PHP CRUD Tutorial (part 1)

This is the part 1 of 3 in PHP CRUD tutorial.

- PHP CRUD tutorial (part 1)
- PHP CRUD tutorial (part 2)
- PHP CRUD tutorial (part 3)
- Advanced PHP CRUD tutorial

Creating CRUD grid is a very common task in web development (CRUD stands for Create/Read/Update/Delete). If you are a senior web developer, you must have created plenty of CRUD grids already. They maybe exist in a content management system, an inventory management system, or accounting software. If you just started web development, you are certainly going to experience lots of CRUD grids' creation work in your later career.

The main purpose of a CRUD grid is that enables users create/read/update/delete data. Normally data is stored in MySQL Database.PHP will be the server-side language that manipulates MySQL Database tables to give front-end users power to perform CRUD actions.

In this tutorial series, we will go through steps of a creating PHP CRUD grid. We want to demonstrate how PHP as a server side language, communicates with backend MySQL, and meanwhile renders front-end HTML. We hope you can learn something from this tutorial.

Table Of Content

1. Creating a sample Database table

In this tutorial, we will work on a simple Database table as below. After this tutorial, you should use the idea here to create CRUD grid on your own Database tables.

Import the Database table below to your own MySQL Database.

?

```
CREATE TABLE `customers` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`name` VARCHAR( 100 ) NOT NULL ,
`email` VARCHAR( 100 ) NOT NULL ,
`mobile` VARCHAR( 100 ) NOT NULL
) ENGINE = INNODB;
```

As you can see, this is a very simple table for tracking customers' information (name, email address and mobile number). And we prefer an auto incremental primary key (id).

2. Connecting to Database

Create a PHP file "database.php"; this file contains a PHP class named "Database". Throughout this application, Database handles all the stuff related to database connections, such as connecting and disconnecting.

?

```php
<?php
class Database
{
    private static $dbName = 'crud_tutorial' ;
    private static $dbHost = 'localhost' ;
    private static $dbUsername = 'root';
    private static $dbUserPassword = 'root';

    private static $cont = null;

    public function __construct() {
        die('Init function is not allowed');
    }

    public static function connect()
    {
       // One connection through whole application
       if ( null == self::$cont )
       {
        try
        {
          self::$cont =  new PDO( "mysql:host=".self::$dbHost.";"."dbname=".self::$dbName, self::$dbUsername, self::$dbUserPassword);
        }
        catch(PDOException $e)
        {
          die($e->getMessage());
        }
       }
       return self::$cont;
    }

    public static function disconnect()
    {
        self::$cont = null;
    }
}
?>
```

As you can see, we are using PDO for database access. There are a lot of benefits of using PDO. One of the most significant benefits is that it provides a uniform method of access to multiple databases.

To use this class, you will need to supply correct values for $dbName, $dbHost, $dbUsername, $dbUserPassword.

- **$dbName**: Database name which you use to store 'customers' table.
- **$dbHost**: Database host, this is normally "localhost".
- **$dbUsername**: Database username.
- **$dbUserPassword**: Database user's password.

Let us take a look at three functions of this class:

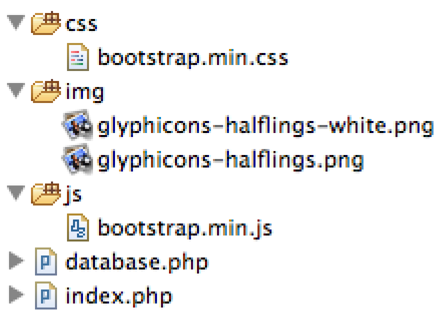
- **__construct()**: This is the constructor of class Database. Since it is a **static class, initialization of this class is not allowed**. To prevent misuse of the class, we use a "die" function to remind users.
- **connect**: This is the main function of this class. It uses singleton pattern to make sure only one PDO connection exist across the whole application. Since it is a static method. We use **Database::connect()** to create a connection.
- **disconnect**: Disconnect from database. It simply sets connection to NULL. We need to call this function to close connection.

3. Create a Twitter Bootstrap powered grid

Here comes the grid without CRUD capabilities. Because CRUD operation can only be performed when there is a grid. We firstly need to build a grid. From there we can subsequently add "Create" page, "Read" page, "Update" page and finally "Delete" page.

Head to Bootstrap official page, and download a copy. In this tutorial, we are using version 2.3.2. After that, create a PHP file "index.php".

Current file structure should look like below if you have followed the steps correctly:



```
css
    bootstrap.min.css
img
    glyphicons-halflings-white.png
    glyphicons-halflings.png
js
    bootstrap.min.js
database.php
index.php
```

Now open file "index.php". And copy the content below:

?

