Experiment: 1

INSTALLATION OF ECLIPSE

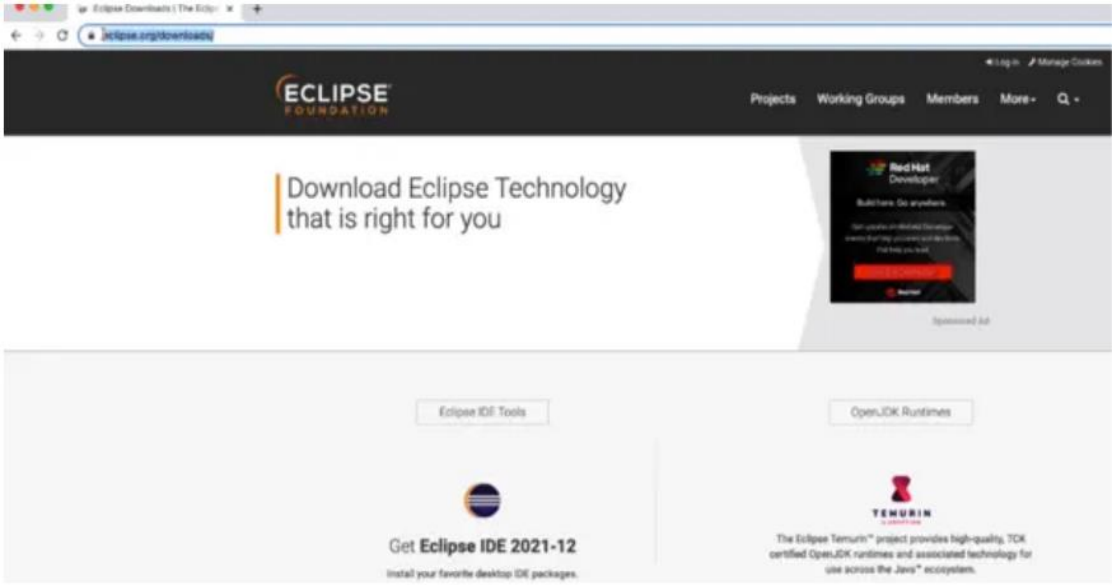
Eclipse IDE is one of the most popular integrated development environments (IDEs). It is mostly used by Java developers, but it can also support almost every other programming language like C/C++, PHP, Scala, Groovy, Clojure, and many more. It has a rich set of features that include support for various programming languages, code analysis, graphical debugging, and unit testing.

In this shot, we will look at the installation process of Eclipse IDE.

Installation steps:

Step 1 - Download the Eclipse installer

The easiest way to install eclipse IDE is to download and run the installer. To download the Eclipse installer, go to this url and select the installer executable as per your operating system platform.

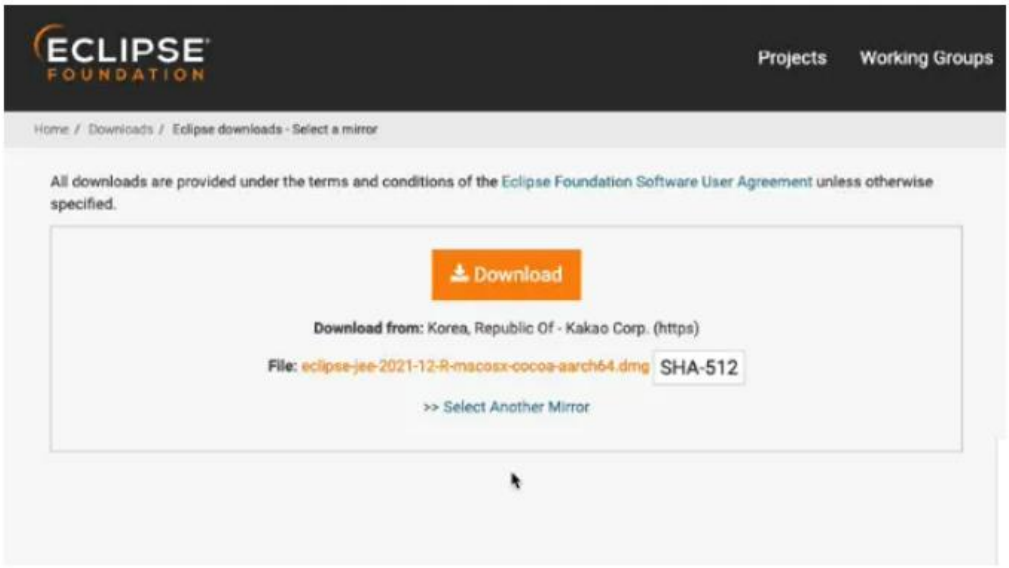


Then, select the closest mirror site to download the required package.

For Windows users, select the folder, e.g., C:/Users/[username]/Downloads to download the installer.

For Mac users, select the folder /Users/[username]/Downloads.

Here, "[username]" represents the username of the operating system you are using.

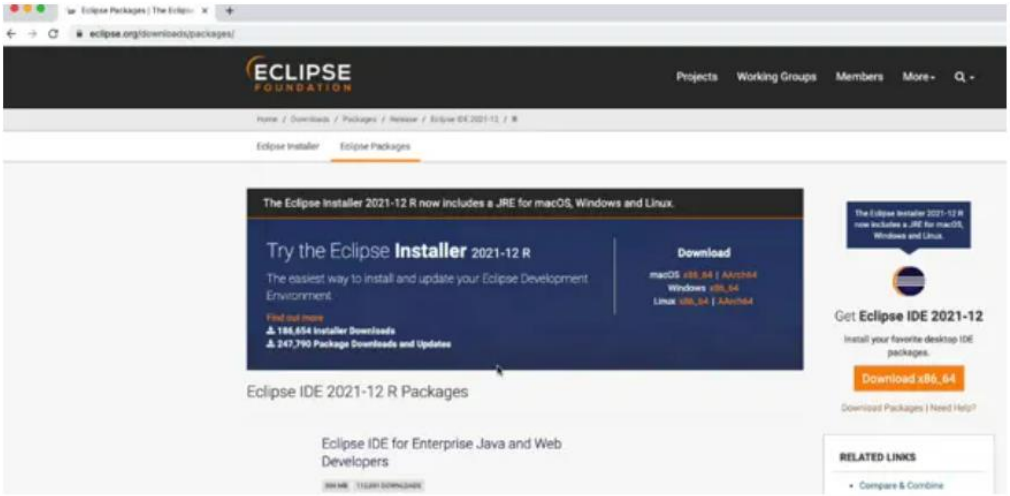


Choosing the closest download mirror site

Step 2 - Running the installer

After the download is completed, execute the Eclipse installer. You may need to extract the content if you use Mac or Linux.

You would also get a security warning for executing a file downloaded from the Internet. Verify if the publisher is the Eclipse Foundation and choose to continue with the file's execution as you can trust this source.



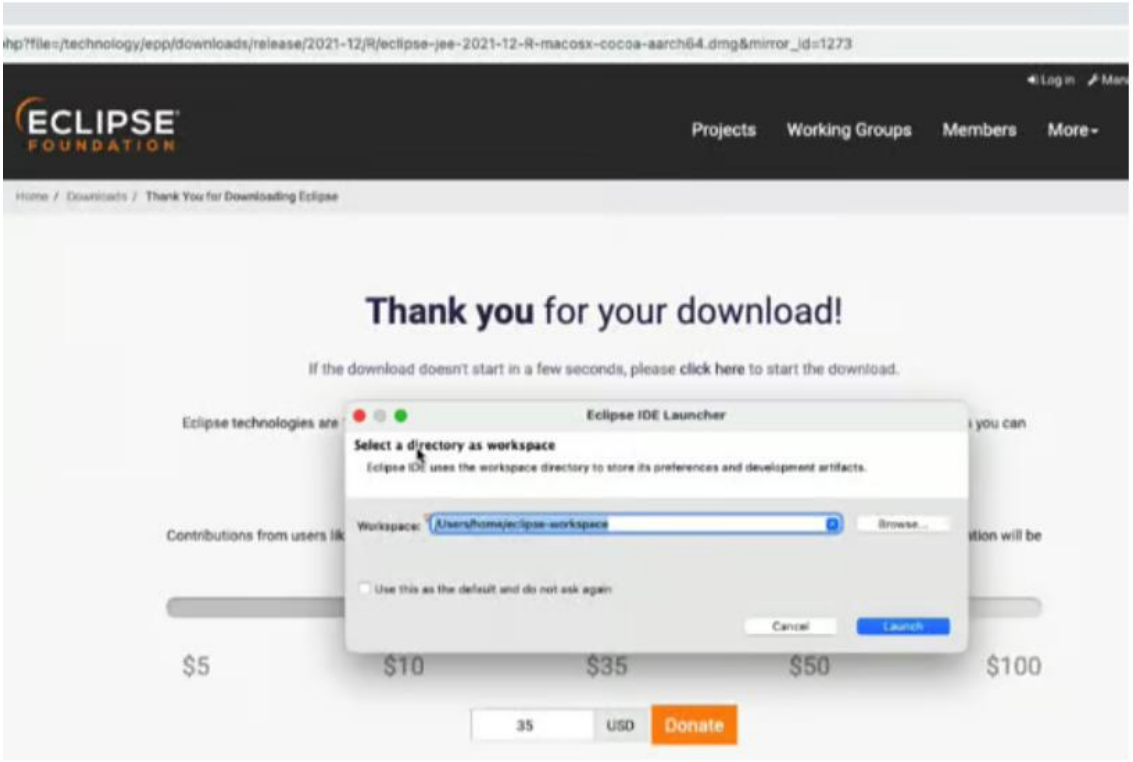
Eclipse IDE package download

Step 3 - Choosing the installation directory/folder

When the installer prompts you to choose the installation location, keep this as your default user directory and begin the installation process. Windows users can also create start menu entries and desktop shortcuts.

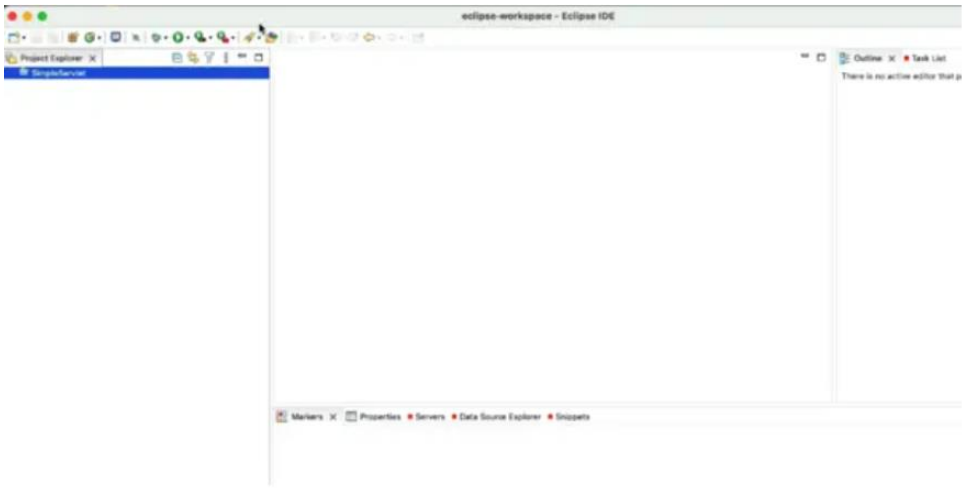
Step 4 - Launching the Eclipse IDE

After the installation is complete, go to the local folder on your system where you installed the application in the previous step. Find the executable file for the Eclipse IDE and run it to launch the IDE.



Selecting workspace directory

During the first run, Eclipse asks users to specify a folder to create the workspace for their projects. After this is provided, the Eclipse IDE gets launched. This completes the installation process.

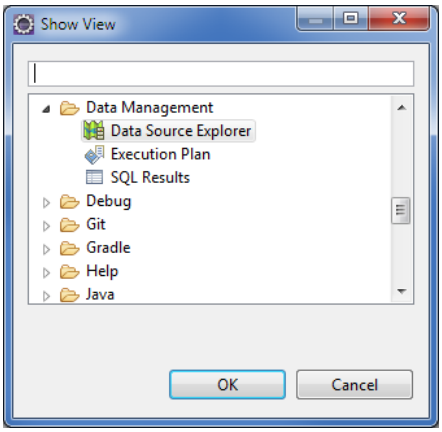


Eclipse IDE

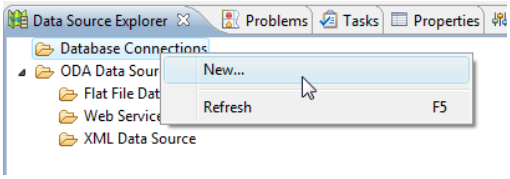
Now you can enjoy coding using the Eclipse IDE!

SETUP DATABASE CONNECTION IN MYSQL

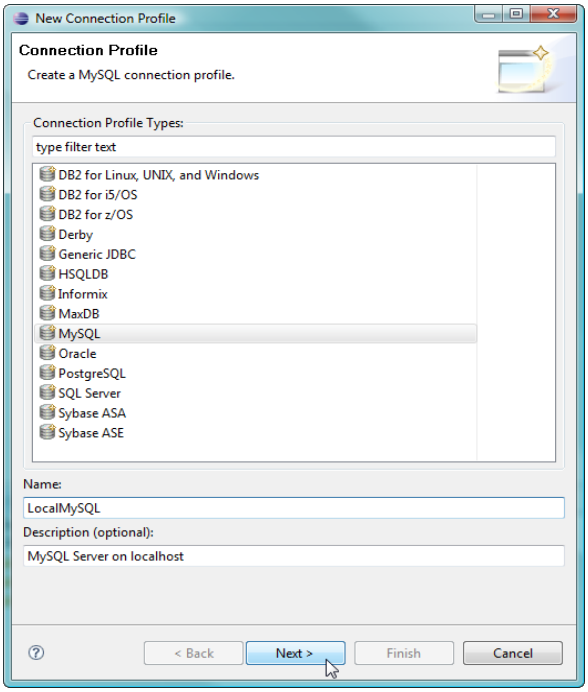
Open Data Source Explorer via [Window]/ [Show View]/ [Other...]/ [Data Management/ [Data Source Explorer]:



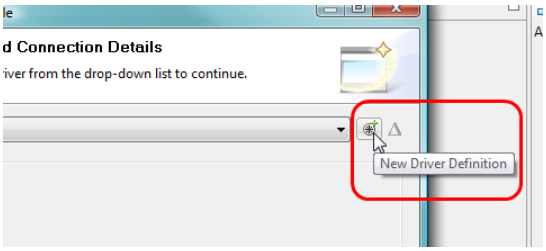
Right click the Database Connection folder, select [New...]



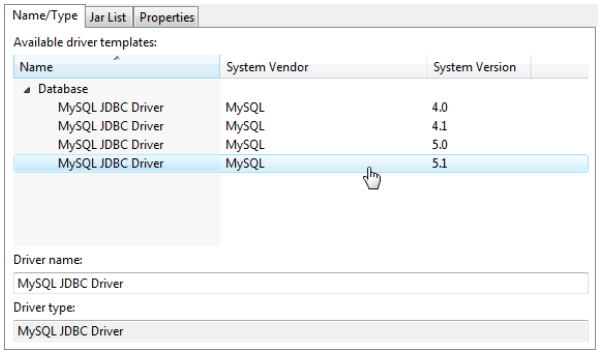
Select the Database you suppose to connect (MySQL 5.1 as the example in this document), you can type custom name and description for mnemonic, click **Next.**



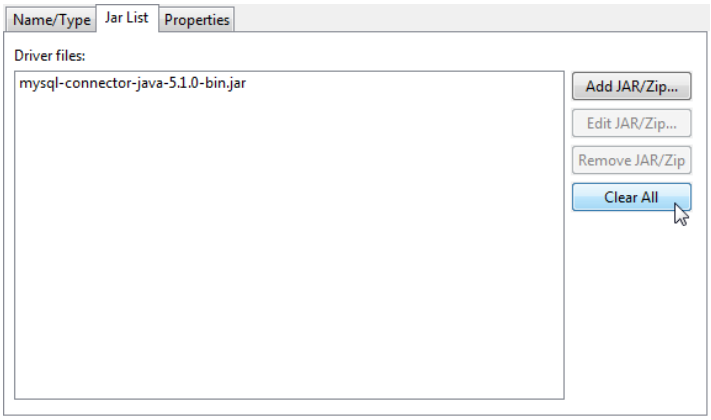
On next page, click the **New Driver** Definition icon next to the Drivers: list box.



