

Database Connectivity

Objectives:

The goal of this lab is to get familiar with

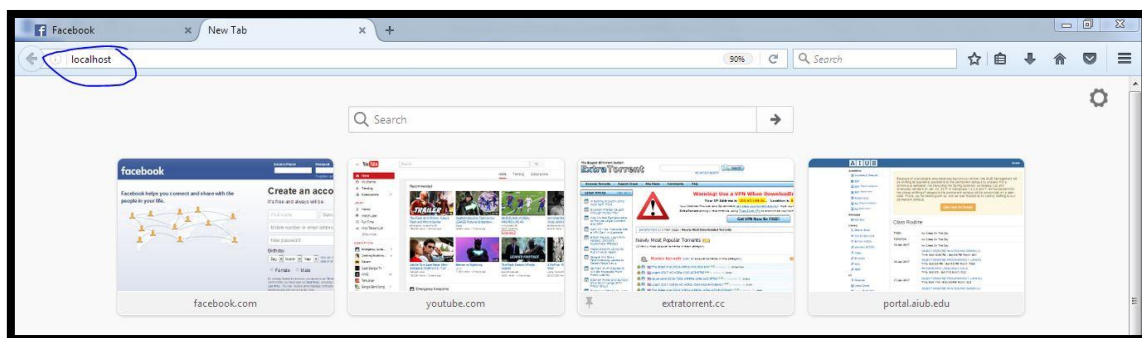
- Connecting MySQL database

Create a Database

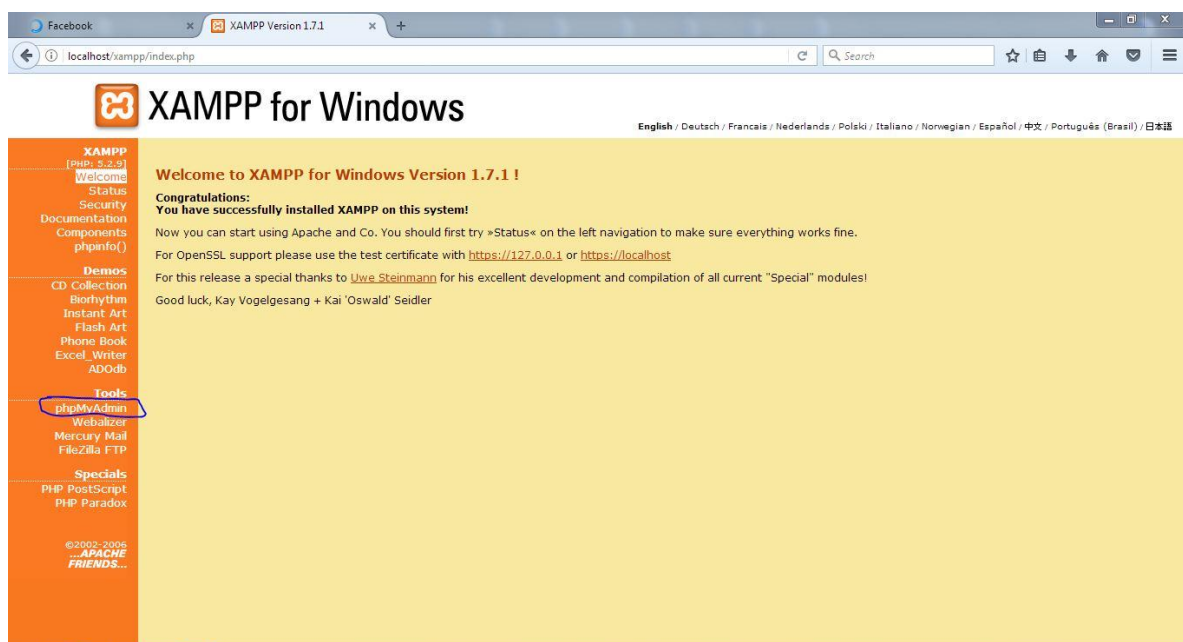
We need to install XAMPP. It is a free and open source cross-platform web server solution stack package. It also gives us an interface to create and manipulate database. Following link can be used to install Xampp:

goo.gl/uLrW3h

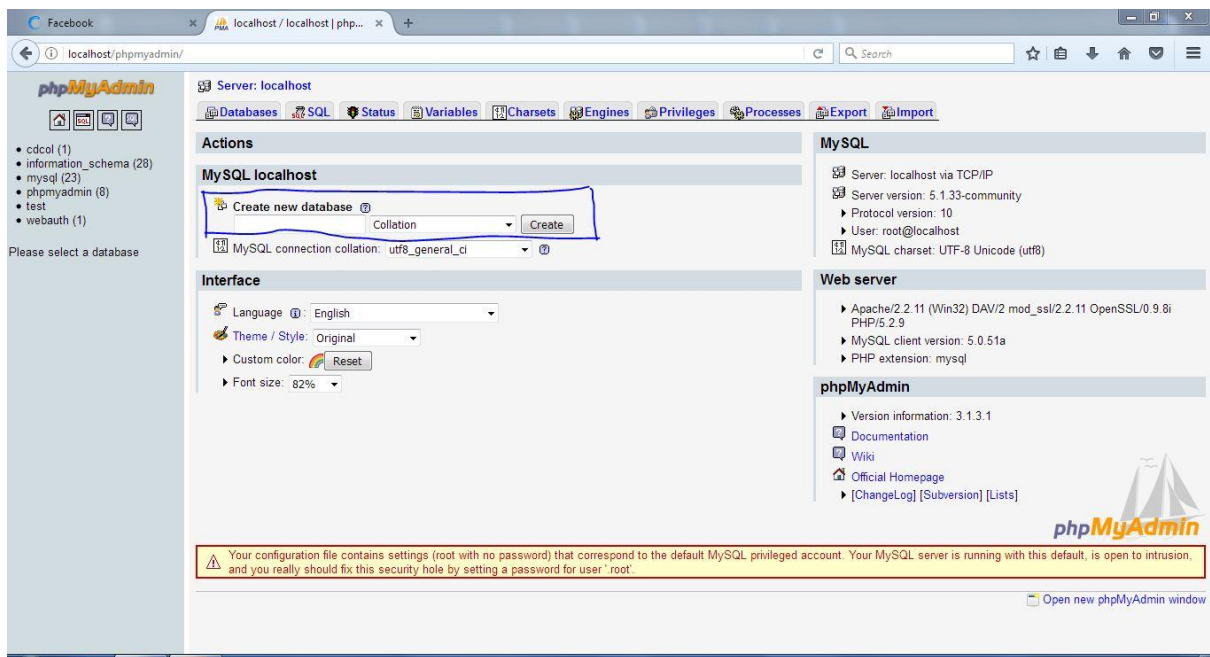
Once you have installed Xampp in your computer, open a browser and type localhost in the address bar.



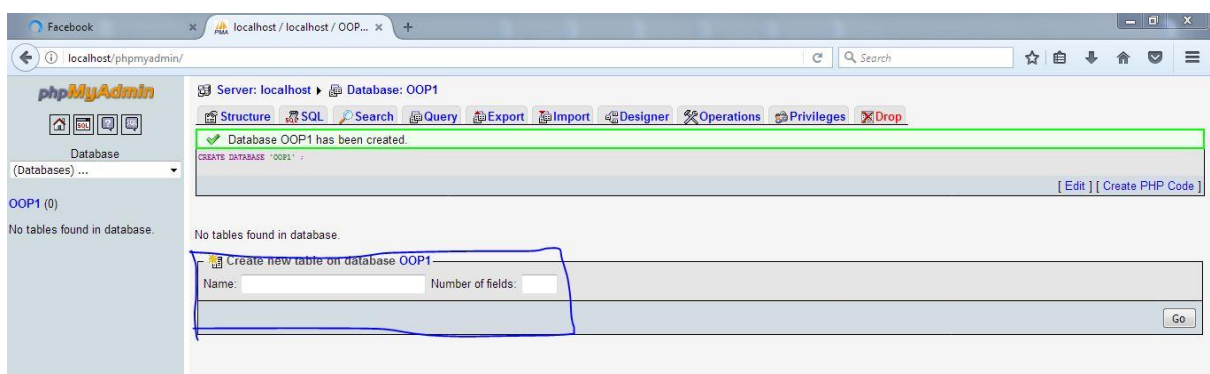
Press the enter key. Select a Language. Now click on phpMyAdmin.



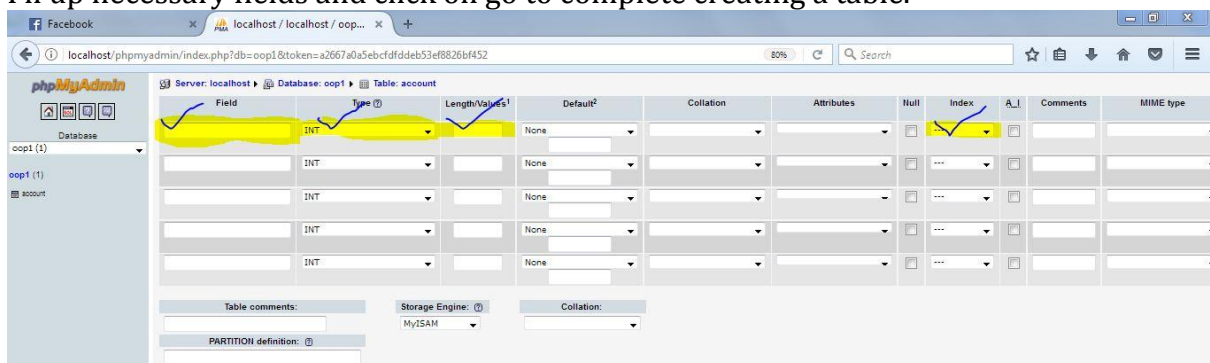
Create a new database. Let the name of database be OOP1.



