CODEOFANINJA

Home → PHP

PHP OOP CRUD Tutorial – Step By Step Guide!

Last update: June 18, 2020 • Date posted: June 5, 2014

385 COMMENTS • Posted by MIKE DALISAY

Previously, we learned how to create or insert, read, update, and delete database records with our PHP and MySQL CRUD tutorial for beginners [https://www.codeofaninja.com/2011/12/php-and-mysql-crud-tutorial.html] . This time, we will learn object-oriented programming with PHP & MySQL.

This post will include the following contents:

If you have a positive feedback about our work, please let us know. If there's a section in this tutorial that is confusing or hard to understand, we consider it as a problem. Please let us know as well.

Write your positive feedback or detailed description of the problem in the comments section below. Before you write a comment, please read this guide [https://www.codeofaninja.com/how-to-write-a-great-comment-on-codeofaninja-com] and our code of conduct [https://www.codeofaninja.com/code-of-conduct] . Thank you!

1.0 OVERVIEW

Our tutorial for today is about creating a simple database application. We can achieve it with the help of this PHP OOP CRUD tutorial. You can use this knowledge in your current or future projects.

We use Bootstrap so that our application will have a decent UI. If you're not yet familiar what Bootstrap is, and you want to learn how to use it in few steps, I highly recommend following our Bootstrap tutorial [https://www.codeofaninja.com/2014/05/bootstrap-tutorial-beginners-step-step.html] first.

There are so many PHP object-oriented programming tutorials on the web today, they have different examples and implementations. Some might be completely correct, some maybe not.

I'm writing this tutorial with a clear goal: to give the best PHP OOP CRUD tutorial for beginners. I welcome your comments and suggestions to help me achieve this.

We want to learn the correct PHP OOP implementation. There are PHP

Those things are one step higher. For now, we will learn object oriented programming with PHP & MySQL. Working with a PHP framework should be easy after following this tutorial.

2.0 PROGRAM OUTPUT - PHP OOP CRUD TUTORIAL

We usually have three LEVELS of source code output. But WHY? Because I believe in "Learning Progression" to ensure efficient learning. Learn more [https://www.codeofaninja.com/2016/07/learning-progression.html]

Below are some screenshots of our script's output. You can click an image to view the larger version of it. Use the left and right arrow to navigate through the screenshots.

Please note that the following images are just output previews. New features might be added already the time you are reading this.

2.1 LEVEL 1 Source Code Output

[espro-slider id=6953]

2.2 LEVEL 2 Source Code Output

[espro-slider id=6960]

2.3 LEVEL 3 Source Code Output

[espro-slider id=6978]

Downloading our source codes is your huge advantage as well. For now, let's proceed to the step by step tutorial of our LEVEL 1 source code. Enjoy!

3.0 DATABASE TABLE STRUCTURE

The files products.sql and categories.sql are also included in the code download, located at the **README** folder.

3.1 Create a database

- Open your PhpMyAdmin (http://localhost/phpmyadmin [http://Run the following SQL statement using your PhpMyadmin.])
- Create a new database.
- Put php_oop_crud_level_1 as database name.
- Click "Create" button.

3.2 Create products table

In this section, we will create the "products" table (using PhpMyAdmin) on the database we just created.

Here's how to run an SQL statement using PhpMyAdmin.

- Click php_oop_crud_level_1 database.
- Click "SQL" tab.
- Copy the SQL statement below and paste it in the text area.
- Click the "Go" button.

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

```
  `price` int(11) NOT NULL,
  `category_id` int(11) NOT NULL,
  `created` datetime NOT NULL,
  `modified` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=38 ;
```

3.3 Insert products sample data

We have to put some database records.

Run the following SQL statement using your PhpMyAdmin.

```
-- Dumping data for table `products`
INSERT INTO `products` (`id`, `name`, `description`, `price`,
(1, 'LG P880 4X HD', 'My first awesome phone!', 336, 3, '2014-
(2, 'Google Nexus 4', 'The most awesome phone of 2013!', 299,
(3, 'Samsung Galaxy S4', 'How about no?', 600, 3, '2014-06-01
(6, 'Bench Shirt', 'The best shirt!', 29, 1, '2014-06-01 01:12
(7, 'Lenovo Laptop', 'My business partner.', 399, 2, '2014-06-
(8, 'Samsung Galaxy Tab 10.1', 'Good tablet.', 259, 2, '2014-0
(9, 'Spalding Watch', 'My sports watch.', 199, 1, '2014-06-01
(10, 'Sony Smart Watch', 'The coolest smart watch!', 300, 2, '
(11, 'Huawei Y300', 'For testing purposes.', 100, 2, '2014-06-
(12, 'Abercrombie Lake Arnold Shirt', 'Perfect as gift!', 60,
(13, 'Abercrombie Allen Brook Shirt', 'Cool red shirt!', 70, 1
(25, 'Abercrombie Allen Anew Shirt', 'Awesome new shirt!', 999
(26, 'Another product', 'Awesome product!', 555, 2, '2014-11-2
(27, 'Bag', 'Awesome bag for you!', 999, 1, '2014-12-04 21:11:
(28, 'Wallet', 'You can absolutely use this one!', 799, 1, '20
(30, 'Wal-mart Shirt', '', 555, 2, '2014-12-13 00:52:29', '201
(31, 'Amanda Waller Shirt', 'New awesome shirt!', 333, 1, '201
(32, 'Washing Machine Model PTRR', 'Some new product.', 999, 1
```

3.4 Create categories table

Categories table are used to store product categories.

Run the following SQL statement using your PhpMyAdmin.

```
  `created` datetime NOT NULL,
  `modified` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=4 ;
```

3.5 Insert categories sample data

We are going to have "Fashion", "Electronics" and "Motors" as categories in our project. I got those three category ideas from eBay, haha!

Run the following SQL statement using your PhpMyAdmin.

```
-- Dumping data for table `categories`
INSERT INTO `categories` (`id`, `name`, `created`, `modified`)
(1, 'Fashion', '2014-06-01 00:35:07', '2014-05-30 17:34:33'),
(2, 'Electronics', '2014-06-01 00:35:07', '2014-05-30 17:34:33
(3, 'Motors', '2014-06-01 00:35:07', '2014-05-30 17:34:54');
```

3.6 Output

In this section, we were able to set up our database using PhpMyAdmin. It should look like the image below.

We don't have a PHP program output yet. Let's continue on the next section to achieve more output.

4.0 CREATE THE LAYOUT FILES

To reduce some code mess, we will create the layout files with the codes and assets it needs.

4.1 Create header layout file

This `layout_header.php` file will be included at the beginning of the PHP files that will need it. This way, we won't have to write the same header codes every time.

We use the Bootstrap framework [https://getbootstrap.com/] to make our project look good. If you're not yet familiar with you, please learn our Bootstrap tutorial here [https://www.codeofaninja.com/2014/05/bootstrap-tutorial-beginners-step-step.html] first.

Bootstrap CSS asset will be included inside the head tags.

- Create `php-oop-crud-level-1` folder and open it.
- Create `layout_header.php` file.
- Place the following code.

```
<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <!-- Latest compiled and minified Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.c

    <!-- our custom CSS -->
    <link rel="stylesheet" href="libs/css/custom.css" />

</head>
<body>

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

        <?php
        // show page header
        echo "<div class='page-header'>
                <h1>{$page_title}</h1>
            </div>";
        ?>
```

4.2 Create footer layout file

This `layout_footer.php` will be included at the end of each PHP files that needs it. This way, we won't have to write the same footer codes every time.

The assets used in this file are:

- jQuery [https://jquery.com/] - needed by Bootstrap JavaScript.
- Bootstrap JavaScript [https://getbootstrap.com/docs/3.3/javascript/] - to make cool UI components work.
- BootboxJS [http://bootboxjs.com/getting-started.html] - to show good looking alert or confirm dialog boxes.

Let's go on and create the footer layout file.

- Open `php-oop-crud-level-1` folder.

```
    </div>
    <!-- /container -->

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="https://code.jquery.com/jquery-3.2.1.min.js [http

<!-- Latest compiled and minified Bootstrap JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/j

<!-- bootbox library -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootbox.js

</body>
</html>
```

4.3 Create custom CSS file

This file is used to change any style we want on our web page. It is also used to override the default style given by Bootstrap.

- Open php-oop-crud-level-1 folder.
- Create libs folder.
- Create css folder.
- Create custom.css file.
- Place the following code.

```
.left-margin{
    margin:0 .5em 0 0;
}

.right-button-margin{
    margin: 0 0 1em 0;
    overflow: hidden;
}

/* some changes in bootstrap modal */
.modal-body {
```

```
.modal-footer{
    text-align: center !important;
}
```

4.4 Output

The layout files we created in this section is meant to be used inside another PHP file. If we try to run the layout files alone, we won't get any meaningful output.