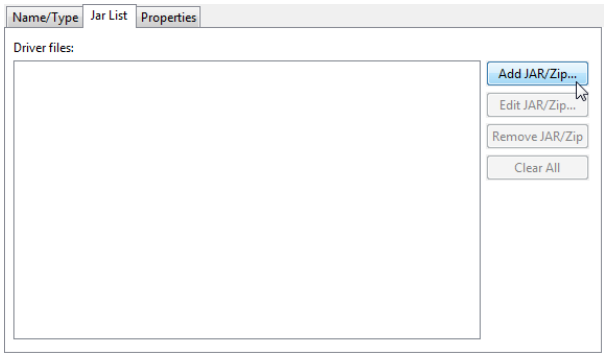
On the popup window, select the Database Version in Name/Type tab.



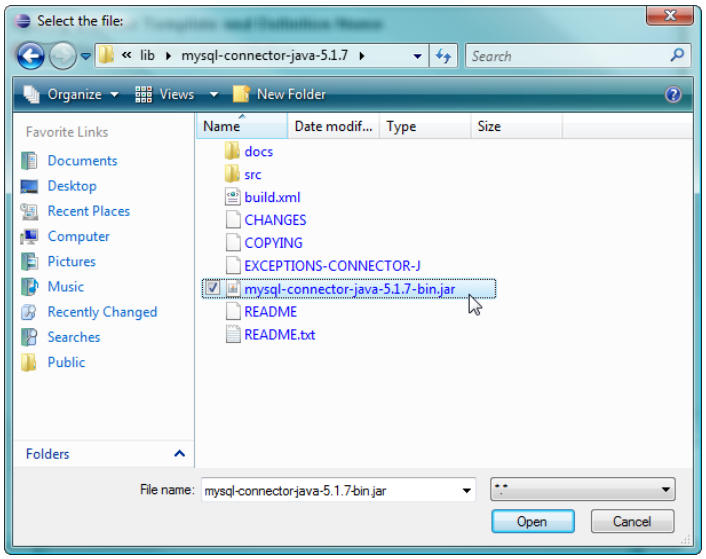
Switch to Jar List tab, click **Clear All** to clear the default false jar file location.



then click **Add JAR/Zp...**.



Locate the jar file inside the folder that just extracted from downloaded MySQL Connector/J zip file.



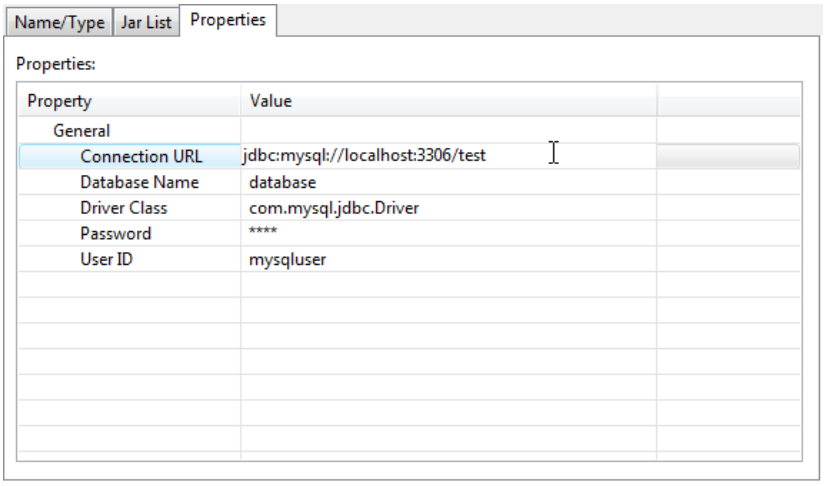
Switch to '"Properties" tab, modify the database connection setting according to your MySQL Database configuration, then click **OK**.

'Note: The JDBC URL format for MySQL Connector/J is as follows, with items in square brackets ([,]) being optional:

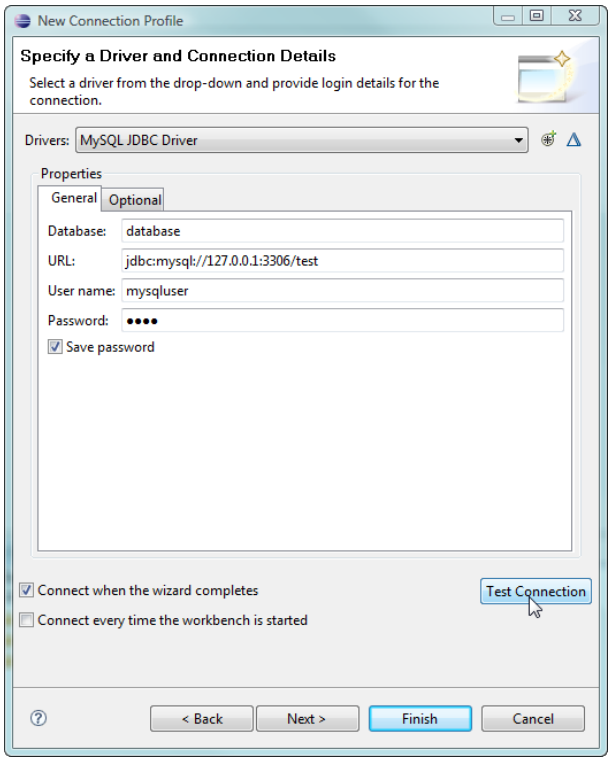
jdbc:mysql://[host:port],[host:port].../[database][?propertyName1][=propertyValue1][&propertyName2][=propertyValue2]...

If the hostname is not specified, it defaults to 127.0.0.1. If the port is not specified, it defaults to 3306, the default port number for MySQL servers. In above image the database name is test.

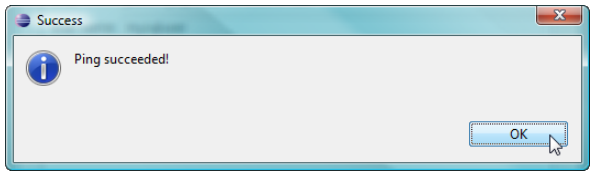
jdbc:mysql://localhost:3306/test



To verify if the setup is functional, click **Test Connection**.

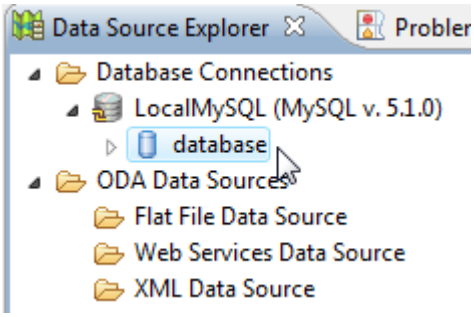


It should popup a Ping succeeded message, Click OK to continue.

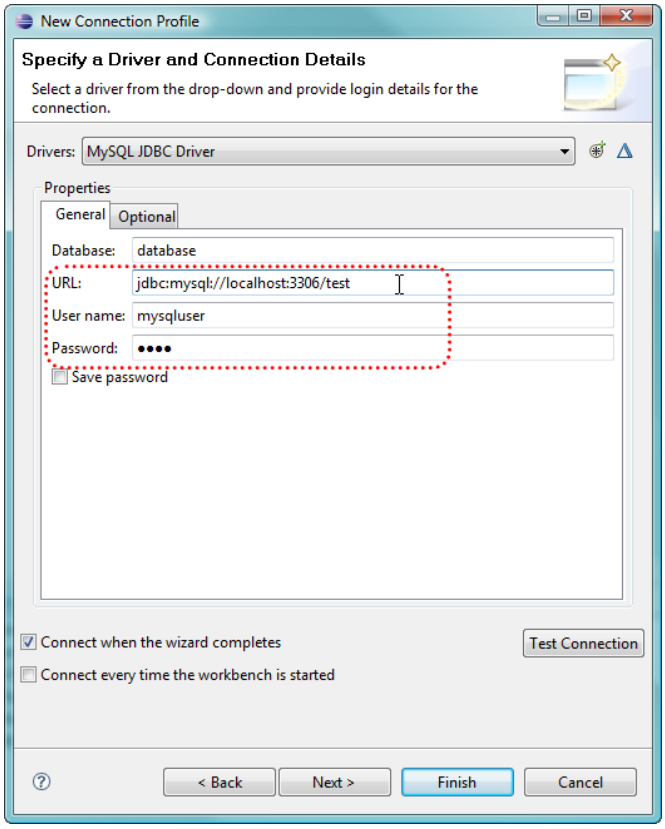


Note: Make sure the MySQL database is up and running. If it is not running, the Test Connection will fail. If you don't know how to find if MySQL is running then please google search it

Press Finish to close the setup wizard window, and you should be able to see there's a Database icon in the Database Connections folder.



Note: Firewall of the System that install Database or security setting of the Database may prevent you from connect the DB successfully. If you had specified the right MySQL Connector/J jar file before, then Step 4 & 5 can be skipped, you can directly modify the following field to match your MySQL Database configuration:

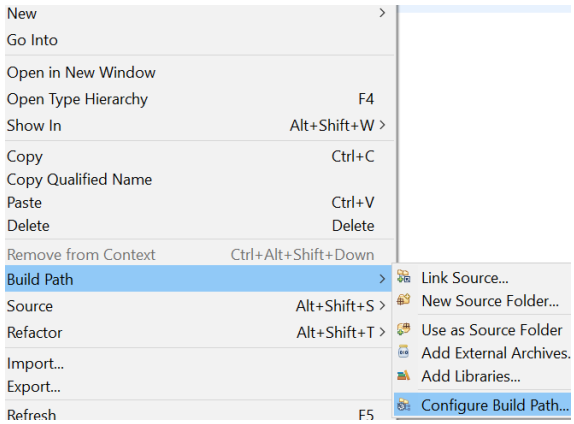


then proceed to the subsequent steps to finish setup.

CONFIGURATION OF MYSQL CONNECTION FROM JAVA ECLIPSE

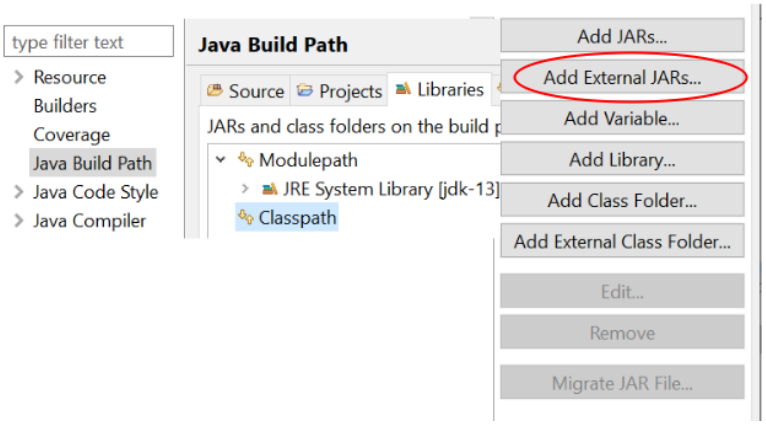
Download mysql-connector.jar file.

Open Package Explorer, select your Java Project, right click and go to Build Path Then select Configure Build Path.

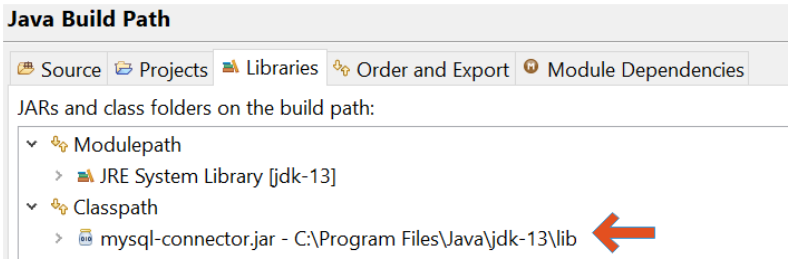


You will get the window for Build Path. Ensure that Class path is selected (Don't select Module path). In this window right side, you will get a button saying Add External Jars.

Point to your downloaded jar file to upload or add. Finally in your Build Path window click Apply & Close



If you reopen the Build Path window you should see your connection jar file like this.



Experiment: 2

MYSQL INSTALLATION

MySQL is one of the most popular database management systems (DBMS) available. It's light, open-source, and easy to install and use, which makes it a good choice for those starting to learn and work with relational databases.

Although it's a good option for beginners, MySQL is also robust enough to support larger production applications, with multi-users and multithreading support.

In this article, we'll cover the process of installing MySQL on your local Windows machine, from downloading to creating and using your first database. When you're done, you'll have a completely functional MySQL server running and ready to use locally.

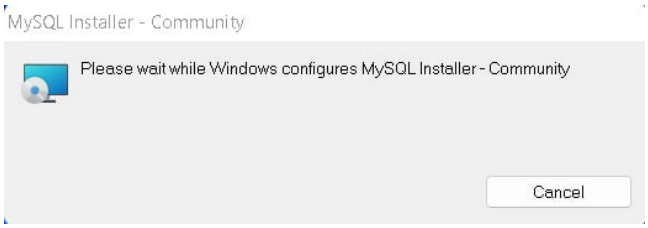
Installation Process

Let's get straight to the point. Installing MySQL on Windows is a very easy but long process. In this article, we'll follow step-by-step how to do it using the MySQL installer, which is the recommended method by the database documentation.

First of all, you need to download the installer. Click here, choose the version compatible with your operating system (32-bit or 64-bit), and hit Download. Notice that you can also choose the web installer or offline installer. The last one is much heavier while the first will be faster to download.

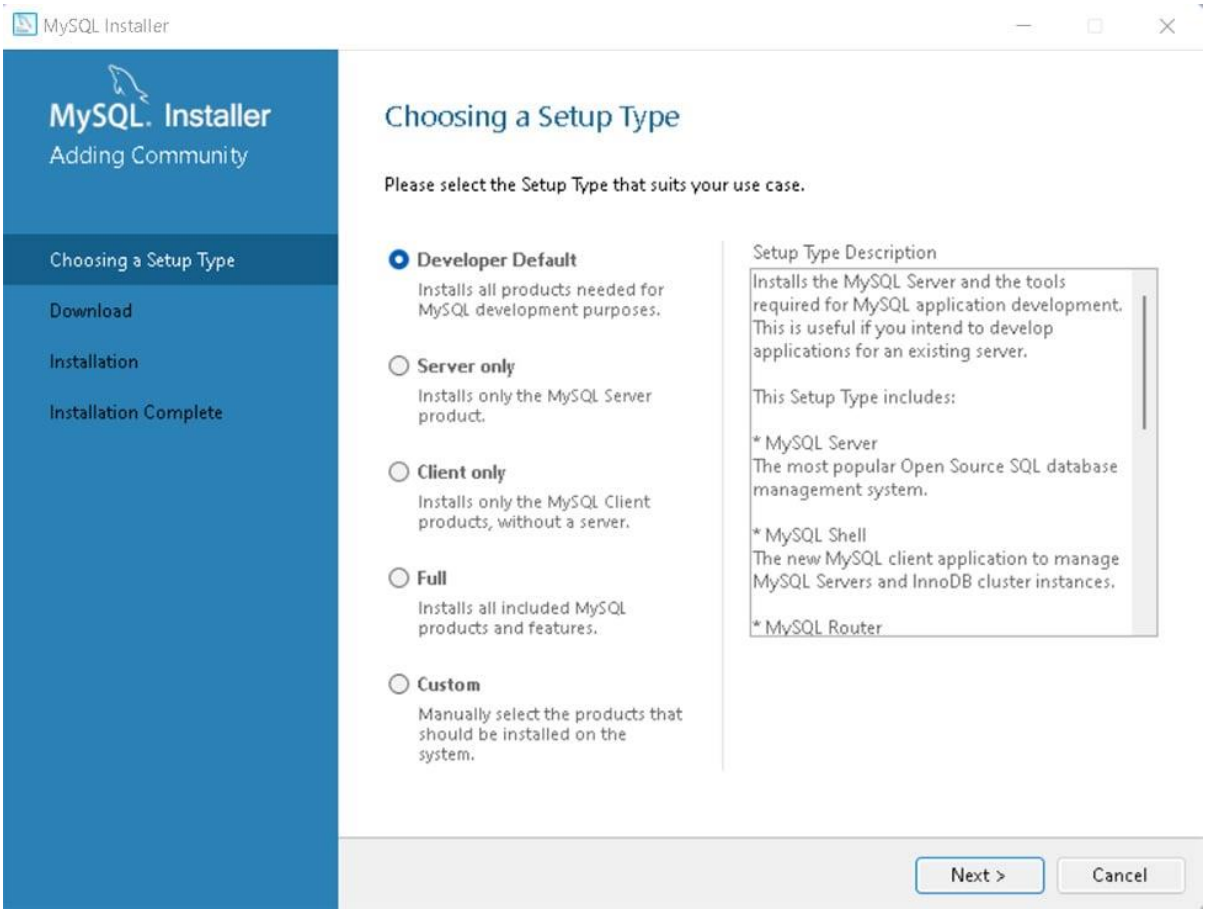
When you try to start the download, the website will ask you to log in or create an account, but you don't have to do so. Note the No thanks, just start my download button.

