# Homework: MySQL and PHP Basics

This document defines the homework assignments for ["PHP Web Development Basic" Course @ Software University](https://softuni.bg/trainings/1746/php-web-developmentbasics-september-2017). Please submit ……………………………………………….???

**Important!** For this exercise you will need to download the ZIP file "**09. PHP-MySQL-and-PHP-Exercise-Databases.zip"** from the <http://softuni.bg/>**.** The file should be on the same page and section (number 20) where you downloaded this exercise (<https://softuni.bg/trainings/1746/php-web-developmentbasics-september-2017#lesson-6663>)

# Part I. Connecting to MySQL from PHP

From PHP 7 the default module to connect to MySQL is PDO (PHP Data Objeccts) or MySQLi. First thing is you should know how to connect to your database from PHP using the MySQLi or PDO driver. In our exercise we will use PDO.

# Connecting through MySQLi

MySQLi is a class available only for MySQL which you may see in PHP code. You can connect the following way:

|  |
| --- |
| $db = new mysqli("127.0.0.1", "my\_user", "my\_password", "my\_db");  // Or  $db = *mysqli\_connect*("127.0.0.1", "my\_user", "my\_password", "my\_db"); |

The function mysqli\_connect() is actually an alias for the constructor of mysqli::\_\_construct(); but it is better to use OOP. You can close the connection with

|  |
| --- |
| mysqli::*close*($db);  //Or  mysqli\_*close*($db); |

For more information on MySQLi visit: <http://php.net/manual/en/book.mysqli.php>

# Connecting through PDO

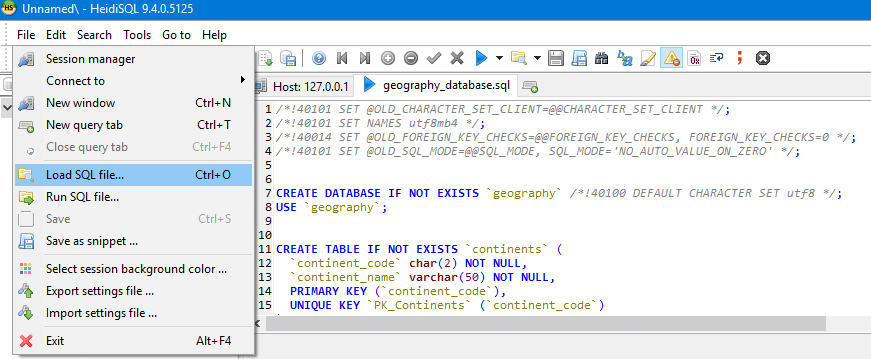
PDO is a universal driver to different databases in PHP. In our exercises we will use the PDO driver because it gives us more functionality. Visit the documentation in: <http://php.net/manual/bg/ref.pdo-mysql.php>. In PDO you connect like this:

|  |
| --- |
| $db = new PDO("mysql:dbname=$db;host=$host", $username, $password); |

# Problem 1. Connect to Geography DB

Create a new project in PhpStrorm and make a new connection to your MySQL server (XAMPP or other).

Use HeidiSQL or another tool to create a new database called **geography**. Import the data from the SQL file **geography\_database.sql** which you can download from the course exercise page.See the screenshot:

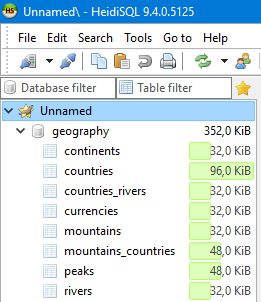
****

Execute the queries (usually F9) and restart HeidiSQL. You should get something as the screenshot on the bo right.

# Connecting through MySQLi\*

In your script first make a new connection using the MySQLi class by making an instance of the class. Test your connection in CLI as given in Part I.