If you run layout_header.php, it will look like this on the browser.

The custom.css file will look like this.

The layout_footer.php is blank. Let's continue on the next section to see a more meaningful output.

5.0 CREATING RECORD IN PHP THE OOP WAY

5.1 Create a file: create_product.php

Go back to php-oop-crud-level-1 folder, create a file with a name create_product.php and put the following code inside it.

```
<?php
// set page headers
$page_title = "Create Product";
include_once "layout_header.php";

// contents will be here

// footer
include_once "layout_footer.php";
```

The following code will render a button. Replace the comments // contents will be here of the previous section with the following.

```
echo "<div class='right-button-margin'>
        <a href='index.php' class='btn btn-default pull-right'
    </div>";

?>
<!-- 'create product' html form will be here -->
<?php
```

5.3 Get a Database Connection

We can use it for retrieving categories or saving new product record later. Put the following code before // set page headers comment of create_product.php file.

```
// include database and object files
include_once 'config/database.php';
include_once 'objects/product.php';
include_once 'objects/category.php';

// get database connection
$database = new Database();
$db = $database->getConnection();

// pass connection to objects
$product = new Product($db);
$category = new Category($db);
```

5.4 Create the Database Configuration Class

Getting a database connection will not work without this class. This class file will be included in most PHP files of our PHP OOP CRUD Tutorial.

Create a config folder and inside that folder, create a database.php file. Open that file and put the following code.

```
    // specify your own database credentials
    private $host = "localhost";
    private $db_name = "php_oop_crud_level_1";
    private $username = "root";
    private $password = "";
    public $conn;

    // get the database connection
    public function getConnection(){

        $this->conn = null;

        try{
            $this->conn = new PDO("mysql:host=" . $this->host
        }catch(PDOException $exception){
            echo "Connection error: " . $exception->getMessage
        }

        return $this->conn;
    }
}
?>
```

5.5 Create a Form in create_product.php

The following code will render an HTML form. Open create_product.php file.

Replace <!-- 'create product' html form will be here --> comment with the following code.

```
<!-- PHP post code will be here -->

<!-- HTML form for creating a product -->
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]

    <table class='table table-hover table-responsive table-bor

        <tr>
            <td>Name</td>
            <td><input type='text' name='name' class='form-con
```

```
            <td><input type='text' name='price' class='form-co
        </tr>

        <tr>
            <td>Description</td>
            <td><textarea name='description' class='form-contr
        </tr>

        <tr>
            <td>Category</td>
            <td>
            <!-- categories from database will be here -->
            </td>
        </tr>

        <tr>
            <td></td>
            <td>
                <button type="submit" class="btn btn-primary">
            </td>
        </tr>

    </table>
</form>
```

5.6 Loop Through the Categories Records to show as Drop-down

The following code will retrieve categories and put it in a "select" drop-down.

Replace <!-- categories from database will be here --> comment of the previous section with the following code.

```
<?php
// read the product categories from the database
$stmt = $category->read();

// put them in a select drop-down
echo "<select class='form-control' name='category_id'>";
    echo "<option>Select category...</option>";
```

```
echo "</select>";
?>
```

5.7 Create the Object Class for Categories

Of course, the previous section won't work without the category object class. Create objects folder. Create category.php file. Place the following code.

```
<?php
class Category{

    // database connection and table name
    private $conn;
    private $table_name = "categories";

    // object properties
    public $id;
    public $name;

    public function __construct($db){
        $this->conn = $db;
    }

    // used by select drop-down list
    function read(){
        //select all data
        $query = "SELECT
                    id, name
                FROM
                    " . $this->table_name . "
                ORDER BY
                    name";

        $stmt = $this->conn->prepare( $query );
        $stmt->execute();

        return $stmt;
    }

}
```

It will get the category name instead of showing just an ID. Add the following code inside our category.php, you will see this method used in the next few sections.

```
// used to read category name by its ID
function readName(){

    $query = "SELECT name FROM " . $this->table_name . " WHERE

    $stmt = $this->conn->prepare( $query );
    $stmt->bindParam(1, $this->id);
    $stmt->execute();

    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    $this->name = $row['name'];
}
```

5.9 Code when the Form was Submitted

The user will enter the values in the HTML form and when the create (submit) button was clicked, values will be sent via POST request, the code below will save it in the database.

Open create_product.php file. Replace <!-- PHP post code will be here --> comment with the following code.

```
<?php
// if the form was submitted - PHP OOP CRUD Tutorial
if($_POST){

    // set product property values
    $product->name = $_POST['name'];
    $product->price = $_POST['price'];
    $product->description = $_POST['description'];
    $product->category_id = $_POST['category_id'];

    // create the product
```

```
    // if unable to create the product, tell the user
    else{
        echo "<div class='alert alert-danger'>Unable to create
    }
}
?>
```

5.10 Create the Object Class for Products

The previous section will not work without the product object. Open objects folder. Create product.php file. Open that file and put the following code.

```
<?php
class Product{

    // database connection and table name
    private $conn;
    private $table_name = "products";

    // object properties
    public $id;
    public $name;
    public $price;
    public $description;
    public $category_id;
    public $timestamp;

    public function __construct($db){
        $this->conn = $db;
    }

    // create product
    function create(){

        //write query
        $query = "INSERT INTO
                    " . $this->table_name . "
                SET
                    name=:name, price=:price, description=:des
```

```
        $this->price=htmlspecialchars(strip_tags($this->price)
        $this->description=htmlspecialchars(strip_tags($this->
        $this->category_id=htmlspecialchars(strip_tags($this->

        // to get time-stamp for 'created' field
        $this->timestamp = date('Y-m-d H:i:s');

        // bind values
        $stmt->bindParam(":name", $this->name);
        $stmt->bindParam(":price", $this->price);
        $stmt->bindParam(":description", $this->description);
        $stmt->bindParam(":category_id", $this->category_id);
        $stmt->bindParam(":created", $this->timestamp);

        if($stmt->execute()){
            return true;
        }else{
            return false;
        }

    }
}
?>
```

5.11 Output

Form to create product.

Categories drop down in the form.

When you fill out the form and clicked the "Create" button.

Changes in the database.

6.0 READING AND PAGING RECORD IN PHP THE OOP WAY

In this part of our PHP OOP CRUD tutorial, we will list the records from the database.

6.1 Create File: index.php

Create a new file and name it index.php. This file will show the main page of our web app. Put the following code inside it.

```
<?php
// set page header
$page_title = "Read Products";
```

```
// set page footer
include_once "layout_footer.php";
?>
```

6.2 Add a "Create Product" button

The following code will render a button. When this button was clicked, it will show us a page where we can create a record. Replace the // contents will be here comments in the previous section with the following code.

```
echo "<div class='right-button-margin'>
    <a href='create_product.php' class='btn btn-default pull-r
</div>";
```

6.3 Configure Pagination Variables

Pagination is very important if you have thousands of data from the database. Put the following code before the set page header comment of index.php file.

```
// page given in URL parameter, default page is one
$page = isset($_GET['page']) ? $_GET['page'] : 1;

// set number of records per page
$records_per_page = 5;

// calculate for the query LIMIT clause
$from_record_num = ($records_per_page * $page) - $records_per_

// retrieve records here
```

6.4 Retrieve Records from the Database

Now we will retrieve data from the database. Replace // retrieve records here comment of index.php with the following code.

```
// instantiate database and objects
$database = new Database();
$db = $database->getConnection();

$product = new Product($db);
$category = new Category($db);

// query products
$stmt = $product->readAll($from_record_num, $records_per_page)
$num = $stmt->rowCount();
```

6.5 Add readAll() Method in product.php

Retrieving records in the previous section won't work without this method. Put the following code inside our "product.php" file which is inside the "objects" folder.

```
function readAll($from_record_num, $records_per_page){

    $query = "SELECT
                id, name, description, price, category_id
            FROM
                " . $this->table_name . "
            ORDER BY
                name ASC
            LIMIT
                {$from_record_num}, {$records_per_page}";

    $stmt = $this->conn->prepare( $query );
    $stmt->execute();

    return $stmt;
}
```

6.6 Display data from the database

This time, we will show the list of records to the user. An HTML table will hold our data. Put the following code after the section 6.2 code.

```
    echo "<table class='table table-hover table-responsive tab
        echo "<tr>";
            echo "<th>Product</th>";
            echo "<th>Price</th>";
            echo "<th>Description</th>";
            echo "<th>Category</th>";
            echo "<th>Actions</th>";
        echo "</tr>";

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){

            extract($row);

            echo "<tr>";
                echo "<td>{$name}</td>";
                echo "<td>{$price}</td>";
                echo "<td>{$description}</td>";
                echo "<td>";
                    $category->id = $category_id;
                    $category->readName();
                    echo $category->name;
                echo "</td>";

                echo "<td>";
                    // read one, edit and delete button will b
                echo "</td>";

            echo "</tr>";

        }

    echo "</table>";

    // paging buttons will be here
}

// tell the user there are no products
else{
    echo "<div class='alert alert-info'>No products found.</di
}
```