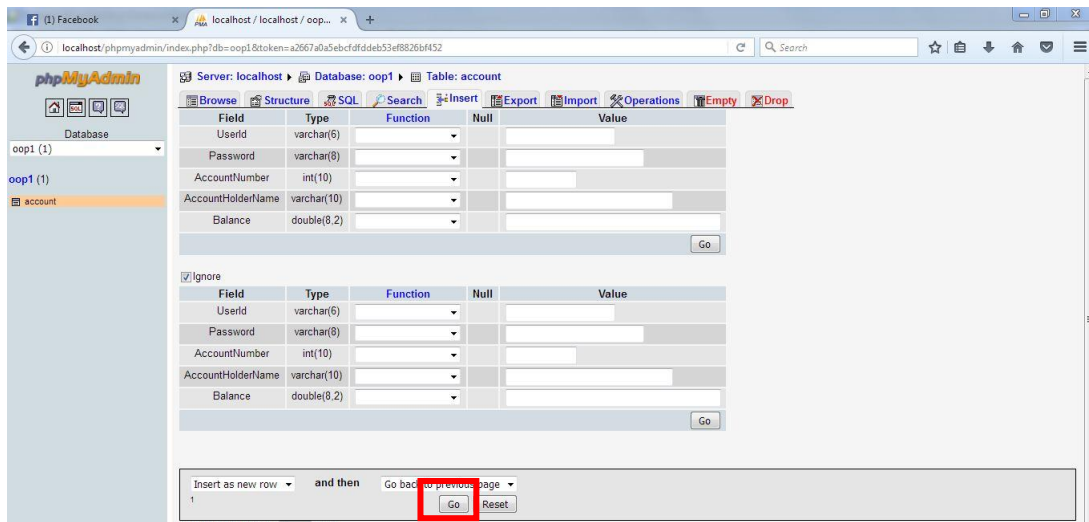
Now, create a table. Let the table name be account and number of fields be 5 and click on the go button.



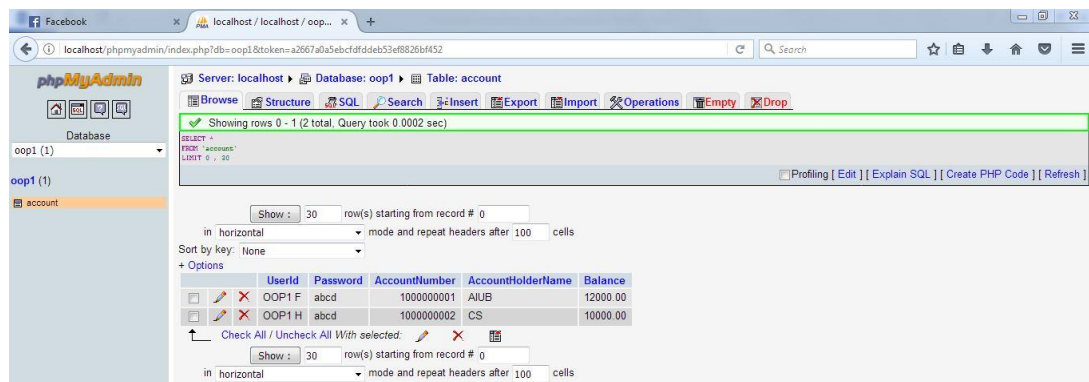
Fil up necessary fields and click on go to complete creating a table.



Now, click on insert and fill up the fields and click on go to insert two rows of data.



You have completed creating a database.



Java Database Connectivity

Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection.

Required .jar File

To perform database connection you would require mysql-connector-java-5.1.38-bin.jar file. It can be downloaded from this link:

<https://dev.mysql.com/downloads/connector/j/>.

After downloading copy the .jar file to JDK/JRE/lib/ext and JRE/lib/ext.