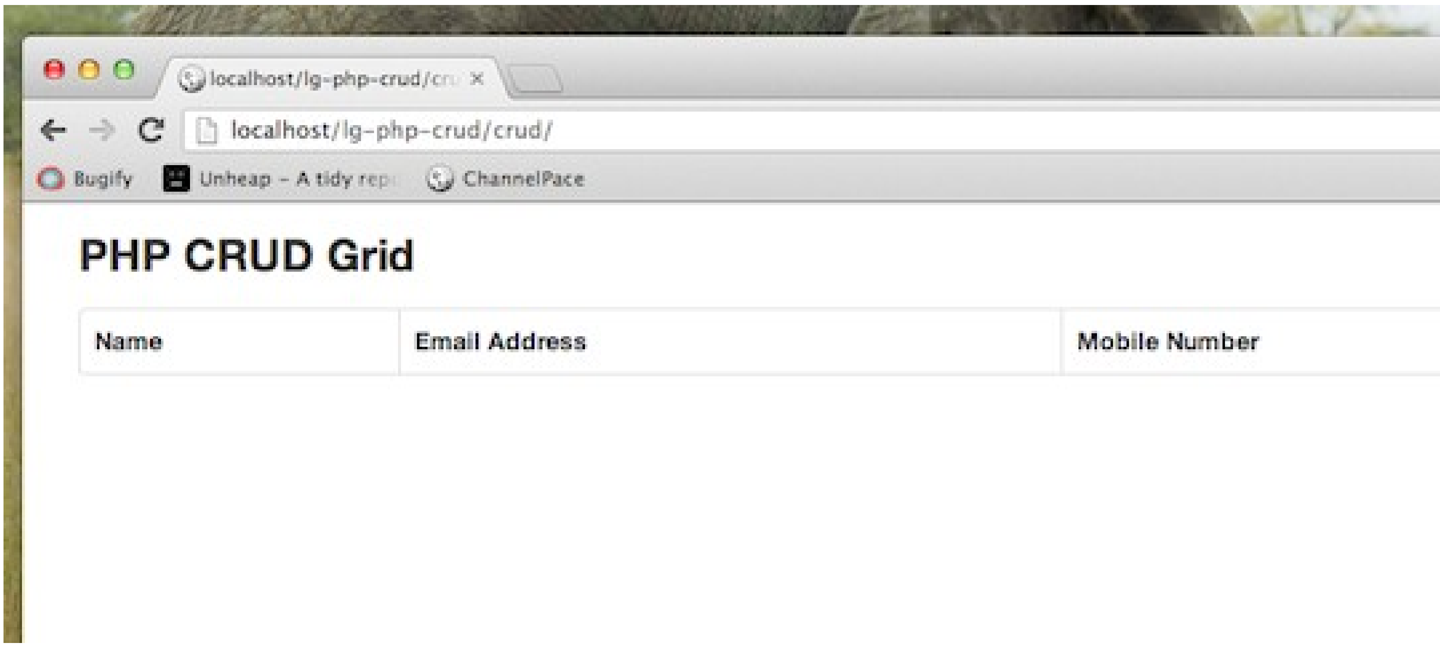
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <link   href="css/bootstrap.min.css" rel="stylesheet">
6      <script src="js/bootstrap.min.js"></script>
7  </head>
8  
9  <body>
10     <div class="container">
11             <div class="row">
12                 <h3>PHP CRUD Grid</h3>
13             </div>
14             <div class="row">
15                 <table class="table table-striped table-bordered">
16                   <thead>
17                     <tr>
18                       <th>Name</th>
19                       <th>Email Address</th>
20                       <th>Mobile Number</th>
21                     </tr>
22                   </thead>
23                   <tbody>
24                   <?php
25                    include 'database.php';
26                    $pdo = Database::connect();
27                    $sql = 'SELECT * FROM customers ORDER BY id DESC';
28                    foreach ($pdo->query($sql) as $row) {
29                             echo '<tr>';
30                             echo '<td>'. $row['name'] . '</td>';
31                             echo '<td>'. $row['email'] . '</td>';
32                             echo '<td>'. $row['mobile'] . '</td>';
33                             echo '</tr>';
34                    }
35                    Database::disconnect();
36                   ?>
37                   </tbody>
38             </table>
39         </div>
40     </div> <!-- /container -->
41   </body>
42 </html>
```

Let us dig into the codes.

- <head> part of this file is straightforward, we include Bootstrap's CSS and JavaScript files.
- Line 15 to 38 is the main part of the file. It is where we retrieve data from database and show it on the grid. Let dig deep into each lines carefully.

- We firstly create a table with headers corresponding to database table "customers"'s fields. Which includes "Name", "Email Address", "Mobile Number":
  ?
  ```
  <thead>
             <tr>
               <th>Name</th>
               <th>Email Address</th>
               <th>Mobile Number</th>
             </tr>
  </thead>
  ```
- Then we include "database.php", create a PDO connection to database, and use a general "SELECT" statement to retrieve data. Lastly, we loop through each row to print content. Do not forget to close the connection as we mentioned at the beginning.
  ?
  ```
  <?php
             include 'database.php';
             $pdo = Database::connect();
             $sql = 'SELECT * FROM customers ORDER BY id
  DESC';
             foreach ($pdo->query($sql) as $row) {
                      echo '<tr>';
                      echo '<td>'. $row['name'] . '</td>';
                      echo '<td>'. $row['email'] . '</td>';
                      echo '<td>'. $row['mobile'] . '</td>';
                      echo '</tr>';
             }
             Database::disconnect();
  ?>
  ```

If you have followed correctly, you should have an empty grid as below if you navigate to "index.php" from browser:



PHP CRUD Grid

Name	Email Address	Mobile Number

The grid is empty because there is no actual data inside "customers" table, to test if it is actually working, you can manually insert data into "customers" table. It should show them on the grid.

4. To be continued

As this post is getting too long, let us move on to next post. In PHP CRUD tutorial part 2, we demonstrate how to create the "Create" and "Read" pages of PHP CRUD grid.

Hopefully this simple tutorial helped you with your development.
If you like our post, please follow us on Twitter and help spread the word. We need your support to continue.
If you have questions or find our mistakes in above tutorial, do leave a comment below to let us know.

In this part of tutorial, we are going to create "Create" and "Read" pages of PHP CRUD grid.

Table Of Content

1. Adding "Create" button and "Read" button

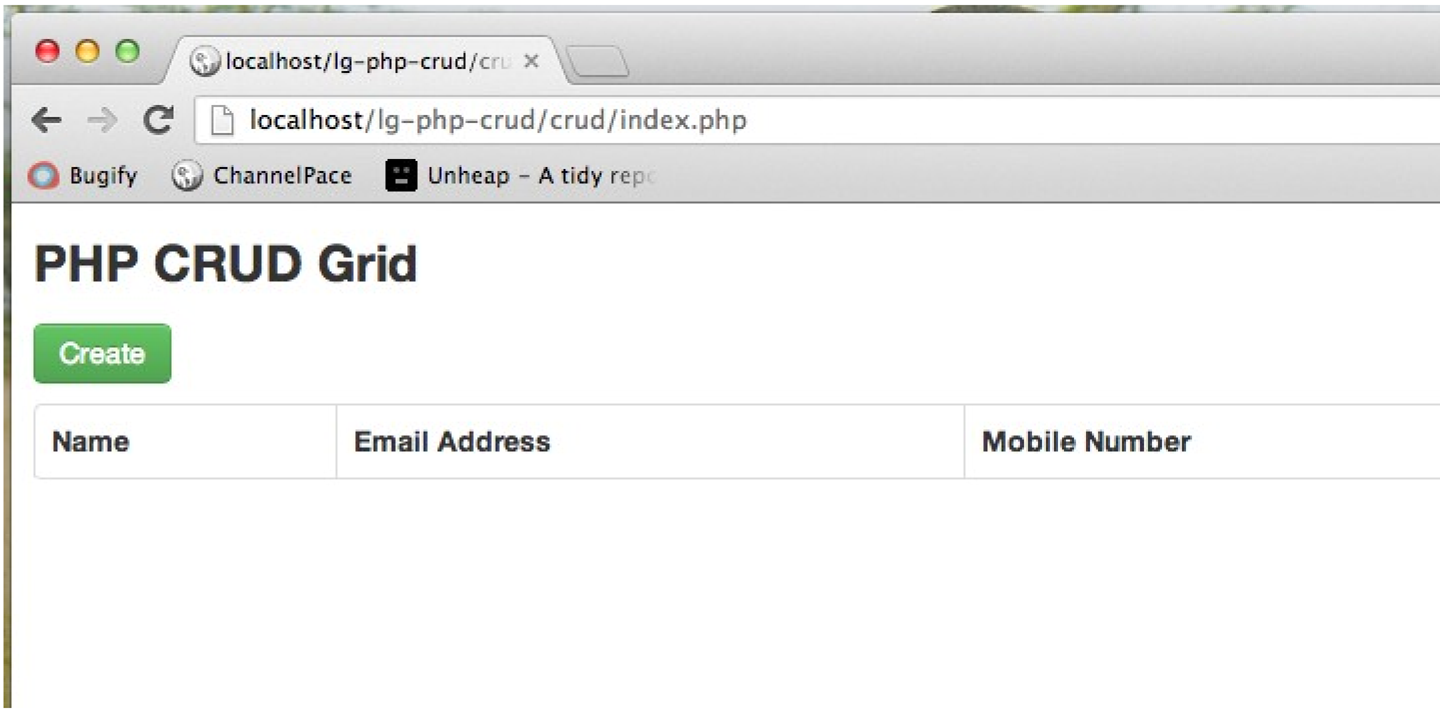
To start, we firstly need a navigation button which leads to "Create" page and also a button to read a record. Open "index.php" file we created in part 1 of this tutorial. And add a "Create" button at the top of the table, and also "Read" buttons for each row of the table.

Now the "index.php" file's code should look like below, the highlighted codes are what we have added:

?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link   href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">
            <div class="row">
                <h3>PHP CRUD Grid</h3>
            </div>
            <div class="row">
                <p>
                    <a href="create.php" class="btn btn-
success">Create</a>
                </p>
                <table class="table table-striped table-
bordered">
                      <thead>
                        <tr>
                          <th>Name</th>
                          <th>Email Address</th>
                          <th>Mobile Number</th>
                          <th>Action</th>
                        </tr>
                      </thead>
                      <tbody>
                      <?php
                       include 'database.php';
                       $pdo = Database::connect();
                       $sql = 'SELECT * FROM customers ORDER BY
id DESC';
                       foreach ($pdo->query($sql) as $row) {
                                echo '<tr>';
                                echo '<td>'. $row['name'] .
'</td>';
                                echo '<td>'. $row['email'] .
'</td>';
                                echo '<td>'. $row['mobile'] .
'</td>';
                                echo '<td><a class="btn"
href="read.php?id='.$row['id'].'">Read</a></td>';
                                echo '</tr>';
                       }
                       Database::disconnect();
                      ?>
                      </tbody>
                </table>
        </div>
    </div> <!-- /container -->
    </body>
  </html>
```