6.7 Put the Read, Edit and Delete Action Buttons

Inside the "while" loop of the previous section, there is a comment "read one, edit and delete button will be here", replace that with the following code.

```
// read, edit and delete buttons
echo "<a href='read_one.php?id={$id}' class='btn btn-primary l
    <span class='glyphicon glyphicon-list'></span> Read
</a>

<a href='update_product.php?id={$id}' class='btn btn-info left
    <span class='glyphicon glyphicon-edit'></span> Edit
</a>

<a delete-id='{$id}' class='btn btn-danger delete-object'>
    <span class='glyphicon glyphicon-remove'></span> Delete
</a>";
```

6.8 Create paging.php for Paging Buttons

The following code will show our pagination buttons. Create a new file and name it "paging.php". Open that file and put the following code.

```
<?php
echo "<ul class='pagination'>";

// button for first page
if($page>1){
    echo "<li><a href='{$page_url}' title='Go to the first pag
        echo "First";
    echo "</a></li>";
}

// calculate total pages
$total_pages = ceil($total_rows / $records_per_page);

// range of links to show
$range = 2;

// display links to 'range of pages' around 'current page'
$initial_num = $page - $range;
```

```
    // be sure '$x is greater than 0' AND 'less than or equal
    if (($x > 0) && ($x <= $total_pages)) {

        // current page
        if ($x == $page) {
            echo "<li class='active'><a href=\"#\">$x <span cl
        }

        // not current page
        else {
            echo "<li><a href='{$page_url}page=$x'>$x</a></li>
        }
    }
}

// button for last page
if($page<$total_pages){
    echo "<li><a href='" .$page_url. "page={$total_pages}' tit
        echo "Last";
    echo "</a></li>";
}

echo "</ul>";
?>
```

6.9 Add the countAll() method in objects/product.php

The following code will be used to count the total number of records in the database. This will be used for pagination.

Open your product.php file which is inside the "objects" folder. Add the following method in the class.

```
// used for paging products
public function countAll(){

    $query = "SELECT id FROM " . $this->table_name . "";

    $stmt = $this->conn->prepare( $query );
    $stmt->execute();
```

6.10 Include paging.php in index.php

The following code will show our pagination buttons under our records list. Put the following code after the closing "table" tag of section 6.6 above.

```
// the page where this paging is used
$page_url = "index.php?";

// count all products in the database to calculate total pages
$total_rows = $product->countAll();

// paging buttons here
include_once 'paging.php';
```

6.11 Output

Run http://localhost/php-oop-crud-level-1/index.php [http://localhost/php-oop-crud-level-1/index.php] on your browser, you should see something like the image below.

List of records, page 1.

List of records, page 2.

7.0 UPDATING RECORD IN PHP THE OOP WAY

I know our PHP OOP CRUD tutorial is kinda long. Please take a break or drink some coffee first!

7.1 Create File: update_product.php

Create update_product.php file, open that file and put the following code.

```
<?php
// retrieve one product will be here

// set page header
$page_title = "Update Product";
include_once "layout_header.php";
```

```
?>
```

7.2 Create a "Read Products" Button

The following code will render a button. This button, when clicked, will let us go back to the records list. Replace the previous section's "contents will be here" comments with the following code.

```
echo "<div class='right-button-margin'>
          <a href='index.php' class='btn btn-default pull-righ
     </div>";

?>
<!-- 'update product' form will be here -->
```

7.3 Retrieve One Product Information Based on the Given ID.

The following code will retrieve data that will populate our HTML form. This is important because this will let the user know what exactly the record he is updating.

Open update_product.php file. Replace "// retrieve one product will be here" comment with the following code.

```
// get ID of the product to be edited
$id = isset($_GET['id']) ? $_GET['id'] : die('ERROR: missing I

// include database and object files
include_once 'config/database.php';
include_once 'objects/product.php';
include_once 'objects/category.php';

// get database connection
$database = new Database();
$db = $database->getConnection();

// prepare objects
$product = new Product($db);
```

```
// read the details of product to be edited
$product->readOne();
```

7.4 Add readOne() method in the Product Object Class.

The readOne() method used in the previous section will not work without the following code inside /objects/product.php file.

```
function readOne(){

    $query = "SELECT
                name, price, description, category_id
            FROM
                " . $this->table_name . "
            WHERE
                id = ?
            LIMIT
                0,1";

    $stmt = $this->conn->prepare( $query );
    $stmt->bindParam(1, $this->id);
    $stmt->execute();

    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    $this->name = $row['name'];
    $this->price = $row['price'];
    $this->description = $row['description'];
    $this->category_id = $row['category_id'];
}
```

7.5 Put the Values in the Form.

Now we can put the latest values to each form elements. Replace "<!-- 'update product' form will be here -->" comment of update_product.php with the following code.

```
<!-- post code will be here -->
```

```
        <tr>
            <td>Name</td>
            <td><input type='text' name='name' value='<?php ec
        </tr>

        <tr>
            <td>Price</td>
            <td><input type='text' name='price' value='<?php e
        </tr>

        <tr>
            <td>Description</td>
            <td><textarea name='description' class='form-contr
        </tr>

        <tr>
            <td>Category</td>
            <td>
                <!-- categories select drop-down will be here
            </td>
        </tr>

        <tr>
            <td></td>
            <td>
                <button type="submit" class="btn btn-primary">
            </td>
        </tr>

    </table>
</form>
```

7.6 Loop Through the Categories Records to show as Drop-down

The following code will list the categories in a drop-down.

Notice that we put "if($product->category_id==$category_id){..." inside the while loop. This is to pre-select the option of the current record.

Replace the previous section's comments "categories select drop-down will be here" with the following code.

```
// put them in a select drop-down
echo "<select class='form-control' name='category_id'>";

    echo "<option>Please select...</option>";
    while ($row_category = $stmt->fetch(PDO::FETCH_ASSOC)){
        $category_id=$row_category['id'];
        $category_name = $row_category['name'];

        // current category of the product must be selected
        if($product->category_id==$category_id){
            echo "<option value='$category_id' selected>";
        }else{
            echo "<option value='$category_id'>";
        }

        echo "$category_name</option>";
    }
echo "</select>";
?>
```

7.7 Code When Form was Submitted

The following code will assign the "posted" values to the object properties. Once assigned, it will update the database with those values using the update() method.

Open update_product.php file. Replace <!-- post code will be here -->" comment with the following code.

```
<?php
// if the form was submitted
if($_POST){

    // set product property values
    $product->name = $_POST['name'];
    $product->price = $_POST['price'];
    $product->description = $_POST['description'];
    $product->category_id = $_POST['category_id'];

    // update the product
    if($product->update()){
```

```
    // if unable to update the product, tell the user
    else{
        echo "<div class='alert alert-danger alert-dismissable
            echo "Unable to update product.";
        echo "</div>";
    }
}
?>
```

7.8 Update Code in the Product Class

The following code will make the previous section's "$product->update()" method work. Open our "product.php" which is inside the "objects" folder and add the following code.

```
function update(){

    $query = "UPDATE
                " . $this->table_name . "
            SET
                name = :name,
                price = :price,
                description = :description,
                category_id  = :category_id
            WHERE
                id = :id";

    $stmt = $this->conn->prepare($query);

    // posted values
    $this->name=htmlspecialchars(strip_tags($this->name));
    $this->price=htmlspecialchars(strip_tags($this->price));
    $this->description=htmlspecialchars(strip_tags($this->desc
    $this->category_id=htmlspecialchars(strip_tags($this->cate
    $this->id=htmlspecialchars(strip_tags($this->id));

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

```
    if($stmt->execute()){
        return true;
    }

    return false;

}
```

7.9 Output

Click any "Edit" button in the index page. The update record form should look like the following.

When you submit the form, a message will be shown.

A record was changed in the database.

We previously made the code for "update record", this section for reading one record from a database will be easier to do.

8.1 Create read_one.php file

This is the page where the data of a single record will be displayed. Create a new file and name it "read_one.php", open that file and put the following code.

```
<?php
// set page headers
$page_title = "Read One Product";
include_once "layout_header.php";

// read products button
echo "<div class='right-button-margin'>";
    echo "<a href='index.php' class='btn btn-primary pull-righ
        echo "<span class='glyphicon glyphicon-list'></span> R
    echo "</a>";
echo "</div>";

// set footer
include_once "layout_footer.php";
?>
```

8.2 Read one record based on given record ID

The following code will read a single record from the database. Put the following code before the "set page headers" comments of the previous section.