# Connecting through PDO

Go back to PHP. Change the PHP code and connect using PDO. Test your connection in CLI:

|  |
| --- |
| File db\_cofig.php:  <?php $db\_host = "localhost"; $db\_name = "geography"; $db\_user = "someuser"; $db\_password = "yourpass";  *File geography*.*php*: Include("db\_config.php"); $db = new PDO("mysql:dbname=$db\_name;host=$host", $db\_user, $db\_password); |

**Important Note!** It is important to store your database login data in a **separate file**, because it is **specific to your server** and because it **contains your password**. In no case should any unauthorized user be able to see it**! Let the configuration file have the lowest rights possible!**

# Part II. Writing MySQL Queries in PHP

# Problem 2. Simple MySQL Query with PHP

We have created the connection now let's do something practical. Let's get from MySQL the names and codes of all continents. See the example below:

### Example

|  |
| --- |
| **Output** |
| Array  (  [continent\_code] => AF  [0] => AF  [continent\_name] => Africa  [1] => Africa  )  . . .  Array  (  [continent\_code] => SA  [0] => SA  [continent\_name] => South America  [1] => South America  ) | |

To do this we will need the following code:

|  |
| --- |
| $continents = $db->query("SELECT *\** FROM `continents`"); foreach($continents as $i => $continent){  *print\_r*($continent);  echo("\n\r"); } |

Modify your code to show only the names of the continents! How would the select change?

# Problem 3. Wrap PDO in Your Class

To use in a more advanced way PDO it is a good idea to wrap it (extend it) in a class that we will create. We will use the previous code from Geography but we will change it. We will do this to introduce our own methods how to handle errors like this:

|  |
| --- |
| File geography\_db.php:  <?php $db\_host = "localhost"; $db\_name = "geography"; $db\_username = "root"; $db\_password = "pass4mysql";  File mypdo.php:  <?php  class MyPDO extends PDO{  public function setErrorSilent(){  $this->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_SILENT*);  }  public function setErrorException(){  $this->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  } }  File geography.php:  <?php  include "geography\_db.php"; include "mypdo.php"; try {  $db = new MyPDO("mysql:dbname=$db\_name;host=$db\_host", $db\_username, $db\_password);  $db->setErrorException(); *// Throw exception on all errors* $continents = $db->query("SELECT *\** FROM `continents`");  foreach ($continents as $i => $continent) {  *print\_r*($continent);  echo("\n\r");  }  $continents = null; *// Close connection* $db = null; } catch (PDOException $e) {  print "PDO Error: " . $e->getMessage() . "<br/>"; } |

Notice the change in geography.php where we can now set all errors to **throw exceptions**:

$db->setErrorException();

Further we could create our own database connection method to handle tasks specific to our project. With try and catch now you can **handle a specific PDOException** that is thrown.

For about the PDOException handling class see: <http://php.net/manual/en/class.pdoexception.php>

**Discussion.** A very disputed question is how and where should database connection happen and how one would pass the database instance to other code. If you work in a group how would you pass **$db** inside a class /object hierarchy that you use?

# Problem 4. List the Continents

Create a simple PHP script using the previous code which will return a list of all continents separated by commas with their code included in brackets. Use SQL to do that.

|  |
| --- |
| **Output** |
| Africa (AF), Antarctica (AN), Asia (AS), Europe (EU), North America (NA), Oceania (OC), South America (SA) |

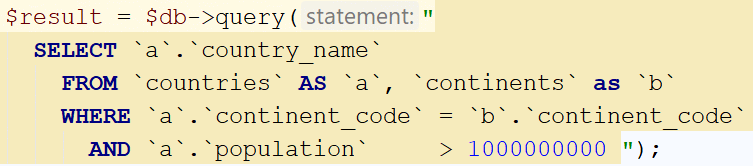
# Problem 5. Highly Populated Countries in Asia

Create a PHP script that will list all countries in Asia with population above 1 billion (1 000 000 000). To do that now join the two tables **continents** and **countries**.