When you open the installer, it will first configure the installation and then ask for the user's permission to proceed:



When this is over, we'll finally see the installer interface. As you can see in the image below, the process consists of four steps:

- Choosing a setup type
- Downloading the files
- Installing the software
- Finishing the installation



There are five types of setups available in this first step and you can check the side box to see what each of them will install. However, we strongly recommend, especially if you're just getting started with SQL, to select the default option.

The most important features, among others, this setup will install are:

MySQL Server: the database server itself

MySQL Workbench: an application to manage the server

MySQL for Visual Studio: this feature enables the users to use MySQL from Visual Studio

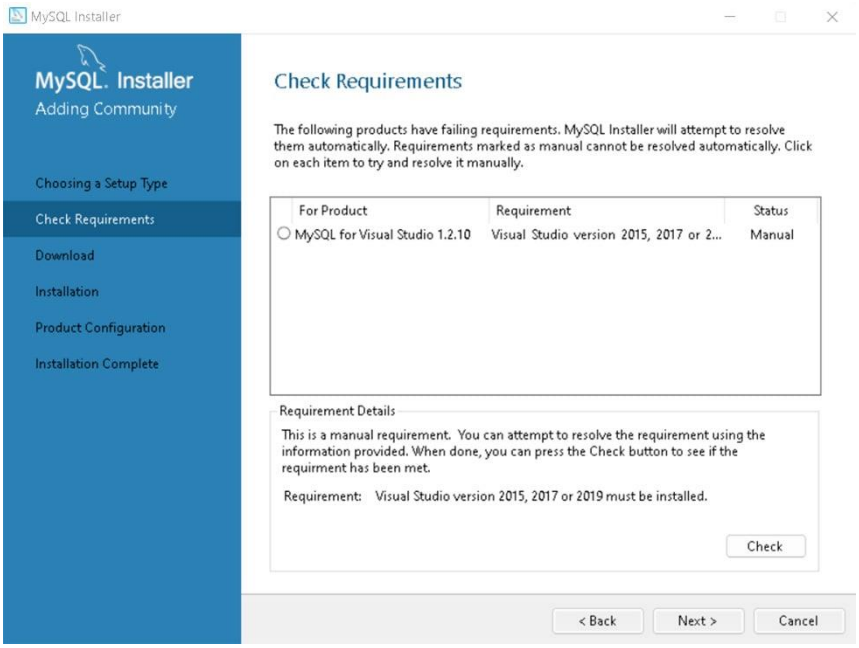
The documentation and tutorials

It's also ok to choose the full setup as this will install all MySQL resources available.

After you choose the setup option, click Next.

Requirements:

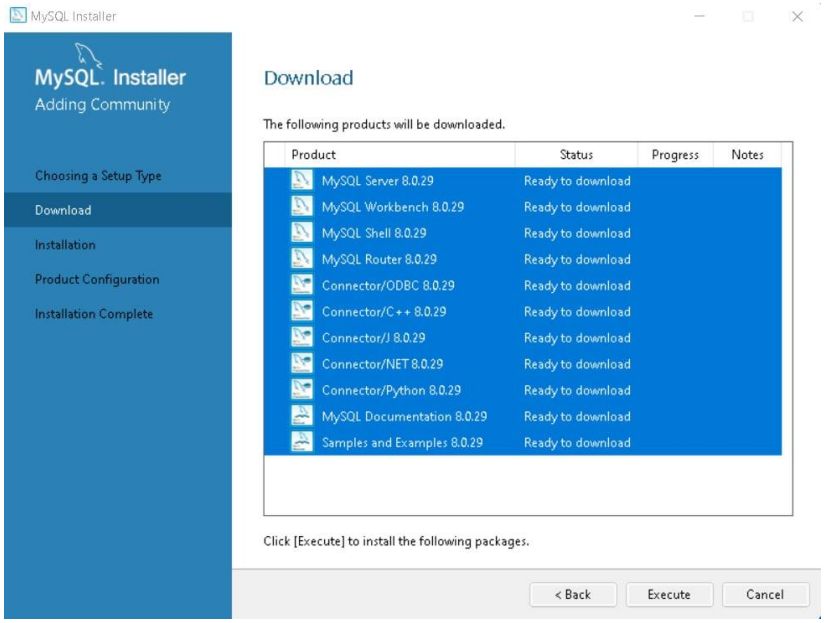
At this point, there's a chance you'll be asked to install some required software, the most common being the Visual Code. The installer can automatically solve some requirement issues; however, this is not the case here:



If you run into this, you can find one of the required versions of Visual Code here. This issue won't stop from proceeding with the installation, though.

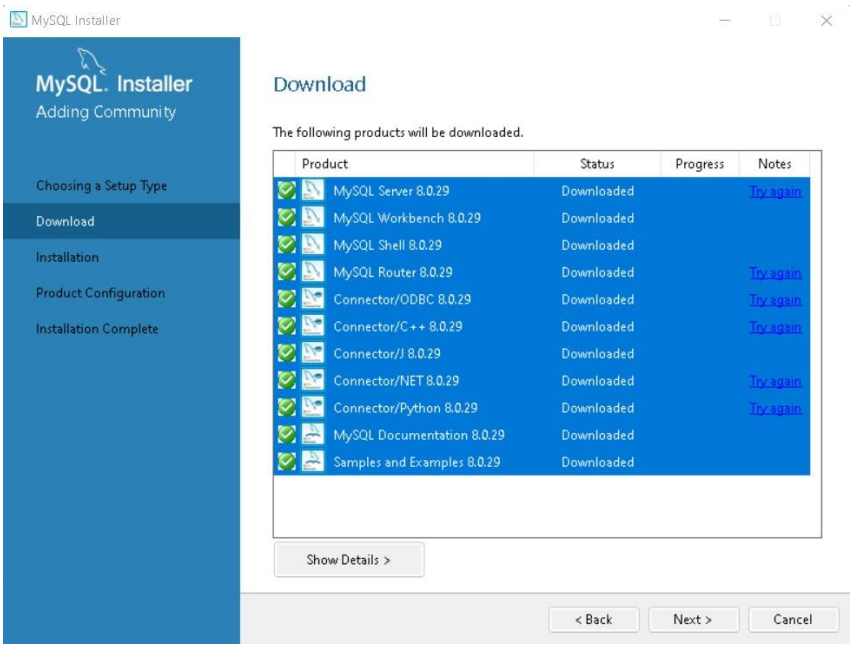
Download & Install

You have now reached the download section. The section name is self-explanatory: you'll download all the components in the setup option you selected.



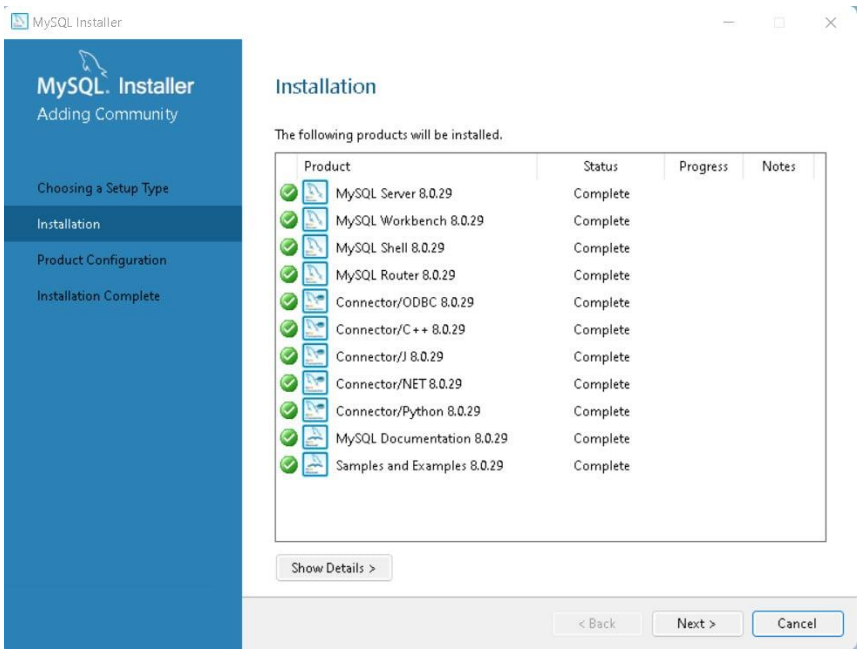
Click on Execute and the download will start. This might take a few minutes to be concluded.

When it's done, you should see tick marks on every item. Then you can proceed.



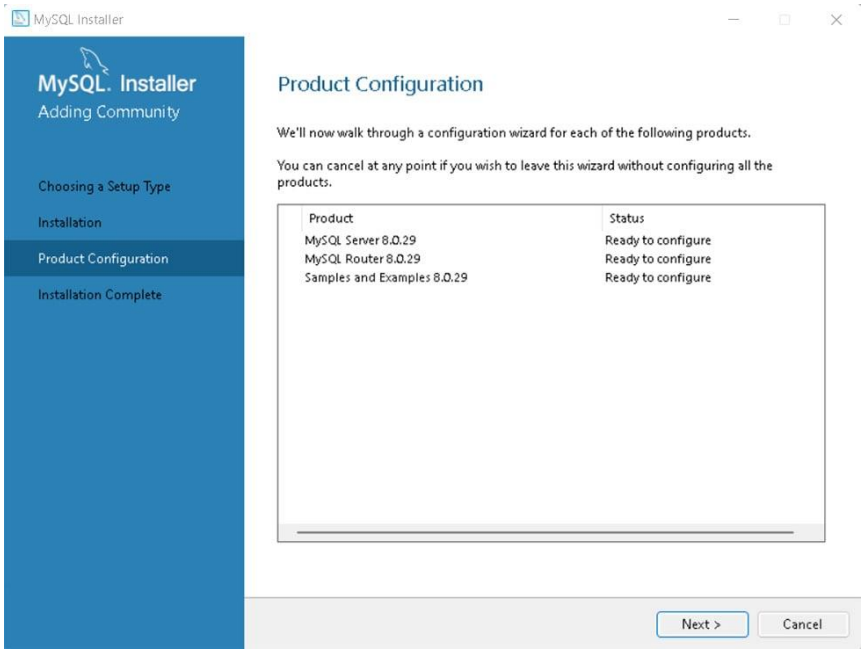
The next screen you'll see is almost the same as the last one, but now it will install all the components you've just downloaded. This step will take significantly longer than the previous one.

When it's over, you'll see all the tick marks again:

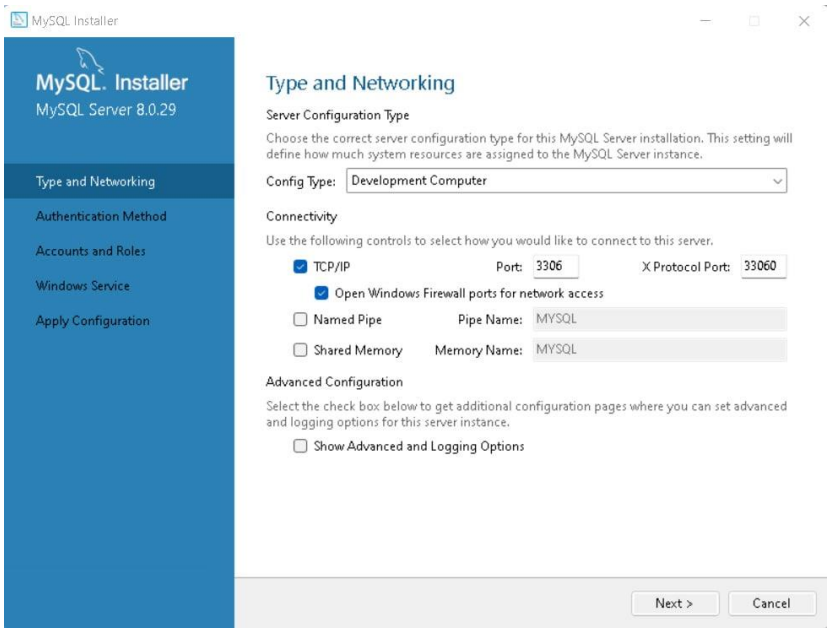


Configuration

The next step is to configure the server. You'll see the following screen. Hit Next.