```
// get ID of the product to be read
$id = isset($_GET['id']) ? $_GET['id'] : die('ERROR: missing I

// include database and object files
include_once 'config/database.php';
include_once 'objects/product.php';
include_once 'objects/category.php';
```

```
// prepare objects
$product = new Product($db);
$category = new Category($db);

// set ID property of product to be read
$product->id = $id;

// read the details of product to be read
$product->readOne();
```

8.3 Display record on HTML table

This time, we will display the record details on an HTML table. Put the following code under the closing "div" tag of "Read Products" button.

```
// HTML table for displaying a product details
echo "<table class='table table-hover table-responsive table-b

    echo "<tr>";
        echo "<td>Name</td>";
        echo "<td>{$product->name}</td>";
    echo "</tr>";

    echo "<tr>";
        echo "<td>Price</td>";
        echo "<td>${$product->price}</td>";
    echo "</tr>";

    echo "<tr>";
        echo "<td>Description</td>";
        echo "<td>{$product->description}</td>";
    echo "</tr>";

    echo "<tr>";
        echo "<td>Category</td>";
        echo "<td>";
            // display category name
            $category->id=$product->category_id;
            $category->readName();
            echo $category->name;
        echo "</td>";
```

8.4 Output

Click any "Read" button in the index page, you should see something like the image below.

9.0 DELETING RECORD IN PHP THE OOP WAY

This is the last coding part of our PHP OOP CRUD Tutorial. Enjoy every code!

9.1 Put this JavaScript code in layout_footer.php

Put the following JavaScript code before the closing "body" tag in layout_footer.php file. We used Bootbox.js [http://bootboxjs.com/] to make a Bootstrap-style confirm dialog box.

```
<script>
```

```
    bootbox.confirm({
        message: "<h4>Are you sure?</h4>",
        buttons: {
            confirm: {
                label: '<span class="glyphicon glyphicon-ok"><
                className: 'btn-danger'
            },
            cancel: {
                label: '<span class="glyphicon glyphicon-remov
                className: 'btn-primary'
            }
        },
        callback: function (result) {

            if(result==true){
                $.post('delete_product.php', {
                    object_id: id
                }, function(data){
                    location.reload();
                }).fail(function() {
                    alert('Unable to delete.');
                });
            }
        }
    });

    return false;
});
</script>
```

9.2 Create delete_product.php

Create a new file and name it "delete_product.php". This file accepts the ID posted by the JavaScript code in the previous section. A record will be deleted from the database based on posted ID.

Open delete_product.php and put the following code.

```
<?php
// check if value was posted
```

```
    include_once 'objects/product.php';

    // get database connection
    $database = new Database();
    $db = $database->getConnection();

    // prepare product object
    $product = new Product($db);

    // set product id to be deleted
    $product->id = $_POST['object_id'];

    // delete the product
    if($product->delete()){
        echo "Object was deleted.";
    }

    // if unable to delete the product
    else{
        echo "Unable to delete object.";
    }
}
?>
```

9.3 Delete Code in Product Class

The previous section will not work with the "delete()" method in the product object. Open "product.php" which is inside the "objects" folder and put the following code.

```
// delete the product
function delete(){

    $query = "DELETE FROM " . $this->table_name . " WHERE id =

    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(1, $this->id);

    if($result = $stmt->execute()){
        return true;
    }else{
        return false;
```

9.4 Output

Click any "Delete" button in the index page. A pop up confirmation will be shown.

If the user clicks "OK" the record will be deleted and gone in the table.

A record was deleted in the database.

10.0 SEARCH RECORDS IN PHP THE OOP WAY

We'll continue by adding the search feature. This will answer the question: How to search data from database in php? This is a very useful feature because you enable your users to easily search a certain data from our MySQL database.

Please note that this is a bonus section. The code in this section is not included in our LEVEL 1 source code download.

10.1 Change index.php

We have to change index.php because we are adding a “search” feature and we want our code to be short. Our index.php will now look like the following code.

```
include_once 'config/core.php';

// include database and object files
include_once 'config/database.php';
include_once 'objects/product.php';
include_once 'objects/category.php';

// instantiate database and product object
$database = new Database();
$db = $database->getConnection();

$product = new Product($db);
$category = new Category($db);

$page_title = "Read Products";
include_once "layout_header.php";

// query products
$stmt = $product->readAll($from_record_num, $records_per_page)

// specify the page where paging is used
$page_url = "index.php?";

// count total rows - used for pagination
$total_rows=$product->countAll();

// read_template.php controls how the product list will be ren
include_once "read_template.php";

// layout_footer.php holds our javascript and closing html tag
include_once "layout_footer.php";
?>
```

10.2 Create read_template.php

Why do we need this template? We need it because exactly the same code can be used by index.php and search.php for displaying a list of records. Using a template means lesser code.

This template holds our search form as well.

```
        $search_value=isset($search_term) ? "value='{$search_t
        echo "<input type='text' class='form-control' placehol
        echo "<div class='input-group-btn'>";
            echo "<button class='btn btn-primary' type='submit
        echo "</div>";
    echo "</div>";
echo "</form>";

// create product button
echo "<div class='right-button-margin'>";
    echo "<a href='create_product.php' class='btn btn-primary
        echo "<span class='glyphicon glyphicon-plus'></span> C
    echo "</a>";
echo "</div>";

// display the products if there are any
if($total_rows>0){

    echo "<table class='table table-hover table-responsive tab
        echo "<tr>";
            echo "<th>Product</th>";
            echo "<th>Price</th>";
            echo "<th>Description</th>";
            echo "<th>Category</th>";
            echo "<th>Actions</th>";
        echo "</tr>";

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){

            extract($row);

            echo "<tr>";
                echo "<td>{$name}</td>";
                echo "<td>{$price}</td>";
                echo "<td>{$description}</td>";
                echo "<td>";
                    $category->id = $category_id;
                    $category->readName();
                    echo $category->name;
                echo "</td>";

                echo "<td>";

                    // read product button
                    echo "<a href='read_one.php?id={$id}' clas
```

```
                    echo "<a href='update_product.php?id={$id}
                        echo "<span class='glyphicon glyphicon
                    echo "</a>";

                    // delete product button
                    echo "<a delete-id='{$id}' class='btn btn-
                        echo "<span class='glyphicon glyphicon
                    echo "</a>";

                echo "</td>";

            echo "</tr>";

        }

    echo "</table>";

    // paging buttons
    include_once 'paging.php';
}

// tell the user there are no products
else{
    echo "<div class='alert alert-danger'>No products found.</
}
?>
```

10.3 Create core.php in "config" folder

Create a new folder and name it "config". Inside that folder, create a new file and name it "core.php".

This file will hold our pagination variables. Using a core.php file is a good practice, it can be used to hold other configuration values that you might need in the future.

Open core.php and put the following code.

```
<?php
// page given in URL parameter, default page is one
```

```
// calculate for the query LIMIT clause
$from_record_num = ($records_per_page * $page) - $records_per_
?>
```

10.4 Change paging.php code

The new paging.php code will look like the following.

```
<?php
echo "<ul class=\"pagination\">";

// button for first page
if($page>1){
    echo "<li><a href='{$page_url}' title='Go to the first pag
        echo "First Page";
    echo "</a></li>";
}

// count all products in the database to calculate total pages
$total_pages = ceil($total_rows / $records_per_page);

// range of links to show
$range = 2;

// display links to 'range of pages' around 'current page'
$initial_num = $page - $range;
$condition_limit_num = ($page + $range)  + 1;

for ($x=$initial_num; $x<$condition_limit_num; $x++) {

    // be sure '$x is greater than 0' AND 'less than or equal
    if (($x > 0) && ($x <= $total_pages)) {

        // current page
        if ($x == $page) {
            echo "<li class='active'><a href=\"#\">$x <span cl
        }

        // not current page
        else {
            echo "<li><a href='{$page_url}page=$x'>$x</a></li>
```

```
// button for last page
if($page<$total_pages){
    echo "<li><a href='" .$page_url . "page={$total_pages}' ti
        echo "Last Page";
    echo "</a></li>";
}

echo "</ul>";
?>
```

10.5 Include core.php and read_template.php

The core.php file will be included at the beginning of index.php file. The read_template.php will be included before the layout_footer.php inclusion. The new index.php will look like the following code

```
<?php
// core.php holds pagination variables
include_once 'config/core.php';

// include database and object files
include_once 'config/database.php';
include_once 'objects/product.php';
include_once 'objects/category.php';

// instantiate database and product object
$database = new Database();
$db = $database->getConnection();

$product = new Product($db);
$category = new Category($db);

$page_title = "Read Products";
include_once "layout_header.php";

// query products
$stmt = $product->readAll($from_record_num, $records_per_page)

// specify the page where paging is used
$page_url = "index.php?";
```

```
include_once "read_template.php";

// layout_footer.php holds our javascript and closing html tag
include_once "layout_footer.php";
?>
```

10.6 Create search.php

This is the most important file of this section. This file will display the records based on a user's search term.

Create a new file and name it "search.php". Open that file and put the following code.