|  |
| --- |
| **Output** |
| China  India |

# Hint

Your query could look like this (in PhpStorm):



Joins will be exercised further in the course. This is a very simple join of the tables **countries** and **continents** which uses their common field **continent\_code.**

# Problem 6. High Peaks in the Andes

A **mountain climber** wants to climb all peaks in the **Andes** mountain (id = 3) which are **higher than 6700 meters**. He needs to start a simple script called **climb.php** to do that. It uses the geography\_db.sql which should be loaded in the MySQL server some way.

|  |
| --- |
| **Output** |
| Aconcagua,6962  Monte Pissis,6793  Ojos del Salado,6893  Cerro Bonete,6759  Mercedario,6720  Pissis,6795 |

# Problem 7. Call Center Application

**John Dew** works at a **call center** for a sports equipment company with **international customers**. For some reason he must know what is the **capital** of the country of his customer to be able to call one or another institution there, and give them the best offer. His customers either gives him the **country name** or **the country code**, or the **ISO code** of country but never three of them. John must get immediately the customer **country and capital** to know which offers are best for him like this:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Palau | Country: Palau  Capital: Melekeok - Palau State Capital |
| Cook Islands | Country: Cook Islands  Capital: Avarua |
| GU | Country: Guam  Capital: Hagåtña |
| TLS | Country: East Timor  Capital: Dili |
| Bye | Good bye! |

The program must run in **CLI mode** all the time end exit when John writes **"Bye"**. It uses the **geography\_db.sql** which should be loaded in the MySQL server some way (see Problem 1)

**Important!** If you use working with strings, you have to know that strings can be multi-byte. Use mb\_ functions if needed. **Match the encoding in PHP with that of the database!**

**Try to use use prepare() and execute() in the places where this is convenient. For more details on the usage visit:**

[**http://php.net/manual/en/mysqli.quickstart.prepared-statements.php**](http://php.net/manual/en/mysqli.quickstart.prepared-statements.php)**,**

[**http://php.net/manual/en/pdo.prepare.php**](http://php.net/manual/en/pdo.prepare.php)**,** [**http://php.net/manual/en/pdostatement.execute.php**](http://php.net/manual/en/pdostatement.execute.php)

# Hint

Where convenient use prepare/execute/fetch that could be something like this:

|  |
| --- |
| $stmt = $db->prepare("  SELECT *\**   FROM `countries`  WHERE `country\_name` = ?"); if ($stmt->execute(array($input))) {  while ($row = $stmt->fetch(PDO::*FETCH\_ASSOC*)) {  *// Todo* } } |

Execute() takes as an input an array of arguments parsed into the prepare() method.

# Add Currency and Continent

Because John is not so good in remembering all, he needs also to know the **local currency** his customer uses and in which **continent** his customer is located.

Try to write the code by using a **join**. Another option to create **two more queries** to the other two tables.

|  |  |
| --- | --- |
| **Input** | **Output** |
| KIR | Country: Kiribati  Capital: Tarawa  Currency: Australia Dollar  Continent: Oceania |
| MV | Country: Maldives  Capital: Malé  Currency: Maldives (Maldive Islands) Rufiyaa  Continent: Asia |

# Hint

KIR is a country ISO (3 characters) and MV is a country code (2 characters). See the database through a graphical client like HeidiSQL to get oriented.

# Customers in the Mountain

A co-worker of John is called **Diana**. But Diana works only with customers that are in a mountain country. Now John wants to forward his calls to those customers to Diana but he **must know if the country is a mountain country**. If it is the application should them him.

|  |  |
| --- | --- |
| **Input** | **Output** |
| IT | Country: Italy  Capital: Rome  Currency: Euro  Continent: Europe  Forward customer for special offers! |

# Hint

You should create a query to the table mountains\_countries . If the country is included, then the application should print "Forward customer for special offers!".