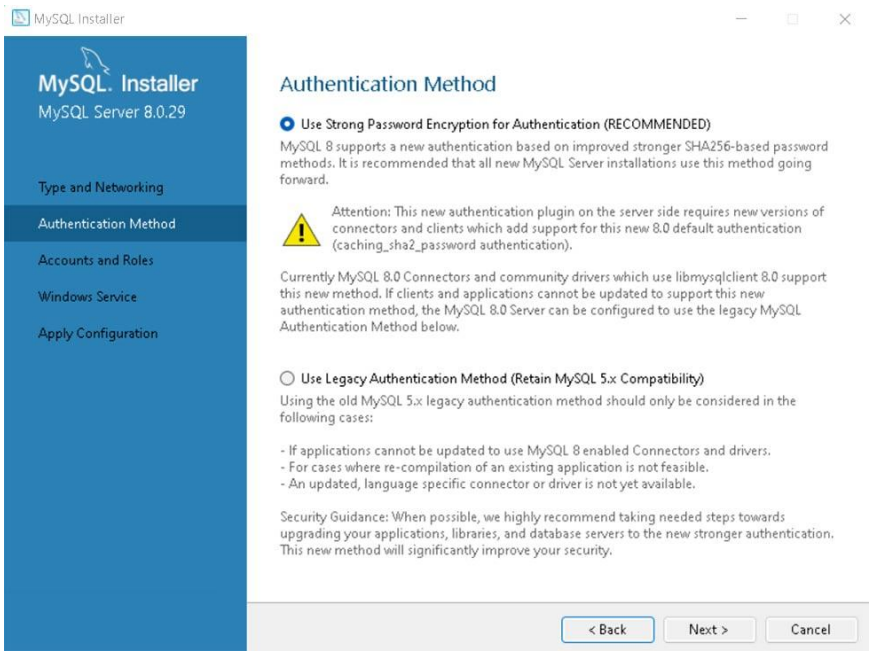


First, the installer will ask you to configure the network:



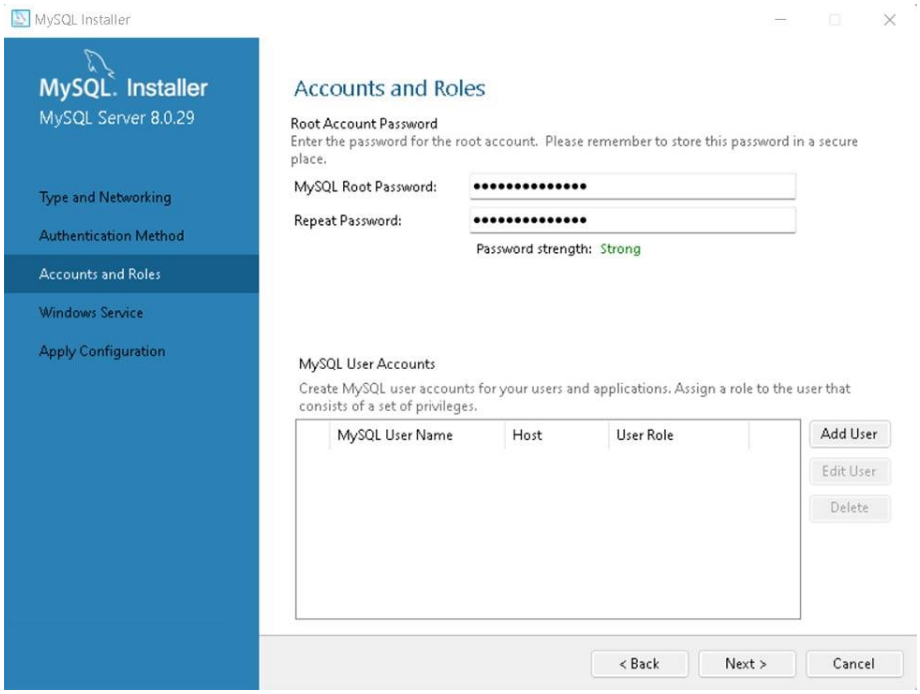
It's important to keep *Development Computer in the Config Type* field as you're probably installing this on your personal computer and not on a dedicated machine. You can choose the port, but the default will work just fine. Click Next.

For the authentication method, let's stick with the recommended option and click Next:

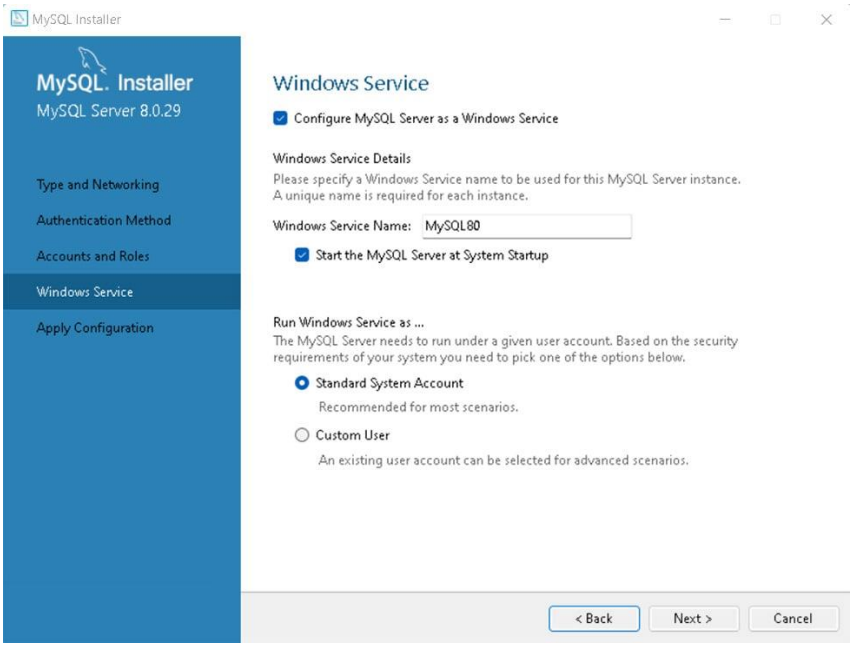


Now it's time to create the root account. You'll be asked to set a password. Remember to use a strong one.

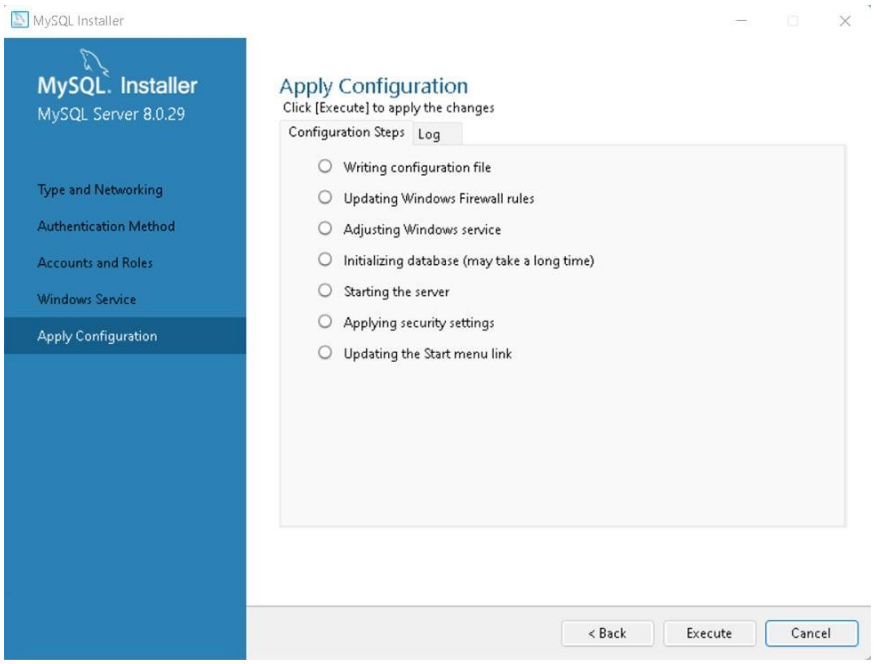
On this same screen, you can create other users and set their passwords and permissions. You just have to click on Add User and fill in the blanks. Then, click Next.



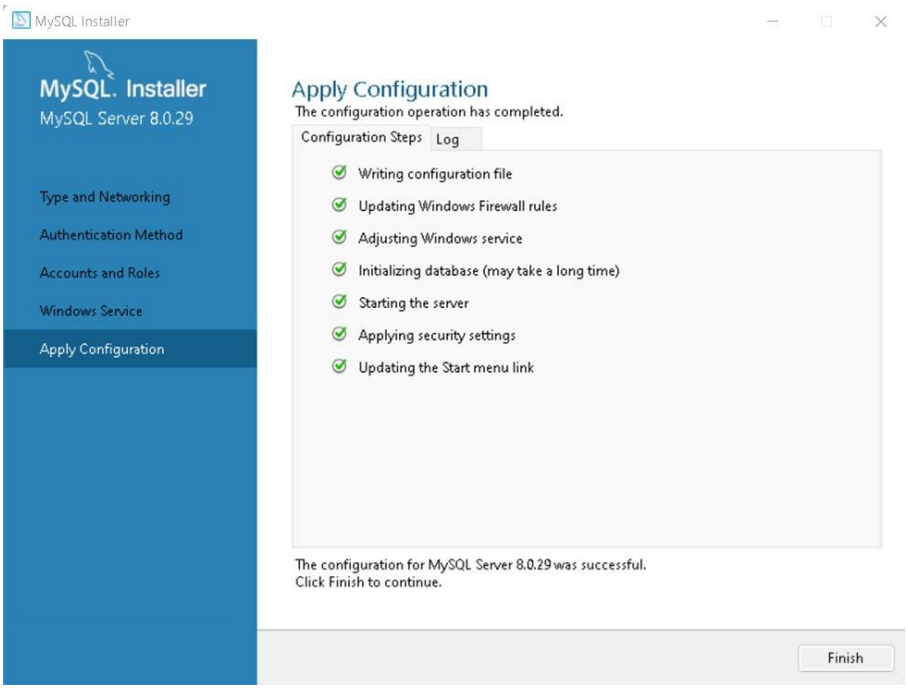
Now you can choose the Windows service details, such as the service name, account type, and if you want to start MySQL when you turn on your computer. Again, the default options will work in most cases:



The next screen applies the configuration. Execute it. This step also takes a while to be concluded.

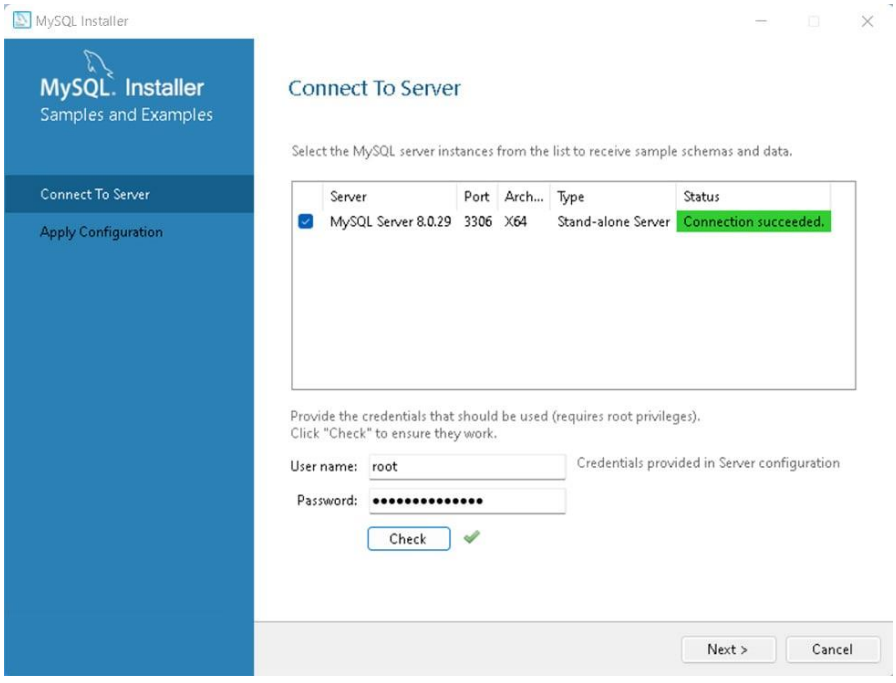


After it ends, just finish the process.



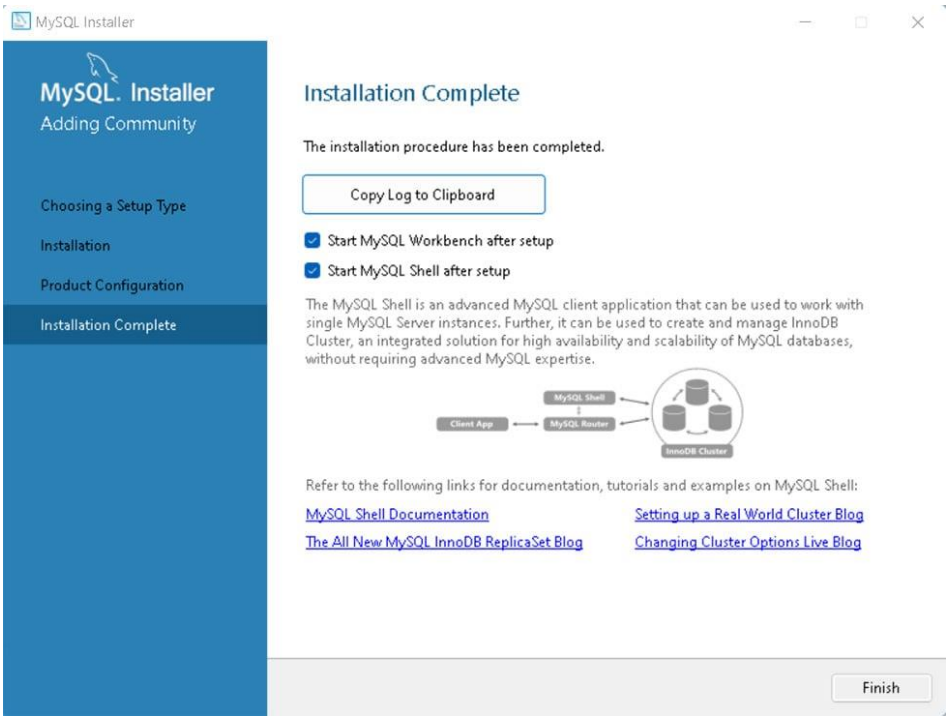
Final Steps

We're almost there! The next step is to connect to the server. Type the root account's password and click Check. You'll see the Connection succeed status:



This screen is followed by another one asking to apply the configuration. Just execute it and click Finish.

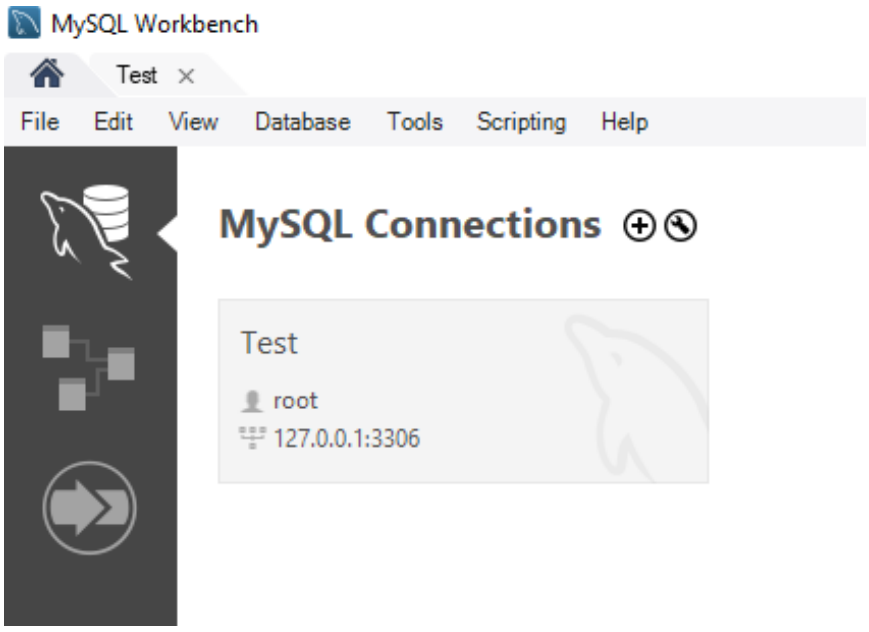
We have finally reached the last screen.



Here, you can choose whether or not to start the Workbench and Shell, and check on the documentation or other examples.

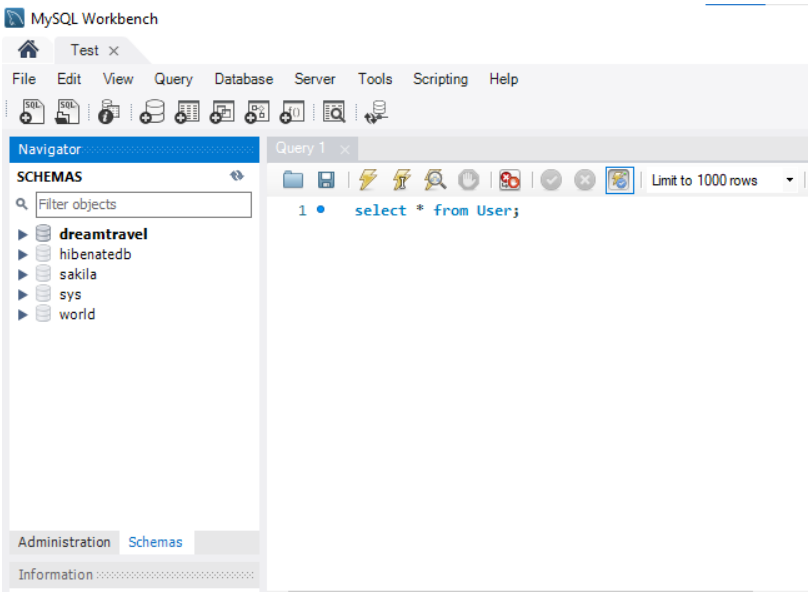
Creating your first Database using MySQL Workbench

If you chose to start the Workbench after finishing the installation, you'll see the following screen:



Choose the connection to the server you created and log into it.

