (answer){
        // if user clicked ok,
        // pass the id to delete.php and execute the delete query
        window.location = 'delete.php?id=' + id;
    }
}
</script>
```

8.3 Delete Record From The Database

The code below will delete a record from the database using the given ID parameter.

This answers the question: how to delete a record with PDO? Create delete.php file and put the following code inside it.

```
<?php
// include database connection
include 'config/database.php';

try {

    // get record ID
    // isset() is a PHP function used to verify if a value is there or not
    $id=isset($_GET['id']) ? $_GET['id'] : die('ERROR: Record ID not found.');

    // delete query
    $query = "DELETE FROM products WHERE id = ?";
    $stmt = $con->prepare($query);
```

```
    $stmt->bindParam(1, $id);

    if($stmt->execute()){
        // redirect to read records page and
        // tell the user record was deleted
        header('Location: read.php?action=deleted');
    }else{
        die('Unable to delete record.');
    }
}

// show error
catch(PDOException $exception){
    die('ERROR: ' . $exception->getMessage());
}
?>
```

9.0 DOWNLOAD SOURCE CODES

You can get the source code by following the whole, well detailed PHP CRUD tutorial above. But isn't it more convenient if you can just download the complete source code we used, and play around it?

To download, see section 9.0 of this link:
https://www.codeofaninja.com/2011/12/php-and-mysql-crud-tutorial.html

10.0 ONLINE RESOURCES

- What is the difference between MySQL, MySQLi and PDO extensions?
- MySQLi or PDO – what are the pros and cons?
- PDO vs. MySQLi: Which Should You Use?
- MySQLi vs. PDO Benchmarks

11.0 WHAT'S NEXT?

After learning how to do CRUD with PHP and MySQL, you have to be one step higher. You can do this by adding some AJAX functionalities to it.

What is AJAX? It stands for "Asynchronous JavaScript and XML".

I'll try to explain it to you in the simplest way: Using AJAX will prevent re-loading the whole page for every button click you make.

As a result, it makes the user experience better. Your web app will be faster.

We have created a tutorial for it, see: CRUD with PHP and AJAX – Step by Step Guide

12.0 RELATED SOURCE CODES

The following related source code tutorials can be very useful to further improve your skills.

See section 12.0 of this link:
https://www.codeofaninja.com/2011/12/php-and-mysql-crud-tutorial.html

13.0 SOME NOTES

#1 Found An Issue?

If you found a problem with this code, we can solve it faster via Email or FB message, please send me a message via email **mike@codeofaninja.com**, or via our official Facebook page!

Please be more detailed about your issue. Best if you can provide an error message and your test or page URL. Thanks!

Please feel free to comment if you have any questions, suggestions, found something wrong or want to contribute to this code.

#2 Become a true Ninja!

If you haven't subscribed yet please, please do! It's free, please fill out the form in the following link: https://www.codeofaninja.com/subscribe

#3 Thank You!

Please share this post if you think this is a useful PHP CRUD Tutorial. Thanks for reading!