```
<?php
// core.php holds pagination variables
include_once 'config/core.php';

// include database and object files
include_once 'config/database.php';
include_once 'objects/product.php';
include_once 'objects/category.php';

// instantiate database and product object
$database = new Database();
$db = $database->getConnection();

$product = new Product($db);
$category = new Category($db);

// get search term
$search_term=isset($_GET['s']) ? $_GET['s'] : '';

$page_title = "You searched for \"{$search_term}\"";
include_once "layout_header.php";

// query products
$stmt = $product->search($search_term, $from_record_num, $reco

// specify the page where paging is used
$page_url="search.php?s={$search_term}&";
```

```
// read_template.php controls how the product list will be ren
include_once "read_template.php";

// layout_footer.php holds our javascript and closing html tag
include_once "layout_footer.php";
?>
```

10.7 Add search() and countAll_BySearch() methods

Open "product.php" file which is inside the "objects" folder. Add the following methods in the class.

```
// read products by search term
public function search($search_term, $from_record_num, $record

    // select query
    $query = "SELECT
                c.name as category_name, p.id, p.name, p.descr
            FROM
                " . $this->table_name . " p
                LEFT JOIN
                    categories c
                        ON p.category_id = c.id
            WHERE
                p.name LIKE ? OR p.description LIKE ?
            ORDER BY
                p.name ASC
            LIMIT
                ?, ?";

    // prepare query statement
    $stmt = $this->conn->prepare( $query );

    // bind variable values
    $search_term = "%{$search_term}%";
    $stmt->bindParam(1, $search_term);
    $stmt->bindParam(2, $search_term);
    $stmt->bindParam(3, $from_record_num, PDO::PARAM_INT);
    $stmt->bindParam(4, $records_per_page, PDO::PARAM_INT);

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

```
public function countAll_BySearch($search_term){

    // select query
    $query = "SELECT
                COUNT(*) as total_rows
            FROM
                " . $this->table_name . " p
            WHERE
                p.name LIKE ? OR p.description LIKE ?";

    // prepare query statement
    $stmt = $this->conn->prepare( $query );

    // bind variable values
    $search_term = "%{$search_term}%";
    $stmt->bindParam(1, $search_term);
    $stmt->bindParam(2, $search_term);

    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    return $row['total_rows'];
}
```

10.8 Output

11.0 FILE UPLOAD IN PHP THE OOP WAY

In this section, we will add a "file upload" feature. This feature is included in the LEVEL 2 source code download.

11.1 Change HTML form

Open create_product.php and find the "form" tag. Change that line to the following code. The "enctype" enables the form to submit a file to the server.

```
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]
```

On the same HTML table, find the closing "tr" tag of the "Category" field. Add the following code. This adds an input field where the user can browse the file he wants to upload.

```
<tr>
    <td>Photo</td>
    <td><input type="file" name="image" /></td>
</tr>
```

11.2 Set value of "image" field

Open create_product.php and add the new "image" field. The value will be the file name of the submitted file. We used the built-in sha1_file() function the make the file name unique.

Open create_product.php file. Place the following code under $product->category_id = $_POST['category_id']; code.

```
$image=!empty($_FILES["image"]["name"])
        ? sha1_file($_FILES['image']['tmp_name']) . "-" . base
```

11.3 Change create() method

Open "objects" folder and open the "product.php" file inside it. Find the "create()" method.

Add the "image" field by changing the query to:

```
// insert query
$query = "INSERT INTO " . $this->table_name . "
            SET name=:name, price=:price, description=:descrip
                category_id=:category_id, image=:image, create
```

On the sanitize section, it will be:

```
$this->image=htmlspecialchars(strip_tags($this->image));
```

Then bind the value.

```
$stmt->bindParam(":image", $this->image);
```

Add the "image" property at the top of the class, maybe after public $category_id;

```
public $image;
```

Using the PhpMyAdmin, add an "image" field in the products table. Set the type to VARCHAR with 512 in length.

11.4 Call uploadPhoto() method

Open create_product.php and find this line.

```
// product was created in database
echo "<div class='alert alert-success'>Product was created.</d
```

Put the following code under the code above. This will call the uploadPhoto() method that will try to upload the file to server.

```
// try to upload the submitted file
// uploadPhoto() method will return an error message, if any.
echo $product->uploadPhoto();
```

11.5 Add uploadPhoto() method

The previous section will not work without the complete code of uploadPhoto() method.

Open "objects" folder and open the "product.php" file inside it. Add the following method inside the class.

```
// will upload image file to server
function uploadPhoto(){

    $result_message="";

    // now, if image is not empty, try to upload the image
    if($this->image){

        // sha1_file() function is used to make a unique file
        $target_directory = "uploads/";
        $target_file = $target_directory . $this->image;
        $file_type = pathinfo($target_file, PATHINFO_EXTENSION

        // error message is empty
        $file_upload_error_messages="";

    }

    return $result_message;
}
```

11.6 Validate submitted file

- Limit the allowed file types.
- Prevent multiple file on the server.
- Deny uploading files with large file size.
- Making sure the "uploads" directory exists.

Add the following code after $file_upload_error_messages=""; of the previous section.

```
// make sure that file is a real image
$check = getimagesize($_FILES["image"]["tmp_name"]);
if($check!==false){
    // submitted file is an image
}else{
    $file_upload_error_messages.="<div>Submitted file is not a
}

// make sure certain file types are allowed
$allowed_file_types=array("jpg", "jpeg", "png", "gif");
if(!in_array($file_type, $allowed_file_types)){
    $file_upload_error_messages.="<div>Only JPG, JPEG, PNG, GI
}

// make sure file does not exist
if(file_exists($target_file)){
    $file_upload_error_messages.="<div>Image already exists. T
}

// make sure submitted file is not too large, can't be larger
if($_FILES['image']['size'] > (1024000)){
    $file_upload_error_messages.="<div>Image must be less than
}

// make sure the 'uploads' folder exists
// if not, create it
if(!is_dir($target_directory)){
    mkdir($target_directory, 0777, true);
}
```

If the file is valid, we will upload the file to server. Specifically, in the "uploads" folder. If there's any error, we will return it to be shown to the user.

Place the following code after the previous section's code.

```
// if $file_upload_error_messages is still empty
if(empty($file_upload_error_messages)){
    // it means there are no errors, so try to upload the file
    if(move_uploaded_file($_FILES["image"]["tmp_name"], $targe
        // it means photo was uploaded
    }else{
        $result_message.="<div class='alert alert-danger'>";
            $result_message.="<div>Unable to upload photo.</di
            $result_message.="<div>Update the record to upload
        $result_message.="</div>";
    }
}

// if $file_upload_error_messages is NOT empty
else{
    // it means there are some errors, so show them to user
    $result_message.="<div class='alert alert-danger'>";
        $result_message.="{$file_upload_error_messages}";
        $result_message.="<div>Update the record to upload pho
    $result_message.="</div>";
}
```

11.8 Show uploaded image file

Open "objects" folder and open "product.php" file. Find readOne() method. Add the "image" field in the method. The new method should look like the following.

```
function readOne(){

    $query = "SELECT name, price, description, category_id, im
        FROM " . $this->table_name . "
        WHERE id = ?
```

```
    $stmt->execute();

    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    $this->name = $row['name'];
    $this->price = $row['price'];
    $this->description = $row['description'];
    $this->category_id = $row['category_id'];
    $this->image = $row['image'];
}
```