Now if you navigate to "index.php page". You should notice a "Create" button. However since its linked page is not created, it will redirect to an error page if you click the button. We will fix that next step:



2. Creating a "Create" page

Create a PHP file "create.php"; This file contains an html form and it is the "Create" part of CRUD grid.

We will go through this file as two parts.

First part of the codes is an html form. It is pretty straightforward; all it does is to create an html form. As we need to provide validation for form entries. Hence there is a PHP variable for holding validation error for each field. Copy the code below to "create.php" file:

?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link   href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">

                <div class="span10 offset1">
                    <div class="row">
                        <h3>Create a Customer</h3>
                    </div>

                    <form class="form-horizontal" 
action="create.php" method="post">
                      <div class="control-group <?php echo !
empty($nameError)?'error':'';?>">
                        <label class="control-label">Name</label>
                        <div class="controls">
                            <input name="name" type="text"  
placeholder="Name" value="<?php echo !empty($name)?$name:'';?>">
                            <?php if (!empty($nameError)): ?>
                                <span class="help-inline"><?php 
echo $nameError;?></span>
                            <?php endif; ?>
                        </div>
                      </div>
                      <div class="control-group <?php echo !
empty($emailError)?'error':'';?>">
                        <label class="control-label">Email 
Address</label>
                        <div class="controls">
                            <input name="email" type="text" 
placeholder="Email Address" value="<?php echo !empty($email)?
$email:'';?>">
                            <?php if (!empty($emailError)): ?>
                                <span class="help-inline"><?php 
echo $emailError;?></span>
                            <?php endif;?>
                        </div>
                      </div>
                      <div class="control-group <?php echo !
empty($mobileError)?'error':'';?>">
                        <label class="control-label">Mobile 
Number</label>
                        <div class="controls">
                            <input name="mobile" type="text"  
placeholder="Mobile Number" value="<?php echo !empty($mobile)?
$mobile:'';?>">
                            <?php if (!empty($mobileError)): ?>
                                <span class="help-inline"><?php 
```

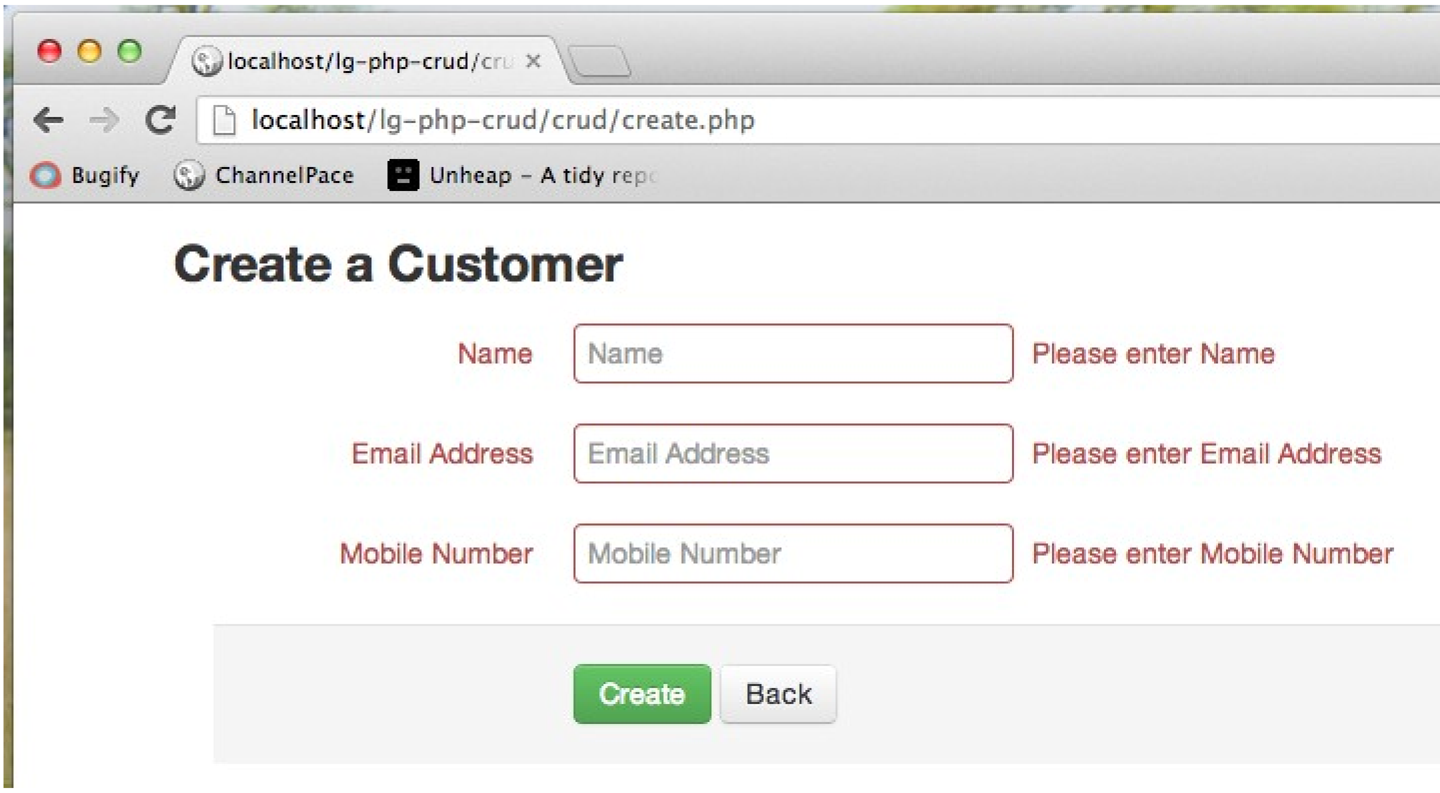
Second part of the codes is where the magic happens. It handles record creation process. Copy the codes below to the beginning of "create.php" file, we will go through them afterwards:

?