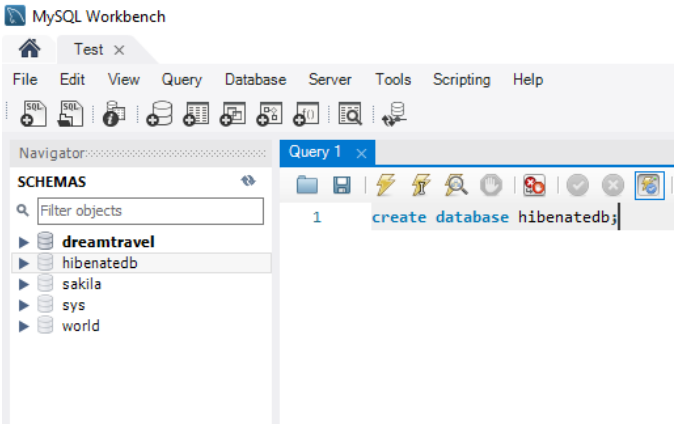
This is your working space:



Notice in the SCHEMAS window that you already have a few sample databases to play with. In the Information window, you can see the database you have selected. Of course, you have the main window to write SQL code.

Use this window to run the following command to create your first database:

CREATE DATABASE “hibernatedb”



Use the lightning icon to run the command and then click the Refresh button in the SCHEMAS window. The new database should be there.

Note that there's a message in the Output window to show that the command was successfully executed.

You'll now have a fully functional database. You can start to create tables, insert data, and build your own applications.

Experiment: 3

BANK APPLICATION CUI using JAVA & JDBC

Write a java program for bank application including with the following services:

- *Register*
- *Login*
- *View Profile*
- *Deposit*
- *Withdrawal*
- *Balance Enquiry*
- *Logout*

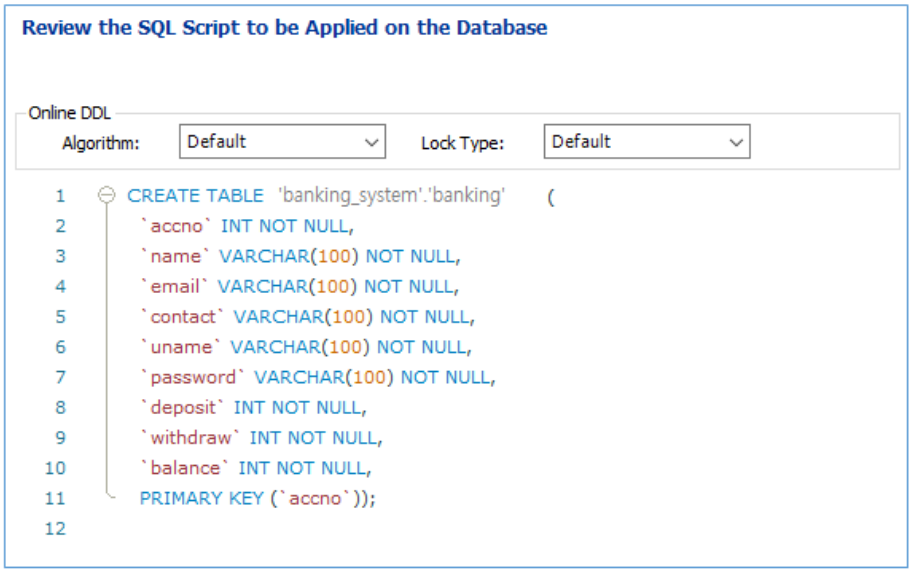
take different classes for every service.

To create a simple Banking Command User Interface Application using JDBC, we need to create the following Database, table and classes.

Database:
Create the database/schema with the name "*banking_system*".

```
1  create schema banking_system;
```

Create table "*banking*" within that schema as mentioned below.



- Open Eclipse IDE and create a new "java project" and name the project as "Banking Management System".
- Add "mysql-connector-j library" to the project folder by using "Build Path".
- Create two packages 1. DBConnection 2. Banking
- In DBConnection Package create a class DBCon to establish JDBC connection and you are done with this package.

DBCon class:

DBCon.java

```
1  package DBConnection;
2  import java.sql.*;
3
4  public class DBCon {
5     public Connection con = null;
6     public void connection() {
7        try {
8        String un = "root";
9        String pss = "password";
10       String url = "jdbc:mysql://localhost:3306/login";
11
12       Class.forName("com.mysql.cj.jdbc.Driver");
13       con = DriverManager.getConnection(url,un,pss);
14       }catch(Exception ms) {
15          System.out.println("Error Connection Establishment");
16       }
17    }
18 }
```

In the Banking package the following classes needed to be developed.

1. **Main class** which is the initiation class of let the project run.

Main.java

```
1  package banking;
2
3  public class Main {
4     public static void main(String[] args) throws Exception {
5        System.out.println("Welcome to MLRITM Banking Corporation !");
6        new Menu().menuList();
7     }
8  }
```

2. Main class redirects the page to the general Menu Options where we have Register, login and exit options which are in *"Menu Class"*.

*Menu.java

```
2  import java.util.Scanner;
3
4  public class Menu {
5     Scanner s = new Scanner(System.in);
6     public void menuList() {
7        System.out.println("1. Register");
8        System.out.println("2. LogIn");
9        System.out.println("3. Exit");
10       int opt = s.nextInt();
11       switch(opt) {
12       case 1:
13             System.out.println("=========| REGISTRATION PAGE |=========");
14             new Registration().register();
15             break;
16       case 2:
17          System.out.println("=========| LOGIN PAGE |=========");
18          new Login().acceptInput();
19          break;
20       case 3:
21          System.exit(200);
22          break;
23          default:
24             System.out.println("=========| WELCOME PAGE |=========");
25             menuList();
26       }
27    }
28 }
```

3. Registration Class

*Registration.java

```
1  package banking;
2
3  import java.util.Scanner;
6
7  public class Registration {
8     Scanner s = new Scanner(System.in);
9     DBCon db = new DBCon();
10    public void register() {
11       System.out.println("Account Number: ");
12       String accno = s.nextLine();
13       System.out.println("Name: ");
14       String name = s.nextLine();
15       System.out.println("EmailID: ");
16       String email = s.nextLine();
```

```
        System.out.println("Contact No: ");
        String con = s.nextLine();
        System.out.println("UeserName: ");
        String username = s.nextLine();
        System.out.println("Enter your Password: ");
        String password = s.nextLine();
        System.out.println("Enter your Initial Deposit Amount (Minimum Rs. 5000): ");
        int deposit = new Deposit().userInput();
        int withdraw = 0;
        int balance = deposit - withdraw;
        System.out.println("Enter your choice:\n1. Submit\n2. Reset\n3. Exit");
        int opt = s.nextInt();
        db.connection();
    try {
      if(opt==1) {
        String qry = "INSERT INTO log(accno,name, email, contact, uname, password,deposit, withdraw, balance) "
            + "VALUES('"+accno+"','"+name+"','"+email+"','"+con+"','"+username+"','"+password+"','"+deposit+"','"+withdraw+"','"+balance+"')";
        Statement st = db.con.createStatement();
        int rs = st.executeUpdate(qry);
        System.out.println(rs);
        if(rs>0) {
          System.out.println("=========| WELCOME PAGE |=========");
          Main.main(null);
        }else {
          System.out.println("Error while registration process...");
          System.out.println("=========| REGISTRATION PAGE |=========");
          register();
        }
      }else if(opt==2) {
        System.out.println("=========| REGISTRATION PAGE |=========");
        register();
      }else if(opt == 3){
        System.exit(150);
      }else {
        System.out.println("=========| WELCOME PAGE |=========");
        Main.main(null);
      }
    }catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
```

In the above program “case 3”-logout is happening without redirecting to Logout class.

4. Login class

Login.java

```
1  package banking;
2
3  import java.sql.ResultSet;
8
9  class Login{
10   static int ac, amount;
11   String pw;
12   DBCon dbc = new DBCon();
13   public void acceptInput(){
14   Scanner scanner = new Scanner(System.in);
15     System.out.print("Enter AccountNumber: ");
16     ac = scanner.nextInt();
17     System.out.print("Enter Password: ");
18     scanner.nextLine();
19     pw = scanner.nextLine();
20     try {
21       verify();
22     }catch(Exception e) {
23       e.printStackTrace();
24     }
25     scanner.close();
26   }
27   public void verify() throws Exception{
28     try {
29       dbc.connection();
30       String qv = "SELECT * FROM log WHERE accno='"+ac+"' AND password='"+pw+"'";
31       Statement sv = dbc.con.createStatement();
32       ResultSet rs = sv.executeQuery(qv);
33       if(rs.next()) {
34         do {
35           System.out.println("You Logged In to MLRITM Bank Account Successfully !");
36           //BankingAccount b = new BankingAccount();
37           System.out.println(" ");
38           amount = rs.getInt("balance");
39           System.out.println("Your Account Balance is: "+amount+" Rupees");
40           System.out.println(" ");
41           new UMenu().showMenu();
42         }while(rs.next());
43       }else{
44         InvalidTransaction loginfailed = new InvalidTransaction("Incorrect Login Credentials !");
45         System.out.println(loginfailed.getMessage());
46         System.out.println("=========| LOGIN PAGE |=========");
47         new Menu().menuList();
48       }
49     }catch(Exception e) {
50       e.printStackTrace();
51     }
52   }
53 }
```

5. UMenu Class (User Menu Class):

UMenu.java

```
1  package banking;
2  import java.util.Scanner;
3
4  class UMenu{
5     int selectedOption;
6     BankingAccount b = new BankingAccount();
7     public void showMenu()
8     {
9        System.out.println("Please Select an option below:");
10       System.out.println("Press 1 to Deposit Amount.");
11       System.out.println("Press 2 to Withdraw Amount.");
12       System.out.println("Press 3 to View Balance");
13       System.out.println("Press any key to Logout");
14       System.out.println(" ");
15       Scanner scanner = new Scanner(System.in);
16       System.out.print("Press any key: ");
17       selectedOption = scanner.nextInt();
18       switch (selectedOption) {
19          case 1:
20             int depamt =  new Deposit().userInput();
21             new Deposit().deposit_amount(depamt);
22             showMenu();
23            break;
24          case 2:
25             System.out.print("Please Enter the amount to withdraw: ");
26             int withamt=new Withdrawal().userInput();
27             new Withdrawal().withdraw_amount(withamt);
28             showMenu();
29            break;
30          case 3:
31             new Balance().viewBalance();
32             showMenu();
33            break;
34          default:
35             System.out.println("Transaction Ended, Your MLRITM Bank Account Logout Successfully !");
36             System.exit(0);
37            break;
38       }
39    }
40 }
```

In the above program “default case” logout is happening directly instead of using Logout class.

6. Deposit Class:

Deposit.java

```
1  package banking;
2  import java.sql.Statement;
6
7  class Deposit{
8     int amount = Login.amount;
9     int ac = Login.ac;
10    DBCon dbc = new DBCon();
11
12    public int userInput(){
13       Scanner scanner = new Scanner(System.in);
14       System.out.print("Enter the amount to be deposited: ");
15       amount = scanner.nextInt();
16       if(amount<=0){
17          InvalidTransaction depositnegativeError = new InvalidTransaction("Invalid Deposit Amount");
18          System.out.println(depositnegativeError.getMessage());
19          userInput();
20       }else{
21          return amount;
22       }
23       return amount;
24    }
25
26    public void deposit_amount(int amt){
27       System.out.println(amount+"---"+amt);
28       amount = amount + amt;
29       try {
30          dbc.connection();
31          String qv = "UPDATE log SET deposit='"+amt+"',balance='"+amount+"' WHERE accno='"+ac+"'";
32          Statement sv = dbc.con.createStatement();
33          int rs = sv.executeUpdate(qv);
34          if(rs>0) {
35             System.out.println("Amount deposited Successfully");
36             System.out.println(" ");
37             System.out.println("Total Balance: " +amount);
38             System.out.println(" ");
39          }
40          Login.amount = amount;
41          new UMenu().showMenu();
42       }catch(Exception e) {
43          System.out.println("Error in SQL Query");
44       }
45    }
46 }
```

7. Withdrawal Class:

Withdrawal.java

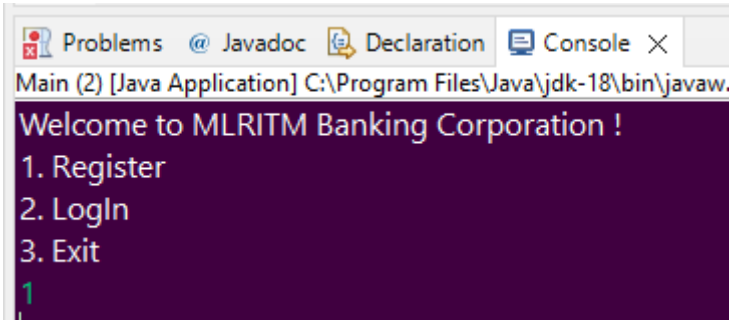
```
1  package banking;
2  import java.sql.Statement;
6
7  class Withdrawal{
8     int amount = Login.amount;
9     int ac = Login.ac;
10    DBCon dbc = new DBCon();
11
12    public void withdraw_amount(int amt){
13       System.out.println(" ");
14       if(amount < amt)
15       {
16          InvalidTransaction invalidWithDraw = new InvalidTransaction("InValid Withdrawal Amount");
17          System.out.println(invalidWithDraw.getMessage());
18       }else{
19          amount = (amount - amt);
20          Login.amount = amount;
21          try {
22             dbc.connection();
23             String qv = "UPDATE log SET withdraw='"+amt+"', balance='"+amount+"' WHERE accno='"+ac+"'";
24             Statement sv = dbc.con.createStatement();
25             int rs = sv.executeUpdate(qv);
26               if(rs>0) {
27                  System.out.println("Please Collect your " + amt +" Rupees");
28                  System.out.println(" ");
29                  System.out.println("Available Balance: " +amount);
30                  System.out.println(" ");
31               }
32          }catch(Exception e) {
33             e.printStackTrace();
34          }
35       }
36    }
37    public int userInput(){
38       Scanner scanner = new Scanner(System.in);
39       System.out.print("Enter the amount to be withdrawn: ");
40       amount = scanner.nextInt();
41       if(amount<=0){
42          InvalidTransaction depositnegativeError = new InvalidTransaction("Invalid Deposit Amount");
43          System.out.println(depositnegativeError.getMessage());
44          userInput();
45       }else{
46          return amount;
47       }
48       return amount;
49    }
50 }
```

8. Balance Class:

Balance.java

```
1  package banking;
2
3  import java.sql.ResultSet;
7
8  public class Balance {
9     int amount = Login.amount;
10    int ac = Login.ac;
11    DBCon dbc = new DBCon();
12
13    public void viewBalance() {
14       try {
15          dbc.connection();
16          String qv = "SELECT * FROM log WHERE accno='"+ac+"' ";
17          Statement sv = dbc.con.createStatement();
18          ResultSet rs = sv.executeQuery(qv);
19          if(rs.next()) {
20             do {
21                amount = rs.getInt("balance");
22             System.out.println("Your available balance is: "+amount);
23             }while(rs.next());
24          }
25          new UMenu().showMenu();
26       }catch(Exception e) {
27          System.out.println("Error in SQL Query");
28       }
29    }
30 }
```