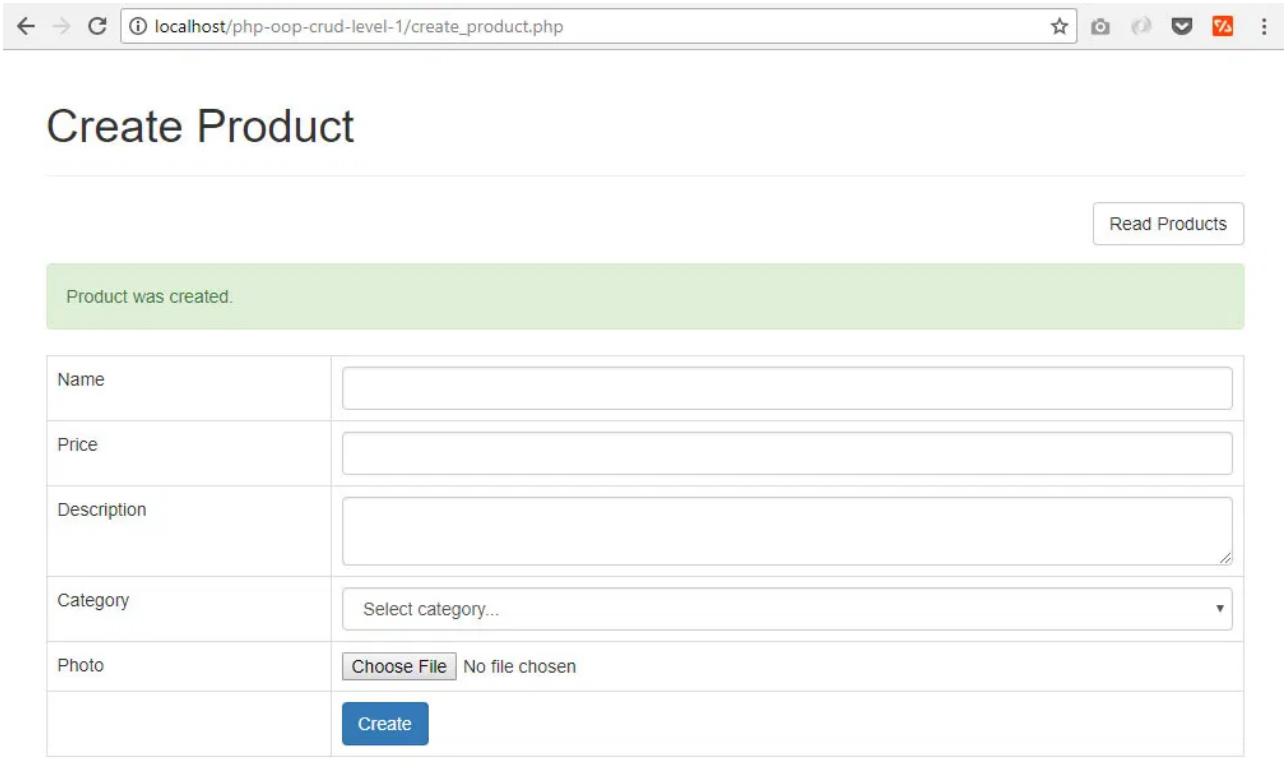
Open read_one.php file and find the closing "tr" tag of the "Category" field in the HTML table. Add the following code. This will show the uploaded image.

```
echo "<tr>";
    echo "<td>Image</td>";
    echo "<td>";
        echo $product->image ? "<img src='uploads/{$product->i
    echo "</td>";
echo "</tr>";
```

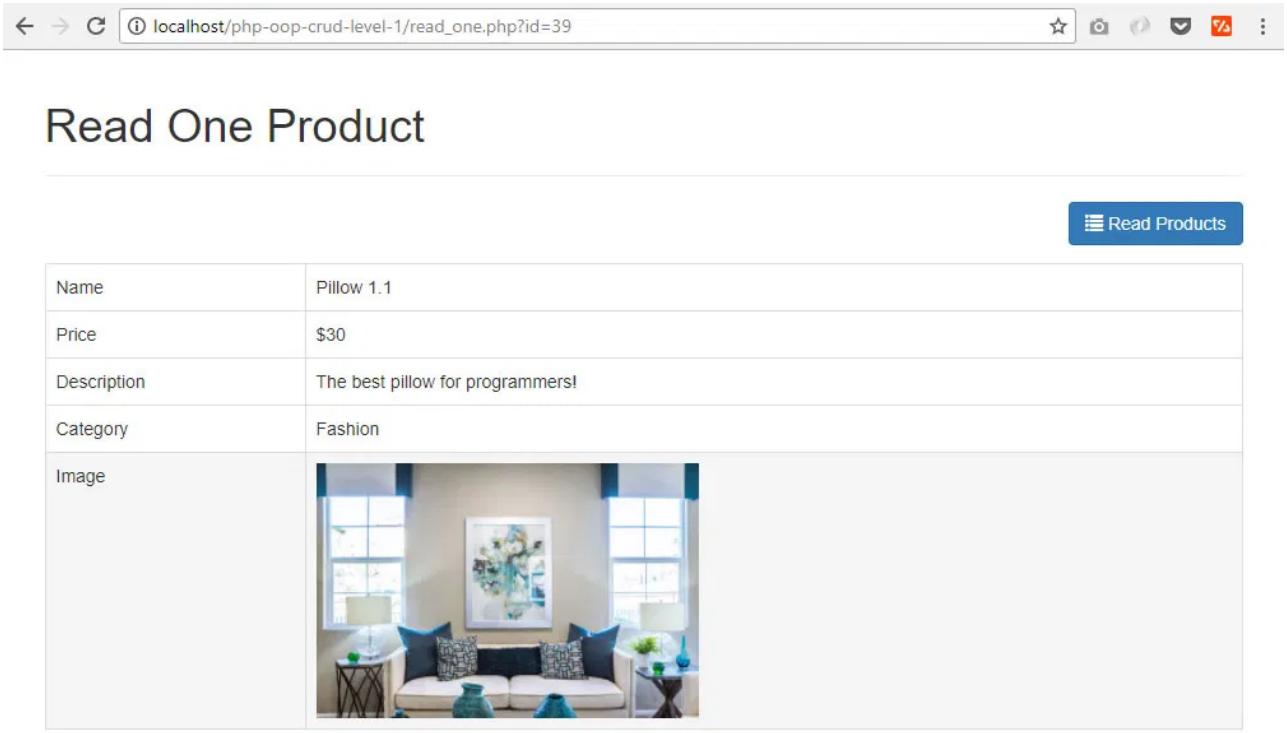
11.9 Output

Click the "Create" button, you will see something like the image below.

When you submitted the form, it will show a message prompt.



If you click the "Read" button of a record with an image, it will look like the following.



If the record has no image, it will say "No image found." message.

12.0 HOW TO RUN THE SOURCE CODES?

We highly recommend for you to follow and study our well-detailed, step-by-step tutorial above first. Nothing beats experience when it comes to learning.

But we believe you will learn faster if you'll see the final source code as well. We consider it as your additional guide.

Imagine the value or skill upgrade it can bring you. The additional income

By now, you need to download our source codes. To do it, use any download buttons in the next few sections below.

Once you downloaded the source codes, here's how you can run it.

1. Extract the files to your server directory.
2. Create your database using PhpMyAdmin, database name is "php_oop_crud_level_3" *
3. Import the SQL file called "php_oop_crud_level_3.sql" located in the "dev" folder.
4. You run index.php file. For example: http://localhost/php-oop-crud-level-3/index.php [http://localhost/php-oop-crud-level-3/index.php]

* Database name and SQL file is different for LEVEL 1 and 2 source codes.

13.0 DOWNLOAD LEVEL 1 SOURCE CODE

FEATURES	LEVEL 1
Object Oriented Programming Source Code	YES
PDO extension used	YES
Create product	YES
Read product	YES
Update product	YES
Delete product	YES

Pagination	YES
Bootstrap UI	YES
SQL file in "dev" folder	YES

☐ LEVEL 1 Source Code	$20.00
☐ LEVEL 2 Source Code	$40.00
☐ LEVEL 3 Source Code	$50.00
☑ ALL LEVELS Source Code	$80.00

DOWNLOAD NOW [#]

14.0 DOWNLOAD LEVEL 2 SOURCE CODE

FEATURES	LEVEL 2
All features of LEVEL 1 above	YES
HTML5 (font-end) validation for create product	YES
HTML5 (font-end) validation for update product	YES
Category selection for create and update product.	YES
Buttons with Glyphicons	YES
Search products by name or description	YES
HTML5 (font-end) validation for search product	YES
Pagination in search	YES

Export / download records to CSV	YES
Price display with dollar sign, comma and decimal point	YES
Multiple delete	YES
File upload field when creating or updating record	YES

☐ LEVEL 1 Source Code $20.00

☐ LEVEL 2 Source Code $40.00

☐ LEVEL 3 Source Code $50.00

☑ ALL LEVELS Source Code $80.00

DOWNLOAD NOW [#]

15.0 DOWNLOAD LEVEL 3 SOURCE CODE

FEATURES	LEVEL 3
All features of LEVEL 1 and 2 above	YES
Bootstrap navigation bar	YES
Select category in navigation	YES
Higlight category in navigation	YES
Create category	YES
Read category	YES
Update category	YES

View products by category	YES
Pagination for category	YES
Search category	YES
Pagination for category search	YES
Server side validation for create product & category	YES
Server side validation for update product & category	YES
Sorting by fields	YES
Pagination for sorting by fields	YES
jQuery UI enabled	YES
Search product by date range - record date created	YES
Pagination for earch product by date range	YES
jQuery UI calendar for picking date	YES

☐ LEVEL 1 Source Code	$20.00
☐ LEVEL 2 Source Code	$40.00
☐ LEVEL 3 Source Code	$50.00
☑ ALL LEVELS Source Code	$80.00

DOWNLOAD NOW [#]

16.0 DOWNLOAD ALL LEVELS

then you click the "Download Now" button.