```php
<?php

    require 'database.php';

    if ( !empty($_POST)) {
        // keep track validation errors
        $nameError = null;
        $emailError = null;
        $mobileError = null;

        // keep track post values
        $name = $_POST['name'];
        $email = $_POST['email'];
        $mobile = $_POST['mobile'];

        // validate input
        $valid = true;
        if (empty($name)) {
            $nameError = 'Please enter Name';
            $valid = false;
        }

        if (empty($email)) {
            $emailError = 'Please enter Email Address';
            $valid = false;
        } else if ( !filter_var($email,FILTER_VALIDATE_EMAIL) ) {
            $emailError = 'Please enter a valid Email Address';
            $valid = false;
        }

        if (empty($mobile)) {
            $mobileError = 'Please enter Mobile Number';
            $valid = false;
        }

        // insert data
        if ($valid) {
            $pdo = Database::connect();
            $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
            $sql = "INSERT INTO customers (name,email,mobile)
values(?, ?, ?)";
            $q = $pdo->prepare($sql);
            $q->execute(array($name,$email,$mobile));
            Database::disconnect();
            header("Location: index.php");
        }
        }
    ?>
```

As you see, firstly we check if there is form submit by checking $_POST variable. If so, we check each entries to ensure they are not empty. Additionally for email address entry, we use PHP filter to verify if it is a valid email address. Then if it passes all validation rules, it inserts data to database using Database class. At last it will redirect to "index.php" using PHP header() function. However if there is any validation error, the validation variables will be showed in the form.

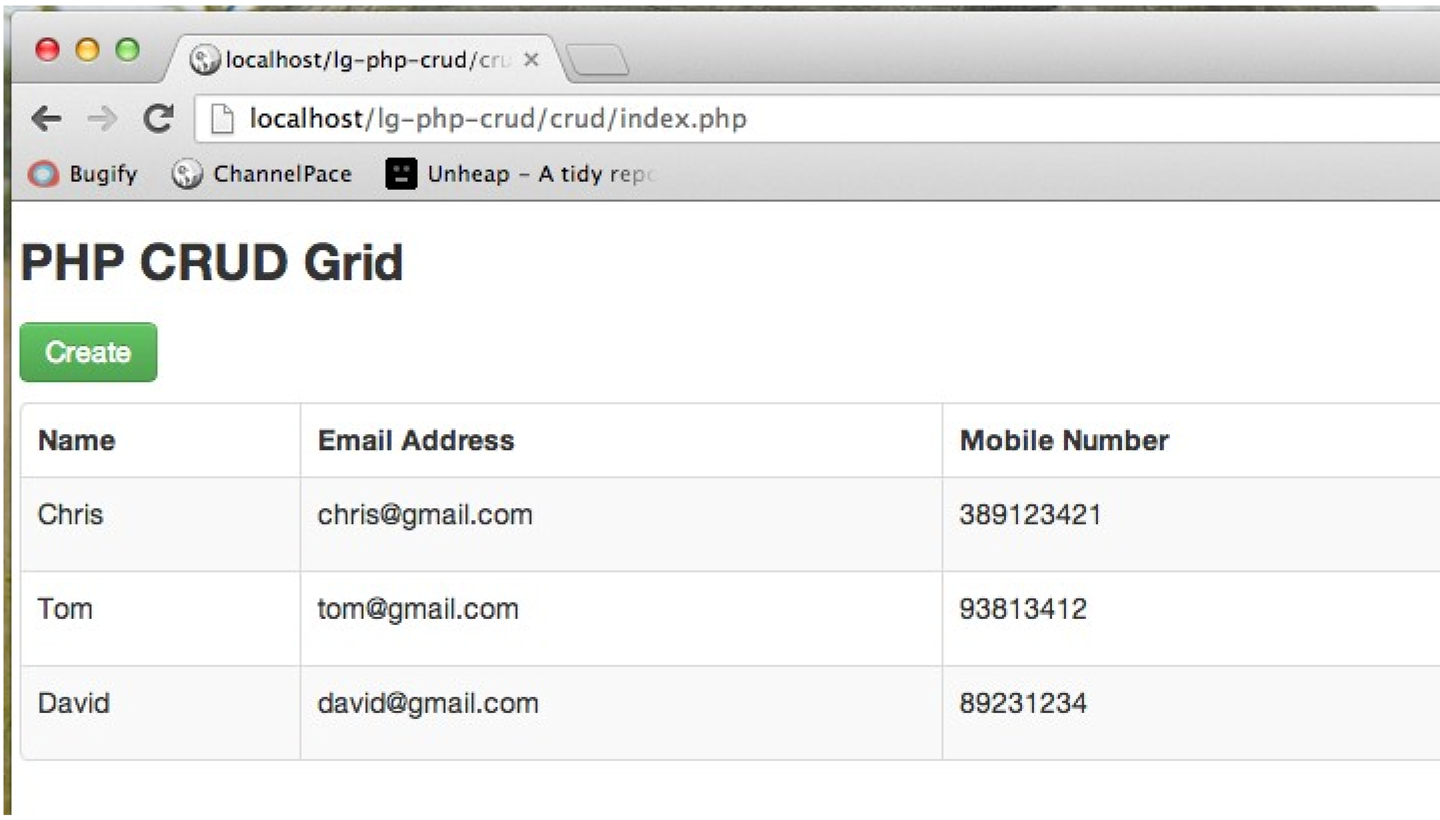
If you have followed correctly. Navigate to "create.php" page, and click on the "Create" button, you should be able to see a PHP form with validation errors as below:


localhost/lg-php-crud/cru
localhost/lg-php-crud/crud/create.php
Bugify
ChannelPace
Unheap – A tidy rep

Create a Customer

Name	Name	Please enter Name
Email Address	Email Address	Please enter Email Address
Mobile Number	Mobile Number	Please enter Mobile Number

Create some records by entering correct customer information. You should be able to see a CRUD grid as below.


localhost/lg-php-crud/cru
localhost/lg-php-crud/crud/index.php
Bugify
ChannelPace
Unheap - A tidy rep
Create

PHP CRUD Grid

Name	Email Address	Mobile Number
Chris	chris@gmail.com	389123421
Tom	tom@gmail.com	93813412
David	david@gmail.com	89231234

You must have noticed a "Read" button for each row. That is what we have added in step 1 of this tutorial. If you click it now, it will lead you to an error page. And that is what we are going to create next step.

3. Creating a "Read" page

Create a PHP file "read.php"; this file is the "Read" part of CRUD grid. Comparing to "Create" part, this step is pretty straightforward. Copy codes below to the file and we will explain afterwards.

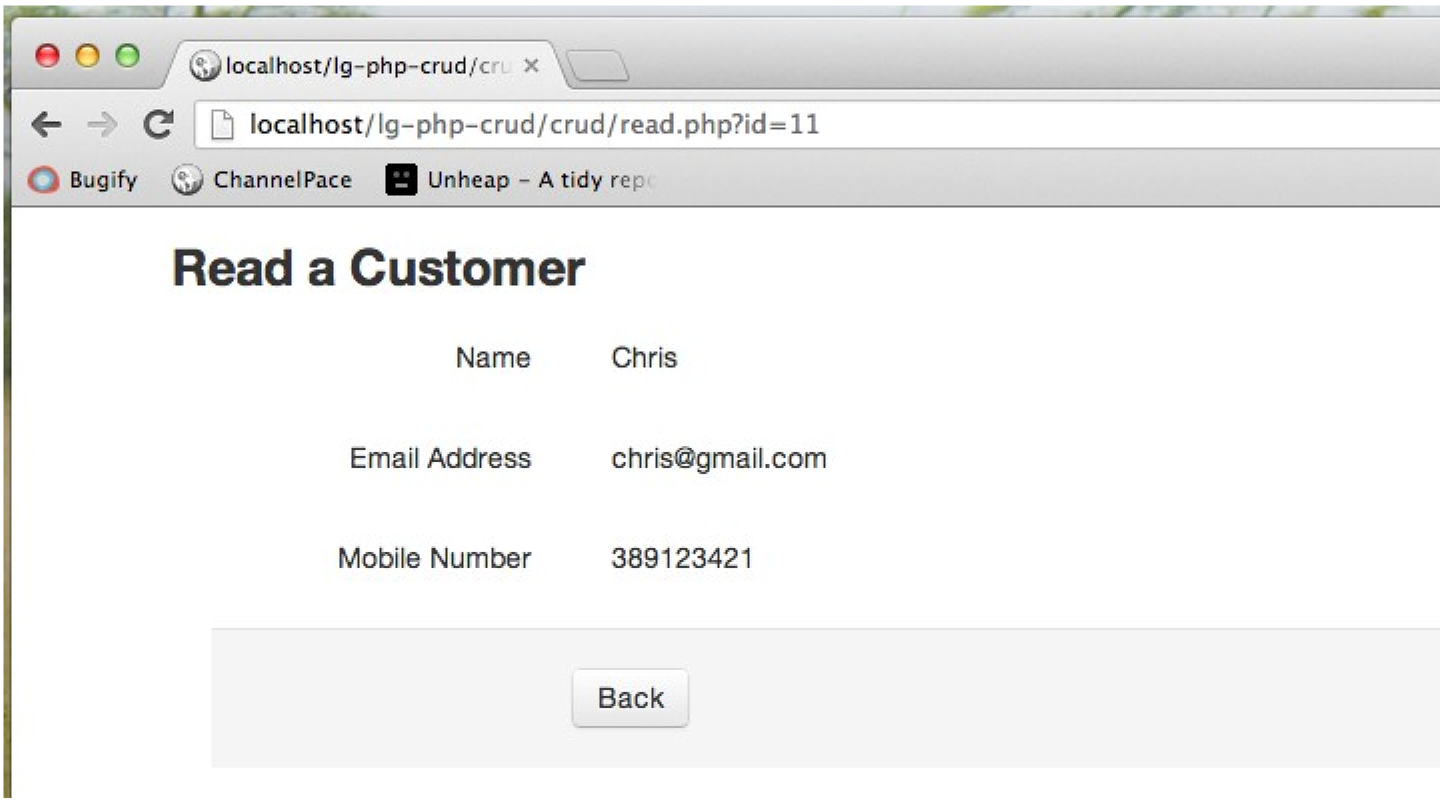
?