Output:



```
=========| REGISTRATION PAGE |=========
Account Number:
110402201
Name:
Shafakhatullah Khan Mohammed
EmailID:
shafakhat@mlritm.ac.in
Contact No:
9988776655
UeserName:
shafakhat
Enter your Password:
Shafakhat@123
Enter your Initial Deposit Amount (Minimum Rs. 5000):
Enter the amount to be deposited: 5000
Enter your choice:
1. Submit
2. Reset
3. Exit
1
1
=========| WELCOME PAGE |=========
Welcome to MLRITM Banking Corporation !
1. Register
2. LogIn
3. Exit
```

```
Problems  Javadoc  Declaration  Console
Main (2) [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe (J
Welcome to MLRITM Banking Corporation !
1. Register
2. LogIn
3. Exit
2
=========| LOGIN PAGE |=========
Enter AccountNumber: 110402201
Enter Password: Shafakhat@123
You Logged In to MLRITM Bank Account Successfully !

Your Account Balance is: 5000 Rupees

Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key:
```

```
Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key: 1
Enter the amount to be deposited: 2000
5000---2000
Amount deposited Successfully

Total Balance: 7000

Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key:
```

```
Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key: 3
Your available balance is: 7000
Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key:
```

```
Press any key: 3
Your available balance is: 7000
Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key: 2
Please Enter the amount to withdraw: Enter the amount to be withdrawn: 5000

Please Collect your 5000 Rupees

Available Balance: 2000

Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key:
```

```
Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key: 3
Your available balance is: 2000
Please Select an option below:
Press 1 to Deposit Amount.
Press 2 to Withdraw Amount.
Press 3 to View Balance
Press any key to Logout

Press any key: 5
Transaction Ended, Your MLRITM Bank Account Logout Successfully !
```

Experiment: 4

DESIGN & STYLE BANKING SYSTEM WEB APPLICATION using [HTML5, CSS3]

Create a website for the banking system with the following web pages:

- *Register*
- *Login*
- *View Profile*
- *Deposit Details*
- *Funds Transfer*
- *Balance Enquiry*

Home Page:

HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>MLRITM BANK</title>
<meta name="keywords" content="MLRITM, MLRITM BANK,BANK" />
<meta name="description" content="MLRITM BANK IS NO 1 BANK IN INDIA" />
<link href="style.css" rel="stylesheet" type="text/css" />

</head>
<body>

<div id="templatemo_wrapper">

       <div id="templatemo_header">
       <div id="site_title">
            <h1><a href="index.html"><img src="images/mb_logo.png" height="100" width="100"
alt="logo" /><span>MLRITM Bank</span></a></h1>
        </div> <!-- end of site_title -->

        <div id="header_right">
            <ul id="social_box">
                <li><a href="login.html">LogIn</a></li>
                <li><a href="signup.html">SignUp</a></li>
            </ul>
            <div id="templatemo_menu">
                <ul>
                      <li><a href="index.html" class="current">START
HERE<span>Home</span></a></li>
                    <li><a href="contactus.html">ANY QUESTION?<span>Contact Us</span></a></li>
                </ul>
            </div> <!-- end of templatemo_menu -->
       </div>
    </div> <!-- end of templatemo_header -->

    <div id="templatemo_sp_box">

       <div class="col_w270">
            <img src="images/bar-graph.png" alt="image" />
            <div class="right">
```

```
                <h5>Home Loan</h5>
              <p>We provide home loans at minimal interest rates.</p>

                    </div>
        </div>

        <div class="col_w270">
               <img src="images/to-do.png" alt="image" />
            <div class="right">
                <h5>Car Loan</h5>
                <p>To make your life hassel free we have car loans for you. </p>

                    </div>
        </div>

        <div class="col_w270 col_last">
               <img src="images/gant-chart.png" alt="image" />
            <div class="right">
                <h5>Education Loan</h5>
                <p>To the bright and glorious future of your kids we provide fabulous
education loan.</p>
                    </div>
        </div>

    </div>
<!-------------------------------UPTO HERE SAME FOR ALL PAGES------------------------------>

    <div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">

     <div class="col_w590 float_l">

                 <h2>MLRITM BANK</h2>

                <div class="image_wrapper image_fl"><a href="index.html"><img
src="images/mb_logo.png" height="200" width="200" alt="image" /></a></div>
                <p align="justify">MLRITM BANK is provided by <a
href="http://www.mlritm.ac.in" target="_parent">MLRITM </a> for everyone. MLRITM Bank Internet
Banking is simple, convenient, 100% secure, and lets you carry out a wide range of banking
transactions and access numerous Net Banking features in just a few clicks. Now, say goodbye
to long queues and unwanted delays. With ICICI Bank, Net Banking instantly unlocks a better
lifestyle anytime and anywhere.</p>
              <p align="justify">MLRITM Bank Internet Banking is simple, convenient, 100%
secure, and lets you carry out a wide range of banking transactions and access numerous Net
Banking features in just a few clicks. Now, say goodbye to long queues and unwanted delays.
With ICICI Bank, Net Banking instantly unlocks a better lifestyle anytime and anywhere</p>

       <div class="cleaner_h20"></div>

        <h3>WORLD CLASS INTERNET BANKING IN JUST 4 STEPS</h3>

                    <ul class="tmo_list col_w270">
              <li><a href="#">1. Call Customer Care and authenticate yourself</a></li>
                    <li><a href="#">2. Choose 'Self Banking' or click the ‘I want my User
ID’ button</a></li>

                    </ul>

                <ul class="tmo_list col_w270 col_last">
                    <li><a href="#">3. Generate your password instantly, online</a></li>
```

```
                                    <li><a href="#">4. Log in with your user ID and password
to access the account</a></li>
                </ul>

        <div class="cleaner"></div>

              </div> <!-- end of col_w590 -->

<!--------------------------------FROM HERE SAME FOR ALL PAGES------------------------------>
            <div class="col_w270 col_last">

              <h3>WE TAKE SECURITY SERIOUSLY !</h3>

              <div class="sb_news_box">

                   <a href="index.html"><img src="images/templatemo_image_02.jpg" alt="image
2" /></a>
                     <div class="nb_right">
                       Peace of mind for you as we have the most advanced technology &
protection</div>
                   <div class="cleaner"></div>

                </div>

                     <div class="sb_news_box">

                   <a href="index.html"><img src="images/templatemo_image_03.jpg" alt="image
3" /></a>
                            <div class="nb_right">
                       Factor i-safe authentication</div>
                   <div class="cleaner"></div>

                </div>

                <div class="sb_news_box">

                   <a href="index.html"><img src="images/templatemo_image_02.jpg" alt="image
4" /></a>
                     <div class="nb_right">
                       End-to-end 256 bit Encryption.</div>
                   <div class="cleaner"></div>

                </div>

                            <div class="sb_news_box">

                   <a href="index.html"><img src="images/templatemo_image_03.jpg" alt="image
4" /></a>
                     <div class="nb_right">
                       We make you feel special with wireless banking.</div>
                   <div class="cleaner"></div>

                </div>
           </div>

            <div class="cleaner"></div>
        </div> <!-- end of templatemo_content -->

        <div id="templatemo_content_bottom"></div>
       </div>
```

```
    <div id="templatemo_footer">

        Copyright © 2023 <a href="#">MLRITM BANK</a> | <a href="http://www.mlritm.ac.in"
target="_parent">MLRITM BANK</a> designed by <a href="http://www.shafakhat.is-best.net"
target="_parent">Shafakhatullah Khan Moammed</a>

    </div> <!-- end of templatemo_footer -->

</div> <!-- end of templatemo_wrapper -->
</body>
</html>
```

Contact Us Page:

HTML Code:

Copy the code of first page and just replace the following div with the div available there in Home page.

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">

              <div class="col_w590 float_l">

                <h2>Contact Us</h2>

              <p>To contact MLRITM Bank’s Customer Care, use the contact details updated here.
Please do not use the contact details updated on any other website as it can be fake and may
risk your account security. </p>
            <div class="cleaner_h40"></div>

              <div class="col_w270">

                     <h6>Company Location One</h6>
                180-240 Quisque odio quam, <br />
                Pulvinar sit amet convallis eget, 10480<br />
                Venenatis ut turpis<br />
               <br />
                Tel: 010-020-0880<br />
               Fax: 020-020-0770</div>

                <div class="col_w270 col_last">

                <h6>Company Address Two</h6>
                160-320 Duis lacinia dictum, <br />
                PVestibulum auctor, 10560<br />
                Nam rhoncus, diam a mollis tempor<br />
                <br />
                Tel: 080-040-0220<br />
                Fax: 090-040-0330</div>


            <div class="cleaner_h50"></div>

              <div id="contact_form">

                   <h4>Contact Form</h4>

                     <form method="post" name="contact" action="#">

                   <div class="col_w270">

                       <input type="hidden" name="post" value="Send" />
                       <label for="author">Name:</label> <input type="text" id="author"
name="author" class="required input_field" />
                       <div class="cleaner_h10"></div>

                       <label for="email">Email:</label> <input type="text" id="email"
name="email" class="validate-email required input_field" />
                       <div class="cleaner_h10"></div>
```

```
                        <label for="url">Phone:</label> <input type="text" name="url" id="url"
class="input_field" />
                        <div class="cleaner_h10"></div>

                    </div>

                    <div class="col_w270 col_last">

                        <label for="text">Message:</label> <textarea id="text" name="text"
rows="0" cols="0" class="required"></textarea>
                        <div class="cleaner_h10"></div>

                        <input type="submit" class="submit_btn float_l" name="submit"
id="submit" value="Send" />
                        <input type="reset" class="submit_btn float_r" name="reset" id="reset"
value="Reset" />
                    </div>

                    </form>

            </div>

            <div class="cleaner"></div>

              </div> <!-- end of col_w590 -->
```

SignUp page (Registration):

HTML Code:

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">
              <div class="col_w590 float_l">
            <div class="cleaner_h20"></div>
              <div id="contact_form">
                    <h2>SignUp</h2>
                      <form onsubmit="validate()" method="post" id="signup" name="signup"
action="#">

                    <div class="col_w270">
                                          <label for="author">Account No:</label> <input
type="text" id="accno" name="accno" class ="input_field"required/>
                        <div class="cleaner_h10"></div>

                                          <label for="author">Name:</label>
                        <input type="text" id="cname" name="cname" class ="input_field"
required/>
                        <div class="cleaner_h10"></div>

                                          <label for="author">Aadhar No:</label>
                        <input type="text" id="ad_no" name="ad_no" class ="input_field"
required/>
                        <div class="cleaner_h10"></div>

                                          <label for="author">Contact No:</label>
                        <input type="text" id="c_no" name="c_no" class ="input_field"
required/>
                        <div class="cleaner_h10"></div>

                        <label for="author">Email:</label>
                        <input type="email" id="email" name="email" class ="input_field"
required/>
                        <div class="cleaner_h10"></div>

                        <label for="email">Password:</label>
                        <input type="password" id="password" name="password" class
="input_field" required/>
                        <div class="cleaner_h10"></div>

                                          <label for="author">Initial Deposit
Amount:</label>
                        <input type="text" id="d_amt" name="d_amt" class="input_field"
required/>
                        <div class="cleaner_h10"></div>

                        <input type="submit" class="submit_btn float_l" name="submit"
id="submit" value="SignUp" />
                        <input type="reset" class="submit_btn float_r" name="reset" id="reset"
value="Reset" />

                    </div>
                    </form>
            </div>

            <div class="cleaner"></div>
```

```
</div> <!-- end of col_w590 -->
```

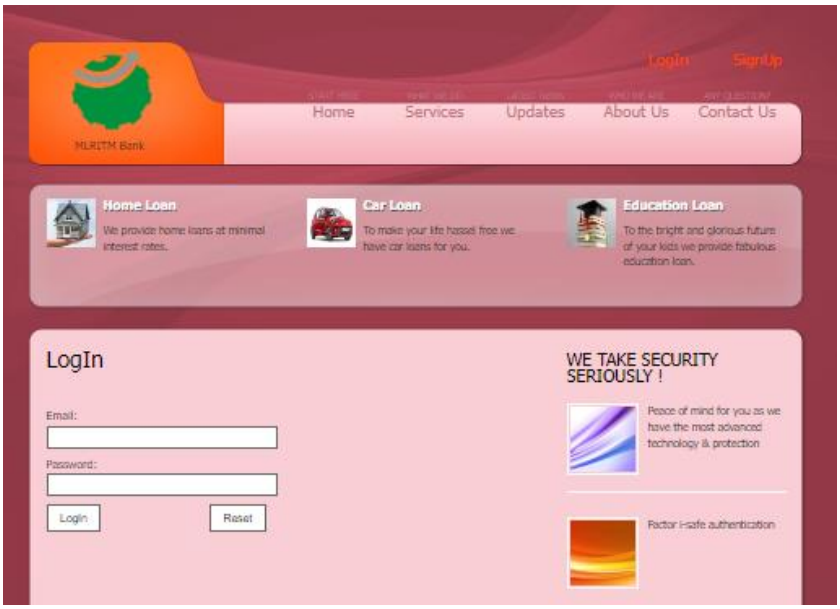
Output:

LogIn Page:

HTML Code:

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">
              <div class="col_w590 float_l">
                 <h2>LogIn</h2>
            <div class="cleaner_h20"></div>
              <div id="contact_form">
                     <form onsubmit = "logval()" method="post" id="login" name="login"
action="#">
                   <div class="col_w270">

                       <label for="author">Email:</label>
                       <input type="email" id="email" name="email" class="input_field"
required/>
                       <div class="cleaner_h10"></div>

                       <label for="email">Password:</label>
                       <input type="password" id="password" name="password"
class="input_field" required />
                       <div class="cleaner_h10"></div>

                       <input type="submit" class="submit_btn float_l" name="submit"
id="submit" value="LogIn" />
                       <input type="reset" class="submit_btn float_r" name="reset" id="reset"
value="Reset" />
                   </div>
                   </form>
            </div>
            <div class="cleaner"></div>
              </div> <!-- end of col_w590 -->
```

Output:



Profile Page:

HTML Code:

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">

              <div class="col_w590 float_l">

                <h2>Profile</h2>

            <div class="cleaner_h20"></div>

              <div id="contact_form">

                           <form method="post" name="contact" action="#">

                    <div class="col_w270">

                        <input type="hidden" name="post" value="Send" />
                                            <label for="author">Account No:</label> <input
type="text" id="author" name="author" class="validate-email required input_field" disabled/>
                        <div class="cleaner_h10"></div>

                                            <label for="author">Name:</label> <input
type="text" id="author" name="author" class="validate-email required input_field" />
                        <div class="cleaner_h10"></div>

                                            <label for="author">Aadhar No:</label> <input
type="text" id="author" name="author" class="validate-email required input_field" />
                        <div class="cleaner_h10"></div>

                                            <label for="author">Contact No:</label> <input
type="text" id="author" name="author" class="validate-email required input_field" />
                        <div class="cleaner_h10"></div>

                        <label for="author">Email:</label> <input type="email" id="author"
name="author" class="validate-email required input_field" />
                        <div class="cleaner_h10"></div>

                        <label for="email">Password:</label> <input type="password"
id="password" name="password" class="validate-password required input_field" />
                        <div class="cleaner_h10"></div>

                    </div>
                    </form>

            </div>

            <div class="cleaner"></div>

              </div> <!-- end of col_w590 -->
```

Output:


MLRITM Bank
LogOut
View
Profile
Check
Balance
Account
Deposit
Funds
Transfer
Home Loan
We provide home loans at minimal interest rates.
Car Loan
To make your life hassel free we have car loans for you.
Education Loan
To the bright and glorious future of your kids we provide fabulous education loan.
Profile
Account No:
Name:
Aadhar No:
Contact No:
Email:
Password:
WE TAKE SECURITY SERIOUSLY !
Peace of mind for you as we have the most advanced technology & protection
Factor i-safe authentication
End-to-end 256 bit Encryption.
We make you feel special with wireless banking.
Copyright © 2023 MLRITM BANK | MLRITM BANK designed by Shafakhatullah Khan Mohammed

Balance Page:

HTML Code:

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">

              <div class="col_w590 float_l">

                <h2>Balance Enquiry</h2>

            <div class="cleaner_h20"></div>


              <div id="contact_form">

                    <form method="post" name="contact" action="#">

                  <div class="col_w270">

                      <input type="hidden" name="post" value="Send" />
                      <label for="author">Available Balance:</label> <input type="email"
id="author" name="author" class="validate-email required input_field" />
                      <div class="cleaner_h10"></div>
                  </div>

                  </form>

            </div>

            <div class="cleaner"></div>

              </div> <!-- end of col_w590 -->
```