☐ LEVEL 1 Source Code	$20.00
☐ LEVEL 2 Source Code	$40.00
☐ LEVEL 3 Source Code	$50.00
☑ ALL LEVELS Source Code	$80.00

DOWNLOAD NOW [#]

Do you need more reasons to get it?

MORE REASONS TO DOWNLOAD THE CODE	ALL
Use new skills for your multiple projects	YES
Save huge amount of time learning Bootstrap and PHP	YES
Code examples are direct to the point	YES
Well explained and commented source code	YES
Fast and friendly email support	YES
Free source code updates	YES

17.0 WHAT'S NEXT?

I hope you learned a lot from our PHP OOP CRUD Tutorial! Learning PHP Object Oriented Programming is fun and it can dramatically improve your career.

A. Learn our PHP Login Script with Session Tutorial – Step by Step Guide! [https://www.codeofaninja.com/2013/03/php-login-script.html] - Let's put our PHP OOP CRUD knowledge to work by building a simple PHP login script.

B. Learn How to create a simple REST API in PHP - Step by Step Guide! [https://www.codeofaninja.com/2017/02/create-simple-rest-api-in-php.html] - this is required before we go to our JavaScript programming tutorials.

18.0 RELATED TUTORIALS

GETTING STARTED

Learn the basics of web programming.

How to Run a PHP Script? [https://www.codeofaninja.com/2013/06/how-to-run-a-php-script.html]

JavaScript Tutorial for Beginners [https://www.codeofaninja.com/2020/07/javascript-tutorial-for-beginners-step-by-step.html]

Bootstrap Tutorial for Beginners [https://www.codeofaninja.com/2014/05/bootstrap-tutorial-beginners-step-step.html]

PHP PROGRAMMING TUTORIALS

Start something awesome with PHP!

PHP CRUD Tutorial for Beginners [https://www.codeofaninja.com/2011/12/php-and-mysql-crud-tutorial.html]

PHP OOP CRUD Tutorial [https://www.codeofaninja.com/2014/06/php

PHP Login Script with Session [https://www.codeofaninja.com/2013/03/php-login-script.html]

PHP Shopping Cart Tutorial [https://www.codeofaninja.com/2015/08/simple-php-mysql-shopping-cart-tutorial.html]

PHP Shopping Cart Tutorial 2.0 [https://www.codeofaninja.com/2013/04/shopping-cart-in-php.html]

REST API TUTORIALS

Welcome to the new world of web development!

PHP REST API Tutorial [https://www.codeofaninja.com/2017/02/create-simple-rest-api-in-php.html]

PHP REST API Authentication Example [https://www.codeofaninja.com/2018/09/rest-api-authentication-example-php-jwt-tutorial.html]

AJAX CRUD Tutorial [https://www.codeofaninja.com/2015/06/php-crud-with-ajax-and-oop.html]

PHP WEB APP SOURCE CODES

Scripts that will help you build a web app.

PHP SHOPPING CART SYSTEM [https://www.codeofaninja.com/2016/04/php-shopping-cart-source-code-download.html]

Download By Modules:

PHP Login & Registration Module [https://www.codeofaninja.com/2016/05/php-login-system-tutorial.html]

PHP Shopping Cart Module

PHP Product Catalog Module [https://www.codeofaninja.com/2016/07/php-product-catalog-script.html]

PHP Content Management Module [https://www.codeofaninja.com/2016/07/php-web-page-content-management-module.html]

PHP Contact Form Module [https://www.codeofaninja.com/2016/07/php-contact-form-messages-module.html]

PHP PayPal Integration Module [https://www.codeofaninja.com/2016/04/paypal-integration-in-php.html]

MORE SCRIPTS

More code examples that can be useful for you!

Shopping Cart Tutorial using COOKIES [https://www.codeofaninja.com/2014/09/php-shopping-cart-tutorial-using-cookies.html]

Extra Tutorials [https://www.codeofaninja.com/extras]

19.0 NOTES

#1 Found An Issue?

If you found a problem with this code, please write a comment below. Please be descriptive about your issue. Please provide the error messages, screenshots (or screencast) and your test URL. Thanks!

Before you write a comment, remember to read this guide [https://www.codeofaninja.com/how-to-write-a-great-comment-on

#2 Become a true Ninja!

We constantly improve CodeOfaNinja. We update our tutorials and source codes. Receive valuable web programming tutorials and updates to your email. Subscribe now!

#3 Thank You!

Please note that this post is in continuous development, meaning I'll update it every now and then.

If you have a friend or know someone who needs this PHP OOP CRUD Tutorial, please share this page to them! I know you will help them a lot by doing it. Thanks!

Featured Comment

Steve Janssen • 3 years ago

Hi Mike
Just noticed ive asked various questions but dont think ive actually commented on how good it is...
This a great script, gives you everything need with easy instruction, clean/organised scripting and adaptable. So are your other scripts - ive tested a few.
This tutorial ive adapted and built on to make an online booking system for a beauty therapy business where the owner can confirm the bookings then set to pay status afterwards.
Ive added additional objects for controlling the different availability dates per location added a transaction ID when confirming payment.
There will also be an accounts section and email notifications based on status added.

All built from your foundation tutorial!

7 ^ | v 2 • Share ›

Mohamed Hüsnü • 4 years ago

you are very generous

72 ^ | v • Reply • Share ›

Mike Dalisay Mike Dalisay ➚ Mohamed Hüsnü • 4 years ago

Thank you!

^ | v • Reply • Share ›

LostInLust • 6 years ago

WOW :D very nice tutor from our sensei here :) this code is very very very nice and well structured compared to the other :D now my jutsu will be more elegant with this :D

60 ^ | v • Reply • Share ›

Mike Dalisay Mike Dalisay ➚ LostInLust • 6 years ago

Hello **@LostInLust**, thanks for the kind words! I'm glad to help make your jutsu one step higher! Please share our site to your fellow Ninjas if you have time. :)

^ | v • Reply • Share ›

sheetesh • 7 years ago

Thank You GREAT CODE (y) ....this coding helps the beginners like me to understand the basic oops CRUD concept with responsive design. and no need to $x++ n i have remove it working nicely.

$category->id = $category_id;

$category->readName();

echo $category->name;

this code was not working n i have replace this code by

$cat=$category->readName($cat_id);

$rowcategory = $cat->fetch(PDO::FETCH_ASSOC);

echo $rowcategory ['name'];

It is working but is that good manner to write code like i wrote.
M very graceful of you Thanks a lot sir..

45 ^ | v • Reply • Share ›

Nathan Soares ➚ sheetesh • 7 years ago

The category is not showing, return an undefined method readName error, please, van u help me?

1 ^ | v 1 • Reply • Share ›

frank lemwett ➚ Nathan Soares • 2 years ago • edited

did you manage to sort...i just did.....include category inside the class not

```
function read(){
//select all data
$query = "SELECT
id, name
FROM
" . $this->table_name . "
ORDER BY
name";

$stmt = $this->conn->prepare( $query );
$stmt->execute();

return $stmt;
}
// used to read category name by its ID
function readName(){

$query = "SELECT name FROM " . $this->table_name . " WHERE id = ?
limit 0,1";

$stmt = $this->conn->prepare( $query );
$stmt->bindParam(1, $this->id);
$stmt->execute();

$row = $stmt->fetch(PDO::FETCH_ASSOC);

$this->name = $row['name'];
}

}

?>
```

1 ^ | v • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ frank lemwett • 2 years ago

Thanks for sharing your solution **@frank lemwett**!

^ | v • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ sheetesh • 7 years ago

Hey **@sheetesh**, you're welcome! Thanks for commenting about it, I'll update the code!

^ | v • Reply • Share ›

Avatar **Nathan Soares** ➚ Mike Dalisay • 7 years ago

Avatar **Mike Dalisay** Mike Dalisay ➚ Nathan Soares • 7 years ago

Hey Nathan Soares, you're welcome! Please try to add the following code in your category.php

```
function readName(){

$query = "SELECT name FROM " . $this->table_name . "
WHERE id = ? limit 0,1";
$stmt = $this->conn->prepare( $query );
$stmt->bindParam(1, $this->id);
$stmt->execute();
$row = $stmt->fetch(PDO::FETCH_ASSOC);

$this->name = $row['name'];

}
```

^ | v • Reply • Share ›

Avatar **Punith** ➚ Mike Dalisay • 2 years ago

Still I am getting same error can you help with this

^ | v • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Punith • 2 years ago

Hi **@Punith**, please make sure your method is inside the class.