```
<?php
    require 'database.php';
    $id = null;
    if ( !empty($_GET['id'])) {
        $id = $_REQUEST['id'];
    }

    if ( null==$id ) {
        header("Location: index.php");
    } else {
        $pdo = Database::connect();
        $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        $sql = "SELECT * FROM customers where id = ?";
        $q = $pdo->prepare($sql);
        $q->execute(array($id));
        $data = $q->fetch(PDO::FETCH_ASSOC);
        Database::disconnect();
    }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link   href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">

                <div class="span10 offset1">
                    <div class="row">
                        <h3>Read a Customer</h3>
                    </div>

                    <div class="form-horizontal" >
                      <div class="control-group">
                        <label class="control-label">Name</label>
                        <div class="controls">
                            <label class="checkbox">
                                <?php echo $data['name'];?>
                            </label>
                        </div>
                      </div>
                      <div class="control-group">
                        <label class="control-label">Email
Address</label>
                        <div class="controls">
                            <label class="checkbox">
                                <?php echo $data['email'];?>
                            </label>
                        </div>
```

Firstly let us look at the beginning part of the PHP codes. What it does is that, it tries to allocate a $_GET['id'] variable. If it is not found, it redirects to "index.php" page. Otherwise, it will read data from database using the "id" field and store data into a PHP variable $data.

Next part, which is the static html part, is pretty simple. It prints out the $data variable.

If you get it working. "read.php" page should look like below:

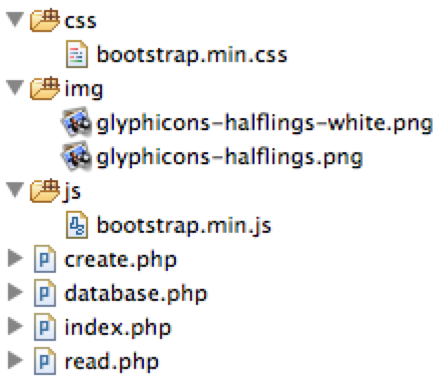

localhost/lg-php-crud/cru
localhost/lg-php-crud/crud/read.php?id=11
Bugify
ChannelPace
Unheap - A tidy rep
Back

Read a Customer

Name	Chris
Email Address	chris@gmail.com
Mobile Number	389123421

4. To be continued

If you have followed along correctly. You should have some files as below:



```
▼ css
    bootstrap.min.css
▼ img
    glyphicons-halflings-white.png
    glyphicons-halflings.png
▼ js
    bootstrap.min.js
▶ create.php
▶ database.php
▶ index.php
▶ read.php
```

In next tutorial (PHP CRUD tutorial part 3), We are going to create the "Update" and "Delete" pages of PHP CRUD grid. Make sure you bookmark this page and come back to visit us next week. We will provide source code package at the last part of this series of tutorial as well. And also if you enjoy this tutorial, please share them using the share buttons on left side of the post.

Hopefully this simple tutorial helped you with your development.
If you like our post, please follow us on Twitter and help spread the word. We need your support to continue.
If you have questions or find our mistakes in above tutorial, do leave a comment below to let us know.

In this part of tutorial, we are going to create "Read" and "Delete" pages of PHP CRUD grid. This will complete CRUD functions of PHP CRUD grid. Process in this post is actually quite similar to part 2.

Table Of Content

1. Adding "Read" button and "Delete" button

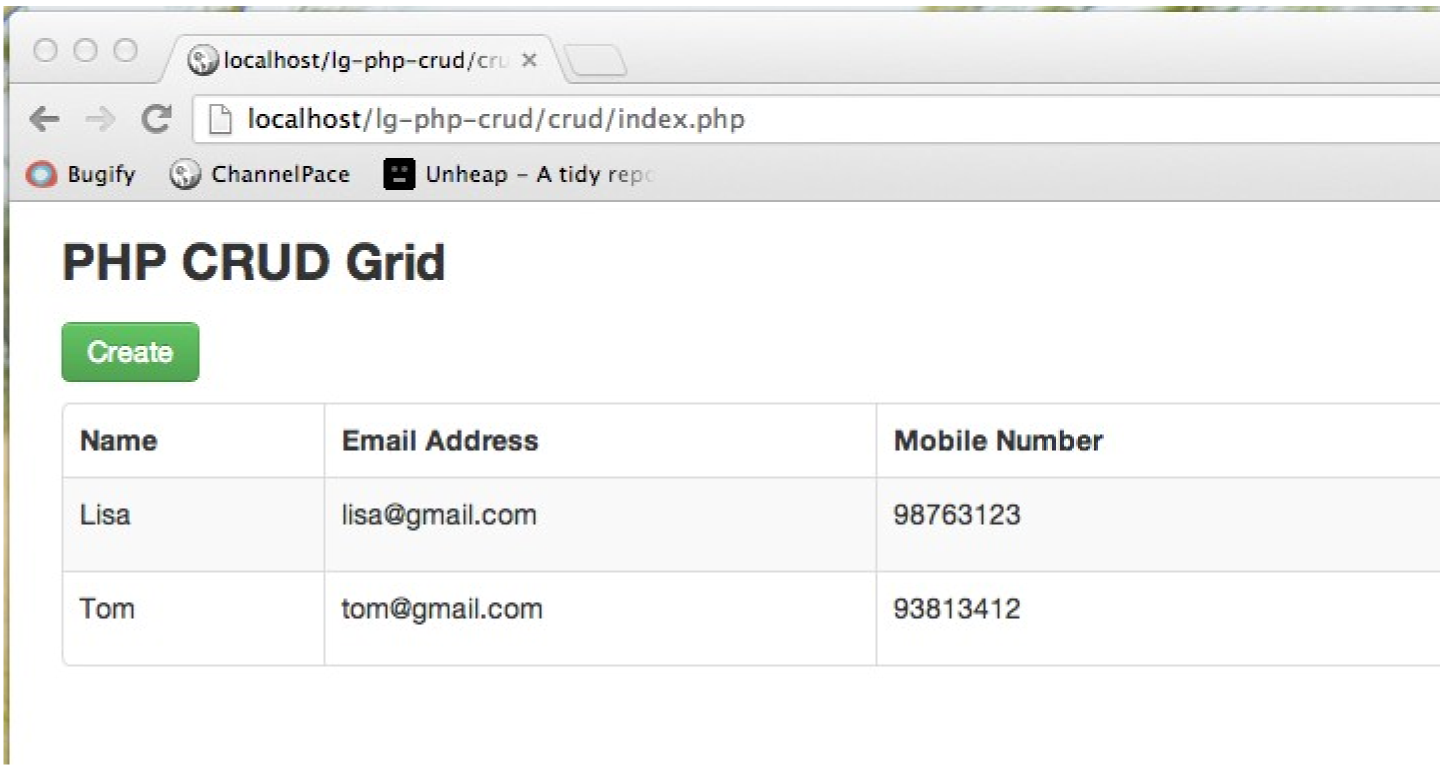
Similar to creating "Create" and "Read" part of PHP CRUD grid. We will firstly need "Update" and "Delete" buttons for each row of the table. Open "index.php" file and add following highlighted codes:

Now the "index.php" file's code should look like below, the highlighted codes are what we have added (or you can just copy all the codes below):

?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link   href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">
            <div class="row">
                <h3>PHP CRUD Grid</h3>
            </div>
            <div class="row">
                <p>
                    <a href="create.php" class="btn btn-
success">Create</a>
                </p>

                <table class="table table-striped table-
bordered">
                      <thead>
                        <tr>
                          <th>Name</th>
                          <th>Email Address</th>
                          <th>Mobile Number</th>
                          <th>Action</th>
                        </tr>
                      </thead>
                      <tbody>
                      <?php
                       include 'database.php';
                       $pdo = Database::connect();
                       $sql = 'SELECT * FROM customers ORDER BY
id DESC';
                       foreach ($pdo->query($sql) as $row) {
                                echo '<tr>';
                                echo '<td>'. $row['name'] .
'</td>';
                                echo '<td>'. $row['email'] .
'</td>';
                                echo '<td>'. $row['mobile'] .
'</td>';
                                echo '<td width=250>';
                                echo '<a class="btn"
href="read.php?id='.$row['id'].'">Read</a>';
                                echo ' ';
                                echo '<a class="btn btn-success"
href="update.php?id='.$row['id'].'">Update</a>';
                                echo ' ';
                                echo '<a class="btn btn-danger"
href="delete.php?id='.$row['id'].'">Delete</a>';
                                echo '</td>';
                                echo '</tr>';
```

Now if you navigate to "index.php page". You should notice "Update" and "Delete" buttons for each row. They are not functional yet. We are going to complete the "Update" page next.



PHP CRUD Grid

Name	Email Address	Mobile Number
Lisa	lisa@gmail.com	98763123
Tom	tom@gmail.com	93813412

2. Creating a "Update" page

Create a PHP file "update.php"; this is the "Update" part of CRUD grid. It is almost identically to "Create" part, besides the fact that we will not only update the record, but also show the data.

We will go through this file as two parts like we did for "Create" page.