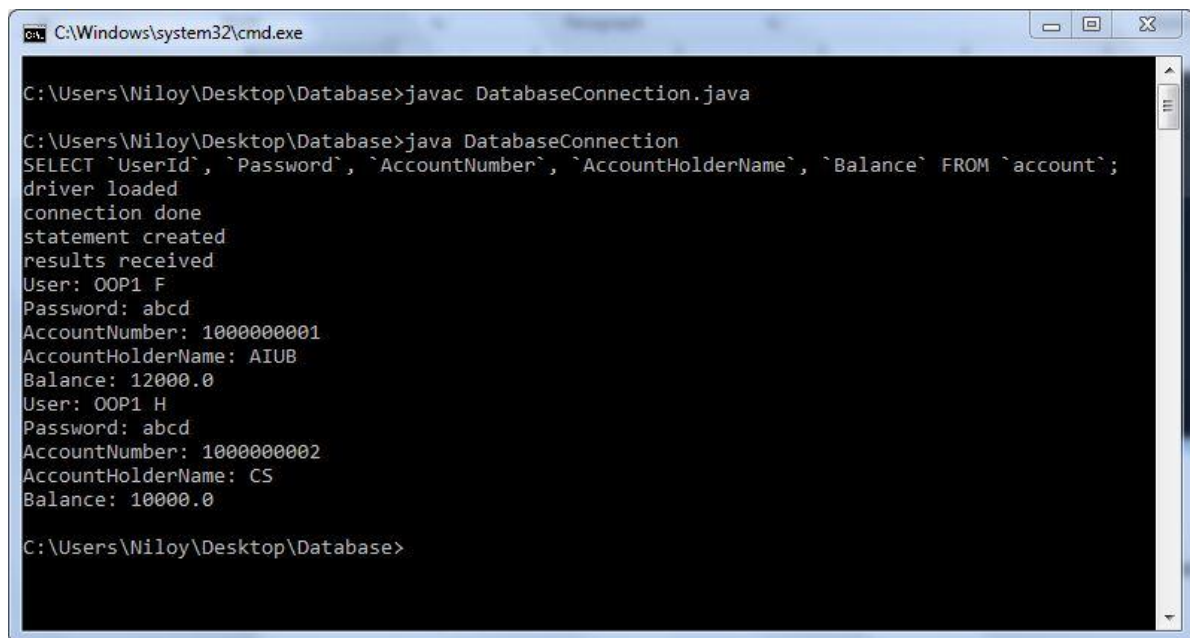
Package java.sql Description

Provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java™ programming language. This API includes a framework whereby different drivers can be installed dynamically to access different data sources. Although the JDBC™ API is mainly geared to passing SQL statements to a database, it provides for reading and writing data from any data source with a tabular format. The reader/writer facility, available through the `javax.sql.RowSet` group of interfaces, can be customized to use and update data from a spread sheet, flat file, or any other tabular data source.

ExampleProgram

As an example, we will create a class that will establish connection with database and read the information from a table of that database.

An example program is given in the `DatabaseConnection.java` file.
This will be the output of this Sample Program:



```
C:\Windows\system32\cmd.exe

C:\Users\Niloy\Desktop\Database>javac DatabaseConnection.java

C:\Users\Niloy\Desktop\Database>java DatabaseConnection
SELECT `UserId`, `Password`, `AccountNumber`, `AccountHolderName`, `Balance` FROM `account`;
driver loaded
connection done
statement created
results received
User: OOP1 F
Password: abcd
AccountNumber: 1000000001
AccountHolderName: AIUB
Balance: 12000.0
User: OOP1 H
Password: abcd
AccountNumber: 1000000002
AccountHolderName: CS
Balance: 10000.0

C:\Users\Niloy\Desktop\Database>
```

Look Carefully at line 20

```
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/oop1","root","");
```

Here,

localhost is your local server.

3306 is the port number that the server uses

“oop1” is the name of the Database

“root” is the username

“” is the password

Port Number

A **port** is identified for each **address** and protocol by a 16-bit number, commonly known as the **port** number. Specific **port** numbers are often used to identify specific services.

Look Carefully at line 27-39

```
while(rs.next())
{
    String userId = rs.getString("UserId");
    String password = rs.getString("Password");
    int accountNumber = rs.getInt("AccountNumber");
    String accountHolderName = rs.getString("AccountHolderName");
    double balance = rs.getDouble("Balance");
    System.out.println("User: " +userId);
    System.out.println("Password: " +password);
    System.out.println("AccountNumber: " +accountNumber);
    System.out.println("AccountHolderName: " +accountHolderName);
    System.out.println("Balance: " +balance);
}
```

| Name of Method | Usage |
|-----------------------------------|---|
| rs.next() | Returns true if there are rows |
| rs.getString("UserId") | Returns value as a string where the column name is UserId |
| rs.getString("Password") | Returns value as a string where the column name is Password |
| rs.getInt("AccountNumber") | Returns value as a integer where the column name is AccountNumber |
| rs.getDouble("Balance") | Returns value as a double where the column name is Balance |