^ | v • Reply • Share ›

Avatar **Steve Janssen** • 3 years ago

Hi Mike

i added an object function for setting the "image" field to NULL using the same javascript promptfor delete however when removing the filename from the database i want to delete to the image also from the uploads directory, i am trying to use the PHP unlink command but i cannot figure it out.

here is my deleteBackup object function , can you advise me how to add the unlink or another way to remove filename from DB and file from the directory?

i can add the unlink command to its own button but when the page loads it runs and removes the image without clicking on the button.

seperate button:
image); ?> class='btn btn-info btn-sm'>Delete Image

objection function:

```
{
$query = "UPDATE " . $this->table_name . " SET image = NULL WHERE id = ?";

$stmt = $this->conn->prepare($query);
$stmt->bindParam(1, $this->id);

if($result = $stmt->execute()){
return true;
}else{
return false;
}
}
```

14 ^ | v • Reply • Share ›

Mike Dalisay Mike Dalisay ➚ Steve Janssen • 3 years ago • edited

Hi **@Steve Janssen**, your deleteBackup() method should look like this:

```php
function deleteBackup(){

    $this->readOne();
    unlink("uploads/" . $this->image);

    $query = "UPDATE " . $this->table_name . " SET image = NULL WHERE id = ?";

    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(1, $this->id);

    if($result = $stmt->execute()){
        return true;
    }

    return false;
}
```

^ | v • Reply • Share ›

Steve Janssen • 2 years ago

Hi Mike,

:i am trying to setup a reminder_alert script using PHPMAILER (will make this CRON job later) that auto emails the client a reminder, the scripts works, however when it runs it only emails the first client on the list and not the others.
Any ideas? am i mssing a foreach or loop statement somewhere?

FUNCTION OBJECT:

```php
function reminder24($from_record_num, $records_per_page){

$query = "SELECT
            status, location, booking_date, booking_time, cust_name, cust_phone,
        FROM
            " . $this->table_name . " WHERE status = 'Confirmed' ";
```

```
public function countReminder24(){

    $query = "SELECT id FROM " . $this->table_name . " WHERE status = 'Confirmed' ";

    $stmt = $this->conn->prepare( $query );
    $stmt->execute();

    $num = $stmt->rowCount();

    return $num;
}
```

REMINDER SCRIPT:

```
getConnection();

$bookings = new Bookings($db);

// query unconfirmed bookings
$stmt = $bookings->reminder24($from_record_num, $records_per_page);

// count total rows - used for pagination
$total_rows=$bookings->countreminder24();
?>
0){

        while ( $row = $stmt->fetch(PDO::FETCH_ASSOC)){

            extract($row);
include_once 'reminder_alert.php';

}}

?>
```

8 ^ | v • Reply • Share ›

Mike Dalisay Mike Dalisay ➚ Steve Janssen • 2 years ago • edited

Hi @Steve, what's in the reminder_alert.php? I think the error is you put the include_once 'reminder_alert.php'; inside your while loop.

I suggest you try to put the actual code instead of a PHP include.

^ | v • Reply • Share ›

Steve Janssen • 3 years ago

Featured by The Code Of A Ninja

Hi Mike

Just noticed ive asked various questions but dont think ive actually commented on how good it is...

This a great script, gives you everything need with easy instruction, clean/organised

Ive added additional objects for controlling the different availability dates per location added a transaction ID when confirming payment.
There will also be an accounts section and email notifications based on status added.

All built from your foundation tutorial!

7 ^ | v 2 • Reply • Share ›

Mike Dalisay Mike Dalisay → Steve Janssen • 3 years ago • edited

Hi **@Steve Janssen**, thanks for the kind words and sharing your story! I'm glad you were able to develop an online booking system with the help of our tutorials. It feels great to know that the purpose of our site is being served. :)

^ | v • Reply • Share ›

Vio Techniques • a year ago

Hi,

Would it be possible in the level 3 downloadable source code to add an image view in the read.php script?

3 ^ | v • Reply • Share ›

Mike Dalisay Mike Dalisay → Vio Techniques • a year ago

Hi **@Vio Techniques**! Do you mean you want to show a thumbnail for each product in the list? Yes, that is possible to do.

^ | v • Reply • Share ›

Maro Pinčević • a year ago

how to make multiple selection category for products

3 ^ | v • Reply • Share ›

Mike Dalisay Mike Dalisay → Maro Pinčević • a year ago

Hi **@Maro Pinčević**! What I will suggest is you create another table that will hold the product IDs and category IDs. That way, you can save multiple categories for a product.

^ | v • Reply • Share ›

jack.kelly7x • a year ago

Hey Mike,
at Section 10.7 within the "public function countAll_BySearch" you are not defining the 2nd parameter for the bindParam, you need to add "$stmt->bindParam(2, $search_term);" after "$stmt->bindParam(1, $search_term);" otherwise your search query does not work

Hi **@jack.kelly7x**, thanks for pointing this out, I already updated the tutorial above.

^ | v • Reply • Share ›

Avatar **Bernie Rebolledo** • 2 years ago

Please help i got an error when searching

Warning: PDOStatement::execute(): SQLSTATE[HY093]: Invalid parameter number: number of bound variables does not match number of tokens in C:\xampp\htdocs\php-oop-crud-level-1\objects\product.php on line 207

2 ^ | v • Reply • Share ›

Avatar **Luis Galindo** ➚ Bernie Rebolledo • a year ago

You should add:

$stmt->bindParam(2, $search_term);

before execute,

1 ^ | v • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Luis Galindo • a year ago

@Luis Galindo I already updated the tutorial above with the fix, thank you!

^ | v • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Bernie Rebolledo • a year ago

Hi **@Bernie Rebolledo**, I'm unable to reproduce the issue. Make sure the number of bound variables are the same with ? or : in your query. Would you show us your code?

^ | v 1 • Reply • Share ›

Avatar **Zaem Shakkir** • 3 years ago • edited

hi Mike, i follow ur step till section 7.5 , but my update_product.php is mess up, could u tell me what is going on ?

2 • Reply • Share ›

Avatar **frank lemwett** ➚ Zaem Shakkir • 2 years ago
start your code with these two lines.....currently they are @your footer
• Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ frank lemwett • 2 years ago
Hi **@frank lemwett**, would you tell us which two lines was it?
• Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Zaem Shakkir • 3 years ago • edited
Hi **@Zaem Shakkir**, I'm unable to replicate the issue. It looks like you misplaced some code. Would you show us the code of your update_product.php?
• Reply • Share ›

Avatar **Justin Pullen** • 7 years ago • edited
Hmmm...I'm getting an error on the pagination part. $total_rows = $product->countAll();, countAll is undefined in $product.
Edit to add exact message.
Fatal error: Call to undefined method $product::countAll()
3 1 • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Justin Pullen • 7 years ago • edited
Hey **@Justin Pullen**, see the new 6.9, just add that countAll() method code in objects/product.php
2 • Reply • Share ›

Avatar **Tim Yakamoto** • 7 years ago
I'm getting the following error:

Fatal error: Call to undefined method Product::getTimestamp() in /home/codio/workspace/app/a_phpcrudexample/objects/product.php on line 24
1 • Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Tim Yakamoto • 6 years ago
@Tim, please see 4.11 above...
• Reply • Share ›

Avatar **Tresna Dwipayana** • 3 months ago

Avatar Hi @Tresna, uploading a different type of file is not within the scope of our tutorial. But you can try adding the file type in the allowed file types array of section 11.6 above.

Also, I've found some resources that might help you: https://stackoverflow.com/q...

Reply • Share ›

Avatar **vieng** • 10 months ago

can i use for progresql database ?

Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ vieng • 10 months ago

Hi **@vieng**, sorry, our code is not tested with PostgreSQL.

Reply • Share ›

Avatar **vieng** ➚ Mike Dalisay • 10 months ago

Can you make it to work with PostgreSQL ?

please..

Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ vieng • 10 months ago

@vieng sorry, it is not part of our tutorial. But we might do it in the future. Please subscribe so you will be notified.

Reply • Share ›

Avatar **Yusuff Mustapha** • 10 months ago

Thank you for your generosity...I can not thank you well enough.

Please I am having this error...Warning: getimagesize(): Filename cannot be empty in C:\xampp\htdocs\test\codeofninja-php-oop-crud-0\objects\product.php on line 244

I followed your suggestions here...

> To prevent this from happening, you should check if the user uploaded an image, if there is not, you should use the previous value.
>
> Use readOne() method to get the previous image value. You can code this the way I described it or you can download our LEVEL 2 source code above.

Reply • Share ›

• Reply • Share ›

Avatar **Maksym Kinash** • 10 months ago

I have a problem. read and change panels do not show database.

• Reply • Share ›

Avatar **Mike Dalisay** Mike Dalisay ➚ Maksym Kinash • 10 months ago

Hi **@Maksym Kinash**, what do you mean by "do not show database"? Would you elaborate?

• Reply • Share ›

Avatar **Maksym Kinash** • 10 months ago

Help me !!! pls