# Special Ski Equipment

If the country is in a region where there is a peak **higher than 4000 meters** Diana will offer some **special equipment** to those customers. You have to extend the application to check all peaks in the country and if the condition is met then it should print "Show high mountain special equipment offers!"

|  |  |
| --- | --- |
| **Input** | **Output** |
| PG | Country: Papua New Guinea  Capital: Port Moresby  Currency: Papua New Guinea Kina  Continent: Oceania  Forward customer for special offers! Show high mountain special equipment offers! |

# Problem 8. Call Center Application in OOP

**Wrap** your Call center application in a **single class** called **CallCenter**. Rewrite your code to work only with one **single instance of CallCenter** with the name of **$app** like this:

|  |
| --- |
| $app = new CallCenter(); // $app contains all functionality! |

Separate your project in the following files: **CallCenter.php** (CallCenter class), **Database.php** (for the Database class) and **callcenter\_app.php** for the initial code.

# 8.1. Create Database Class

Rename MyPDO from Problem 3 to Database and create a method that creates a **single instance of the Database** class called connect() and returns it when it is called. The Database class **should check whether there is an instance and return it if it is connected**. If convenient use a static variable.

Put all database configuration in your new class and use class properties instead of simple variables. Put this in a separate file **Database.php**

# 8.1. Connect to the Database in CallCenter

In your CallCenter class create a new method that connects you to the database by using the Database class.

# Problem 9. Add Customer Functionality

Write a method for CallCenter class from Problem 8 (Call Center App in OOP) which will add customer **id**, **customer\_name**, **customer\_family** and **country\_code** in a new table called **customer.** Name the method addCustomer(). Use a MySQL insert statement.

Change the input like this: (country code/name/ISO, first name, last name. See now the input/output:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Customer KIR, Taraba, Garabe | Country: Kiribati  Capital: Tarawa  Currency: Australia Dollar  Continent: Oceania  Name: Taraba  Garabe: Garabe |
| Customer Bulgarrrria, Georgi,Ivanov | Exception: Country doesn't exist. |

**Do not add** a new customer **if he already exists!** AddCustomer()should check for this. Use an SQL query to check this. If the country is not found then throw an Exception " Country doesn't exist."

# Hint

The **customer** table is **related to countries** by **country\_code.**

# Input: Add Data

**From CLI add** the following **customers** which will be all in **Bulgaria**:

|  |
| --- |
| **Input** |
| Customer Bulgaria, Ivan, Ivanov  Customer Bulgaria, Boyan, Stoyanov  Customer Bulgaria, Stoyan, Ganchev  Customer Bulgaria, Божидар, Божидаров |

# Problem 10. Delete Customer Functionality

In the call center they need now to be able to **delete a customer**. This is rarely done but it is needed. Especially it is needed when someone dies

# Input: Delete Data

**From CLI delete** the following **customer:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Remove, Stoyan, Ganchev | Customer Stoyan Ganchev removed. |

# Problem 11. Customers in Specific Country\*

At the end of his working day John wants to know the **names of his customers in a specific country**. Create a method getCustomersInCountry( $country\_name ) which prints a list of all customers in a specific country. The input and output is as follows:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Customers in Bulgaria? | Customers in Bulgaria:  Ivan Ivanov  Boyan Stoyanov  Божидар Божидаров |

# Problem 12. Customers in Specific Continent\*\*

At the end of his working day John wants to know the **names of his customers in a specific continent**. Create a method getCustomersInContinent( $country\_name ) which prints a list of all customers in a specific continent. The input and output is as follows:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Customers in continent Europe? | Customers in Bulgaria:  Ivan Ivanov  Boyan Stoyanov  Божидар Божидаров |

Doing this for Europe should for now display all customers in Bulgaria.

# Hint

You may use an SQL join because you need **continents**, **countires** and **customers** tables. You know the continent, you must get the countries and then the customers in the continent. Try to join three of them.