First part of the codes is an html form. This form part is exactly the same as "Create" page. Copy the code below to "update.php" file:

?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link   href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">

                <div class="span10 offset1">
                    <div class="row">
                        <h3>Update a Customer</h3>
                    </div>

                    <form class="form-horizontal"
action="update.php?id=<?php echo $id?>" method="post">
                      <div class="control-group <?php echo !
empty($nameError)?'error':'';?>">
                        <label class="control-label">Name</label>
                        <div class="controls">
                            <input name="name" type="text"
placeholder="Name" value="<?php echo !empty($name)?$name:'';?>">
                            <?php if (!empty($nameError)): ?>
                                <span class="help-inline"><?php
echo $nameError;?></span>
                            <?php endif; ?>
                        </div>
                      </div>
                      <div class="control-group <?php echo !
empty($emailError)?'error':'';?>">
                        <label class="control-label">Email
Address</label>
                        <div class="controls">
                            <input name="email" type="text"
placeholder="Email Address" value="<?php echo !empty($email)?
$email:'';?>">
                            <?php if (!empty($emailError)): ?>
                                <span class="help-inline"><?php
echo $emailError;?></span>
                            <?php endif;?>
                        </div>
                      </div>
                      <div class="control-group <?php echo !
empty($mobileError)?'error':'';?>">
                        <label class="control-label">Mobile
Number</label>
                        <div class="controls">
                            <input name="mobile" type="text"
placeholder="Mobile Number" value="<?php echo !empty($mobile)?
$mobile:'';?>">
                            <?php if (!empty($mobileError)): ?>
                                <span class="help-inline"><?php
```

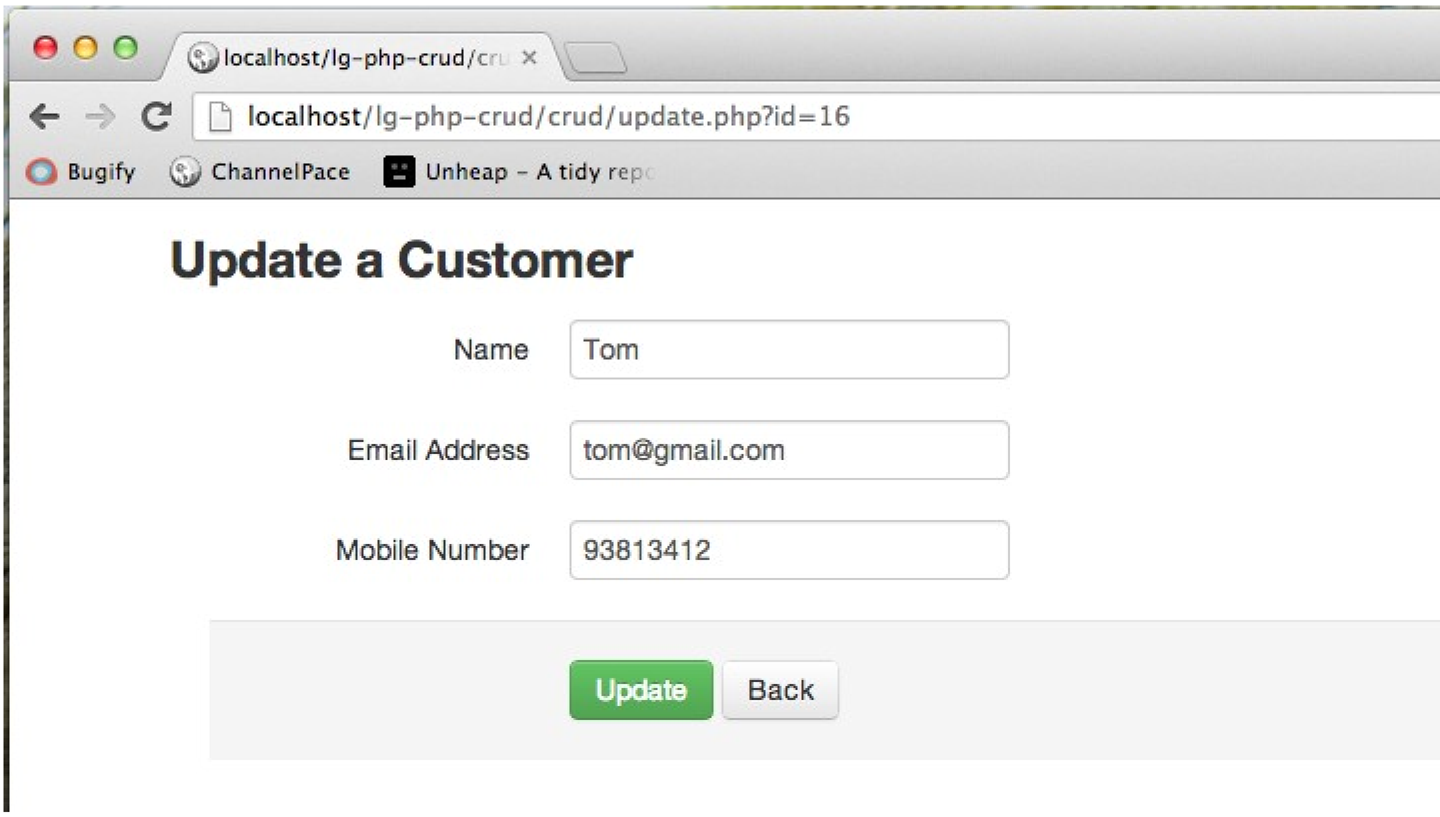
Second part of the codes is where record update process happens. Copy the codes below to the beginning of "update.php" file; we will go through them afterwards:

?

```php
<?php
    require 'database.php';

    $id = null;
    if ( !empty($_GET['id'])) {
        $id = $_REQUEST['id'];
    }

    if ( null==$id ) {
        header("Location: index.php");
    }

    if ( !empty($_POST)) {
        // keep track validation errors
        $nameError = null;
        $emailError = null;
        $mobileError = null;

        // keep track post values
        $name = $_POST['name'];
        $email = $_POST['email'];
        $mobile = $_POST['mobile'];

        // validate input
        $valid = true;
        if (empty($name)) {
            $nameError = 'Please enter Name';
            $valid = false;
        }

        if (empty($email)) {
            $emailError = 'Please enter Email Address';
            $valid = false;
        } else if ( !filter_var($email,FILTER_VALIDATE_EMAIL) ) {
            $emailError = 'Please enter a valid Email Address';
            $valid = false;
        }

        if (empty($mobile)) {
            $mobileError = 'Please enter Mobile Number';
            $valid = false;
        }

        // update data
        if ($valid) {
            $pdo = Database::connect();
            $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
            $sql = "UPDATE customers  set name = ?, email = ?,
mobile =? WHERE id = ?";
            $q = $pdo->prepare($sql);
            $q->execute(array($name,$email,$mobile,$id));
            Database::disconnect();
            header("Location: index.php");
        }
    } else {
        $pdo = Database::connect();
        $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        $sql = "SELECT * FROM customers where id = ?";
        $q = $pdo->prepare($sql);
        $q->execute(array($id));
        $data = $q->fetch(PDO::FETCH_ASSOC);
        $name = $data['name'];
        $email = $data['email'];
        $mobile = $data['mobile'];
        Database::disconnect();
    }
?>
```

As you see, firstly we check if there is form submit by checking $_POST variable. If so, we check each entries to ensure they pass validation rules, after that it updates database using $_POST data. At last it will redirect to "index.php" using PHP header() function. On the flip side, it is a $_GET request, it will retrieve data record from database.

If you have followed correctly. Clicking "Update" button from "index.php" page should lead you to a page similar to below:


localhost/lg-php-crud/crud/update.php?id=16

Update a Customer

Name: Tom

Email Address: tom@gmail.com

Mobile Number: 93813412

3. Creating a "Delete" page

Here comes the final piece. Create a PHP file "delete.php"; Copy codes below to the file and we will explain afterwards.

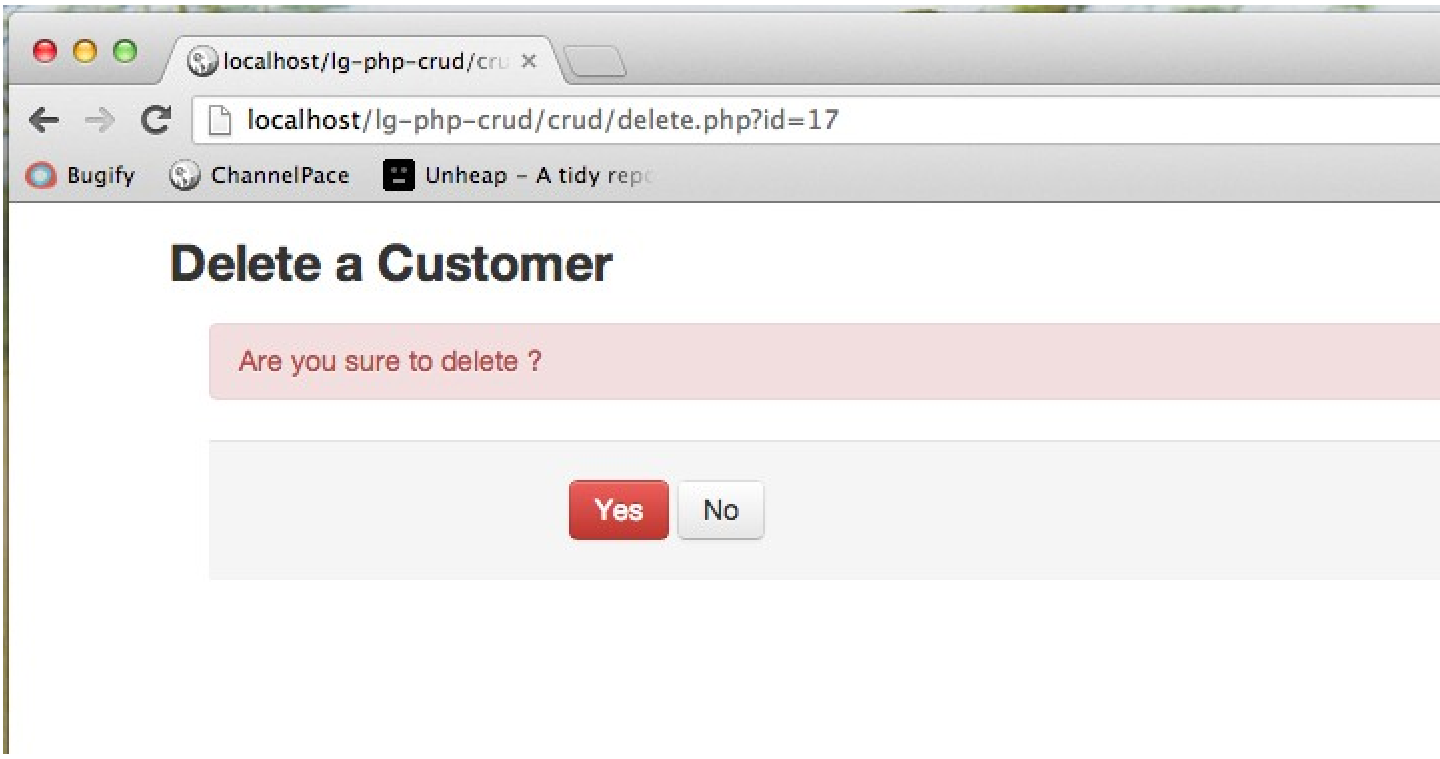
?

```
<?php
    require 'database.php';
    $id = 0;

    if ( !empty($_GET['id'])) {
        $id = $_REQUEST['id'];
    }

    if ( !empty($_POST)) {
        // keep track post values
        $id = $_POST['id'];

        // delete data
        $pdo = Database::connect();
        $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        $sql = "DELETE FROM customers  WHERE id = ?";
        $q = $pdo->prepare($sql);
        $q->execute(array($id));
        Database::disconnect();
        header("Location: index.php");

    }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link   href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/bootstrap.min.js"></script>
</head>

<body>
    <div class="container">

                <div class="span10 offset1">
                    <div class="row">
                        <h3>Delete a Customer</h3>
                    </div>

                    <form class="form-horizontal"
action="delete.php" method="post">
                      <input type="hidden" name="id" value="<?php
echo $id;?>"/>
                      <p class="alert alert-error">Are you sure
to delete ?</p>
                      <div class="form-actions">
                          <button type="submit" class="btn btn-
danger">Yes</button>
                          <a class="btn" href="index.php">No</a>
                        </div>
                    </form>
                </div>
```

Let us look at the beginning part of the PHP codes. It firstly capture the $id from $_GET request. Once a $_GET request is determined. It shows a confirmation page. If a $_POST request is detected, it indicates that user has click confirmation button "Yes". Then it will proceed to delete the data record and redirects to "index.php" page.

Next part, which is the static html part is pretty simple. It simple store the $_GET['id'] to a hidden field.

If you get it working. "delete.php" page should look like below. If you click "Yes" button, it should direct to index page and the selected record should be deleted.



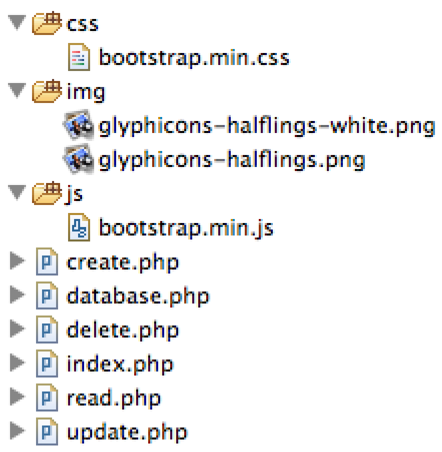
localhost/lg-php-crud/crud/delete.php?id=17

Delete a Customer

Are you sure to delete ?

4. Final and Source Code

If you have followed along correctly. You should have some files as below:



```
▼ css
      bootstrap.min.css
▼ img
      glyphicons-halflings-white.png
      glyphicons-halflings.png
▼ js
      bootstrap.min.js
▶ create.php
▶ database.php
▶ delete.php
▶ index.php
▶ read.php
▶ update.php
```

If you follow along with the tutorials step by step, you will get all the source code in place. However, if you are feeling lazy or have a need to download the complete source code from us. You can do so by paying us a small fee. Your support will enable us to produce better and more in-depth tutorials.

Download Source Code ($9)

Hopefully, this simple tutorial helped you with your development.
If you like our post, please follow us on Twitter and help spread the word. We need your support to continue.
If you have questions or find our mistakes in above tutorial, do leave a comment below to let us know.

https://www.startutorial.com/articles/view/php-crud-tutorial-part-1