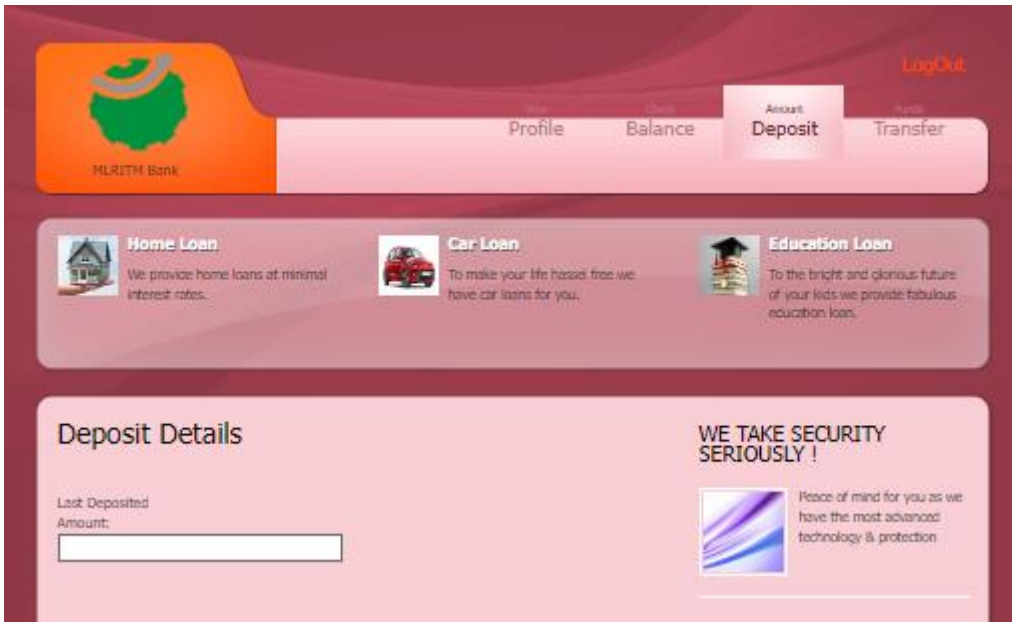
Output:

Deposit Details Page:

HTML Code:

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">

              <div class="col_w590 float_l">

                <h2>Deposit Details</h2>

            <div class="cleaner_h20"></div>


              <div id="contact_form">

                     <form method="post" name="contact" action="#">

                   <div class="col_w270">

                       <input type="hidden" name="post" value="Send" />
                       <label for="author">Last Deposited Amount:</label> <input type="email"
id="author" name="author" class="validate-email required input_field" />
                       <div class="cleaner_h10"></div>
                    </div>

                   </form>

            </div>

            <div class="cleaner"></div>

              </div> <!-- end of col_w590 -->
```

Output:



Funds Transfer Page:

HMTL Code:

```
<div id="templatemo_content_wrapper">
       <div id="templatemo_content_top"></div>
        <div id="templatemo_content">

              <div class="col_w590 float_l">

                 <h2>Funds Transfer</h2>

            <div class="cleaner_h20"></div>

              <div id="contact_form">

                     <form method="post" name="contact" action="#">

                   <div class="col_w270">

                       <input type="hidden" name="post" value="Send" />
                       <label for="author">Enter Account No:</label> <input type="text"
id="author" name="author" class="validate-email required input_field" />
                       <div class="cleaner_h10"></div>

                       <label for="email">IFSC CODE:</label> <input type="text" id="text"
name="password" class="validate-password required input_field" />
                       <div class="cleaner_h10"></div>

                                          <label for="email">Amount:</label> <input
type="text" id="text" name="password" class="validate-password required input_field" />
                       <div class="cleaner_h10"></div>

                       <input type="submit" class="submit_btn float_l" name="submit"
id="submit" value="Transfer" />
                       <input type="reset" class="submit_btn float_r" name="reset" id="reset"
value="Decline" />

                   </div>

                   </form>

            </div>

            <div class="cleaner"></div>

              </div> <!-- end of col_w590 -->
```

Output:


LogOut
MLRITM Bank
Profile
Balance
Deposit
Transfer
Home Loan
We provide home loans at minimal interest rates.
Car Loan
To make your life hassel free we have car loans for you.
Education Loan
To the bright and glorious future of your kids we provide fabulous education loan.
Funds Transfer
Enter Account No:
IFSC CODE:
Amount:
Transfer
Decline
WE TAKE SECURITY SERIOUSLY !
Peace of mind for you as we have the most advanced technology & protection
Factor i-safe authentication
End-to-end 256 bit Encryption.
We make you feel special with wireless banking.
Copyright © 2023 MLRITM BANK | MLRITM BANK designed by Shahabatullah Khan Mohammad

CSS FILE:

style.css:

```
/*
Credit: http://www.shafakhat.is-best.net
*/

body {
	margin: 0px;
	padding: 0px;
	color: #333;
	font-family: Tahoma, Geneva, sans-serif;
	font-size: 13px;
	line-height: 1.5em;
	background-color: #923947;
	background-image: url(images/templatemo_body.jpg);
	background-repeat: no-repeat;
	background-position: top center;
}

a, a:link, a:visited { color: #ff4301; text-decoration: none; }
a:hover { color: #999900; text-decoration: underline; }

p { margin: 0px; padding: 0; }
img { border: none; }

h1, h2, h3, h4, h5, h6 { color: #000; }
h1 { font-size: 40px; font-weight: normal; margin: 0 0 30px 0; padding: 5px 0; }
h2 { font-size: 28px; font-weight: normal; margin: 0 0 30px 0; padding: 0; }
h3 { font-size: 21px; margin: 0 0 15px 0; padding: 0; padding: 0; color: #000; font-weight:
normal; }
h4 { font-size: 18px; margin: 0 0 20px 0; padding: 0; color: #333; }
h5 { font-size: 16px; margin: 0 0 10px 0; padding: 0; color: #fff; text-shadow: 1px 1px 1px
#555 }
h6 { font-size: 14px; margin: 0 0 5px 0; padding: 0; color: #333; }

.cleaner { clear: both; width: 100%; height: 0px; font-size: 0px;  }
.cleaner_h10 { clear: both; width:100%; height: 10px; }
.cleaner_h20 { clear: both; width:100%; height: 20px; }
.cleaner_h30 { clear: both; width:100%; height: 30px; }
.cleaner_h40 { clear: both; width:100%; height: 40px; }
.cleaner_h50 { clear: both; width:100%; height: 50px; }
.cleaner_h60 { clear: both; width:100%; height: 60px; }


.float_l { float: left; }
.float_r { float: right; }

.image_wrapper { display: inline-block; padding: 4px; border: 1px solid #fff; background:
none; margin-bottom: 10px; }
.image_fl { float: left; margin: 3px 15px 0 0; }
.image_fr { float: right; margin: 3px 0 0 15px; }

blockquote { font-style: italic; margin-left: 10px;}
cite { font-weight: bold; color:#3b3823; }
cite span { color: #696443; }
em { color: #000; }

.tmo_list { margin: 20px 0; padding: 0; list-style: none; }
```

```
.tmo_list li { background: transparent url(images/templatemo_list.png) no-repeat scroll 0 0px;
margin:0 0 20px; padding:0 0 0 25px;        line-height: 1em; }
.tmo_list li a { color: #000; }
.tmo_list li a:hover { color: #990; }


.button a { color: #000; font-weight: bold; }

.button a span { color: #000; font-size: 18px; }

.button a:hover { color: #900; text-decoration: none; }

#templatemo_wrapper {
	width: 970px;
	margin: 0 auto;
}

#templatemo_header {
	width: 910px;
	height: 160px;
	padding: 0 30px;
	margin: 50px 0 20px;
	background: url(images/templatemo_header.png) no-repeat bottom center;
}

#templatemo_header  #site_title {
	float: left;
	width: 210px;
	padding: 10px 0 0 30px;
}

#templatemo_header  #site_title a {
	margin: 0px;
	padding: 0px;
	font-size: 40px;
	color: #ffffff;
	font-weight: bold;
	text-decoration: none;
}

#templatemo_header  #site_title a span {
	display: block;
	font-size: 14px;
	color: #333;
	font-weight: normal;
	margin-top: 10px;
	margin-left: 5px;
}

#header_right {
	float: right;
	width: 660px;
}

#social_box {
	float: right;
	display: inline-block;
	margin: 20px 0 10px;
	font-size:1.5em;
	padding: 0;
	list-style: none;
```

```
}

#social_box li {
	display: block;
	float: left;
	padding: 0;
	margin-left: 50px;
}

#social_box li a {
	width: 46px;
	height: 46px;
	margin-right: 5px;
}


/* menu */

#templatemo_menu {
	clear: both;
	float: right;
	display: inline-block;
}

#templatemo_menu ul {
	display: inline-block;
	height: 75px;
	margin: 0;
	padding: 0;
	list-style: none;
}

#templatemo_menu ul li {
	padding: 0;
	margin: 0;
	display: inline-block;
}

#templatemo_menu ul li a {
	float: left;
	display: block;
	height: 60px;
	width: 120px;
	padding: 15px 0 0 0;
	font-size: 11px;
	color: #bc7682;

	text-align: center;
	text-decoration: none;
	font-weight: normal;
	outline: none;
	border: none;
}

#templatemo_menu ul li a span {
	display: block;
	font-size: 20px;
	font-weight: normal;
	color: #bc7682;
}
```

```
#templatemo_menu ul li .last {
	background: none;
}

#templatemo_menu ul li a:hover, #templatemo_menu ul .current {
	color: #6e202c;
	background: url(images/templatemo_menu_hover.png) center center no-repeat;
}

 #templatemo_menu ul li a:hover span, #templatemo_menu ul .current span {
	 color: #6e202c;
 }

#templatemo_menu ul li a span:hover {
	color: #6e202c;
}
/* end of menu */

/* content */

#templatemo_content_wrapper {
	clear: both;
	width: 970px;
}

#templatemo_content_top {
	width: 970px;
	height: 20px;
	background: url(images/templatemo_content_top.png) no-repeat center bottom;
}

#templatemo_content_bottom {
	width: 970px;
	height: 20px;
	background: url(images/templatemo_content_bottom.png) no-repeat center top;
}

#templatemo_content {
	padding: 10px 30px;
	background: url(images/templatemo_content.png) repeat-y center;
}


#templatemo_content p {
	margin-bottom: 10px;
}

.col_w425 {
	width: 425px;
}

.col_w270 {
	float: left;
	margin-right: 50px;
	width: 270px;
}

.col_w590 {
	width: 590px;
	margin-right: 50px;
}
```

```
.col_last {
	margin-right: 0;
}

.sb_news_box {
	margin-bottom: 20px;
	padding: 10px 0 20px 0;
	border-bottom: 1px solid #fff;
}

.sb_news_box img {
	float: left;
	width: 80px;
	height: 80px;
	border: 1px solid #FFF;
	padding: 2px;
}

.sb_news_box .nb_right {
	float: right;
	width: 170px;
	margin-bottom: 0;
	padding-bottom: 0;
}

.sb_news_box .nb_right a {
	text-decoration: none;
}

.sb_news_box .nb_right a h6 {
	color: #6d232e;
	text-decoration: underline;
}

.news_box {
	clear: both;
	margin-bottom: 30px;
	padding-bottom: 30px;
	border-bottom: 1px dashed #fff;
}

.news_box .news_meta {
	margin-bottom: 10px;
}

.news_box img {
	float: right;
	width: 240px;
	height: 180px;
	margin-left: 30px;
	margin-bottom: 10px;
	border: 1px solid #fff;
	padding: 4px;
}

.news_box h2 {
	margin-bottom: 10px;
}

.news_box_last {
```

```
	margin: 0;
	padding: 0;
	border: none;
}


.service_box {
	clear: both;
	padding-bottom: 30px;
	margin-bottom: 30px;
	border-bottom: 1px dashed #fff;
}

.service_box img {
	float: left;
	width: 48px;
	height: 48px;
}

.service_box .sb_right {
	float: right;
	width: 500px;
}

.comment{
	font-weight: bold;
}

/* end of content */

/* sp_box */

#templatemo_sp_box {
	clear: both;
	width: 910px;
	padding: 20px 30px;
	height: 118px;
	margin-bottom: 20px;
	background: url(images/templatemo_sp_box.png) no-repeat center top;
}

#templatemo_sp_box img {
	float: left;
	width: 60px;
	height: 60px;
}

#templatemo_sp_box .right {
	float: right;
	width: 200px;
}

#templatemo_sp_box p {
	margin-bottom: 15px;
}

#templatemo_sp_box .button a span { color: #000; }
#templatemo_sp_box .button a { color: #000;  }
#templatemo_sp_box .button a:hover { color: #900; }

/* end of sp_box */
```

```
#contact_form {
	float: left;
	padding: 0;
}

#contact_form form {
	margin: 0px;
	padding: 0px;
}

#contact_form form .input_field {
	width: 270px;
	padding: 5px 0;
	background: #fff;
	border: 1px solid #666;
	padding: 5px;
}

#contact_form form label {
	display: block;
	width: 100px;
	margin-bottom: 2px;
	font-size: 14px;
}

#contact_form form textarea {
	width: 270px;
	height: 200px;
	background: #fff;
	border: 1px solid #666;
	padding: 5px;
}

#contact_form form .submit_btn {
	background: #fff;
	border: 1px solid #666;
	padding: 8px 15px;
}

#templatemo_footer {
	width: 910px;
	padding: 10px 30px;;
	text-align: center;
	color: #000;
}

#templatemo_footer a {
	color: #000;
}
```

Experiment: 5

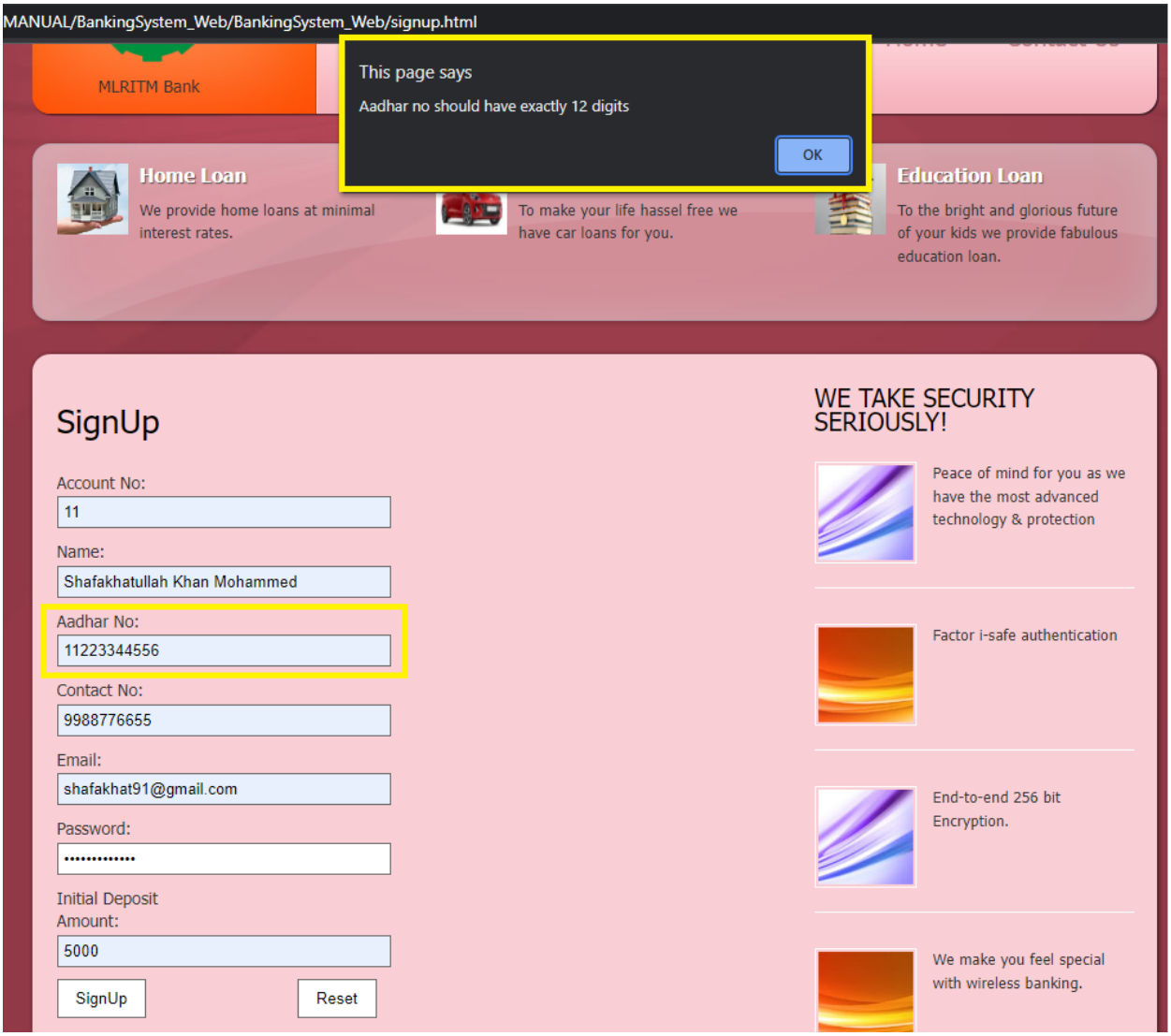
VALIDATE BANK WEB APPLICATION using [JAVASCRIPT]

Validating Banking System web application using JavaScript.

Signup form validation using JavaScript:

```
<script>
    function validate(){
        var accno = document.getElementById("accno").value;
        var name = document.getElementById("cname").value;
        var aadhar = document.getElementById("ad_no").value;
        var contact = document.getElementById("c_no").value;
        var email= document.getElementById("email").value;
        var password =  document.getElementById("password").value;
        var deposit = document.getElementById("d_amt").value;
        deposit = parseInt(deposit);
        if(aadhar.length == 12){
            if(contact.length == 10){
                if(password.length >=6 || password.length <=20){
                    if(deposit >=5000){
                        document.getElementById("signup").action = "login.html";
                    }else{
                       window.alert("Minimum Deposit amount is Rs. 5000");
                    }
                }else{
                    window.alert("Password should be minimum 6 characters and maximum 20");
                }
            }else{
                    window.alert("Contact no should have exactly 10 digits");
            }
        }else
            window.alert("Aadhar no should have exactly 12 digits");
    }
</script>
```

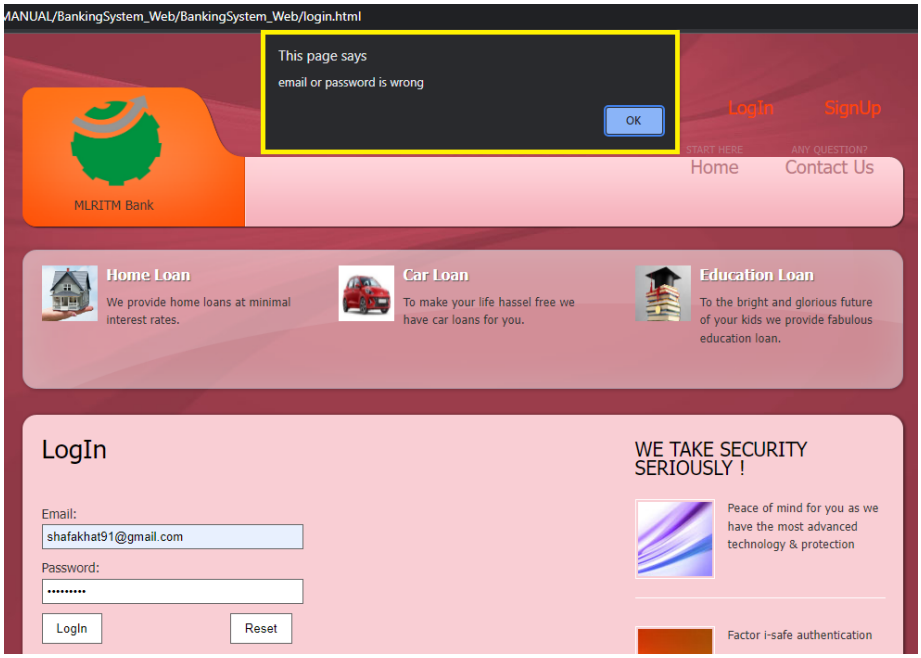
Output:


MANUAL/BankingSystem_Web/BankingSystem_Web/signup.html
MLRITM Bank
This page says
Aadhar no should have exactly 12 digits
OK
Home Loan
We provide home loans at minimal interest rates.
To make your life hassel free we have car loans for you.
Education Loan
To the bright and glorious future of your kids we provide fabulous education loan.
SignUp
Account No:
11
Name:
Shafakhatullah Khan Mohammed
Aadhar No:
11223344556
Contact No:
9988776655
Email:
shafakhat91@gmail.com
Password:
Initial Deposit Amount:
5000
SignUp
Reset
WE TAKE SECURITY SERIOUSLY!
Peace of mind for you as we have the most advanced technology & protection
Factor i-safe authentication
End-to-end 256 bit Encryption.
We make you feel special with wireless banking.

Login form validation using JavaScript:

```
<script>
    function logval(){
        var email= document.getElementById("email").value;
        var password =  document.getElementById("password").value;
        if(email == "shafakhat@shafakhat.com" && password == "Shafakhat@123")
            document.getElementById("login").action = "profile.html";
        else
            window.alert("email or password is wrong");
    }
</script>
```

Output:



Experiment: 6

INSTALLING AND CONFIGURING SPRING BOOT IN ECLIPSE

Install and configure servers (open source) Apache tomcat or related spring boot server

Installation and configuration of Sprint Boot in Eclipse using any of the following three methods:

Methods:

1. Using the Eclipse Marketplace
2. Installing the Spring Tools Suite Distribution of Eclipse and
3. Creating a Spring Boot Project.

Java Spring Framework is an open-source framework that is used for creating enterprise-grade, stand-alone applications that run on the Java Virtual Machine. As useful as it is, Java Spring Framework takes a lot of time and knowledge to set up, and deploy. Spring Boot makes this process easier using auto configuration, and an opinionated approach that allows Spring Boot to decide which dependencies and packages are right for your project. Spring Boot helps developers that run on their own without relying on an external web server. On Eclipse, Spring Boot is referred to as Spring Tools Suite. In this practical you will learn how to install and configure Spring Tools Suite.

Method 1: Using the Eclipse Marketplace

Step 1:

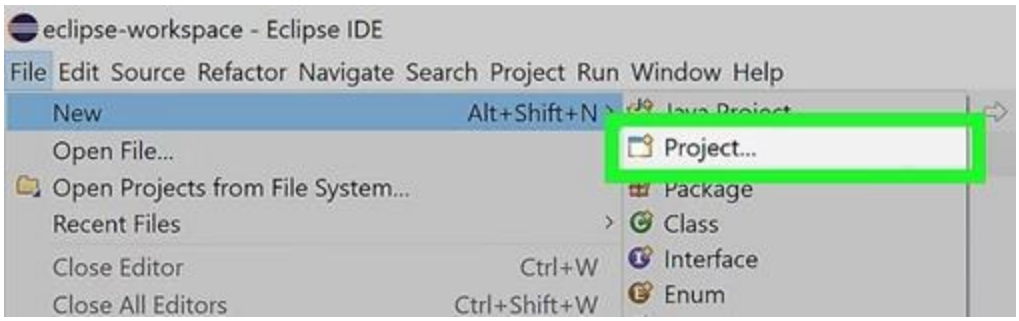
Launch Eclipse. Eclipse has an icon that resembles a blue circle with white horizontal lines and a yellow crescent moon to the left. Click the icon on your desktop, Windows Start menu, Applications folder (Mac), or Apps menu (Linux) to open Eclipse.

The first time you open Eclipse, you will need to select a folder to use as your workspace. Click Launch in the lower-right corner to use the default workspace folder. Click Browse to select a different location.

Step 2:

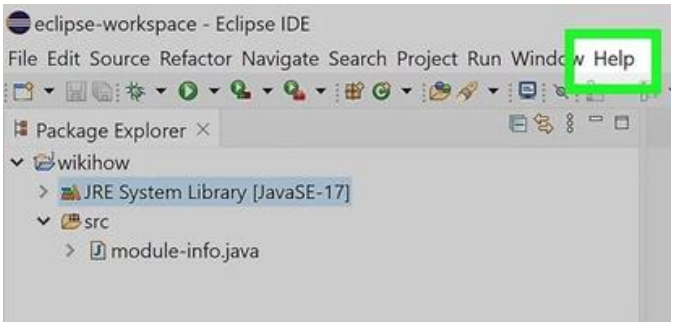
Open or create a new project. By default, Eclipse will open the last project you were working on. To create a new project, click File in the menu bar at the top, and then click New. To open an existing project, click File in the menu bar, and then click Open. Select a file and click Open.

The first time you open Eclipse, a screen will appear giving you a variety of options. Click the option to open a new Java project to start a new project. Alternatively, you can click the option to open an existing project to begin work on an existing project.



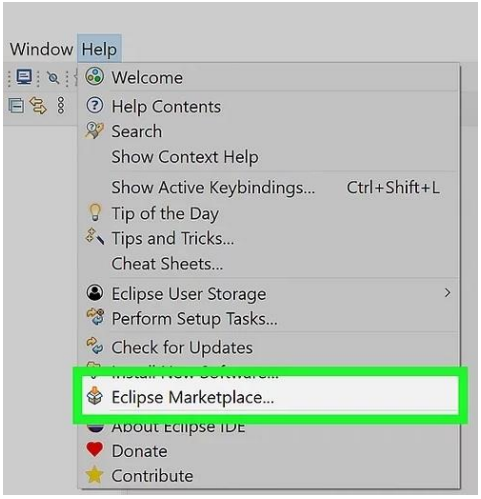
Step 3:

Click Help. It's the last option in the menu bar at the top of the screen. This displays the Help menu.



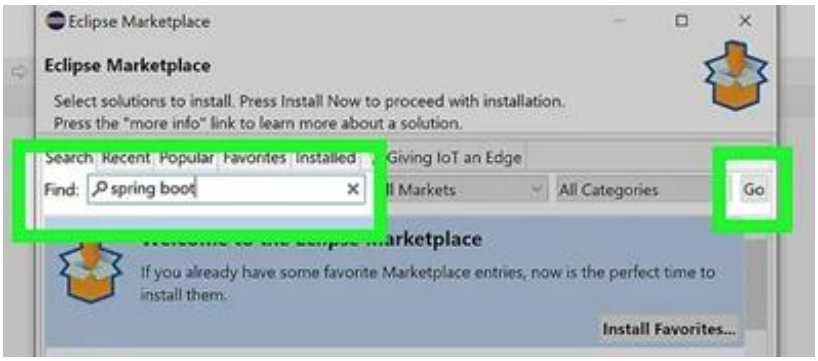
Step 4:

Click Eclipse Marketplace. It's near the bottom of the Help menu. This opens the Eclipse Marketplace in a new window.



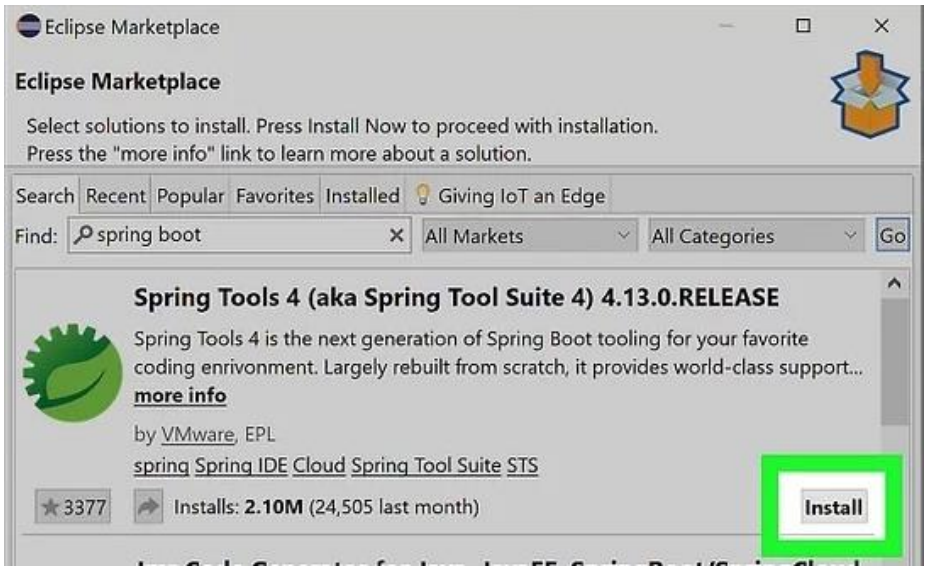
Step 5:

Type Spring Boot in the search box and press ↵ Enter. This displays a list of search results related to Spring Boot. On Eclipse, Spring Boot is called Spring Tools Suite.



Step 6:

Click Install below the latest version of Spring Tools. The latest version of Spring Tools should appear at the top of the list. Click the Install button in the lower-right corner of the box. This will display a checklist of all the packages that will be installed.



Step 7:

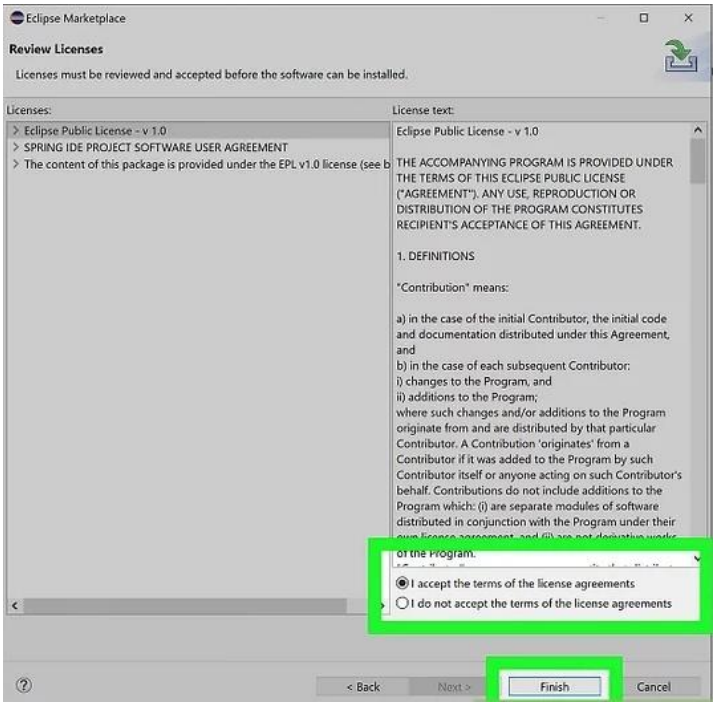
Click Confirm. It's at the bottom of the Eclipse Marketplace window. This confirms that you want to install all the packages that are checked in the list. This begins the process of installing the selected packages.

If there are any packages you do not want to install, uncheck them before clicking "Confirm."

If you want to install any additional plug-ins, such as the add-on for previous versions of Spring Tools, click Install More at the bottom of the Eclipse Marketplace window. Then click Install below any additional plug-ins you want to install. Click Install Now at the bottom of the Eclipse Marketplace window, when you are ready to install all the selected plug-ins.

Step 8:

Agree to the Terms and Conditions and click Finish. Click the radio button next to "I accept the terms of the license agreements" and click Finish.



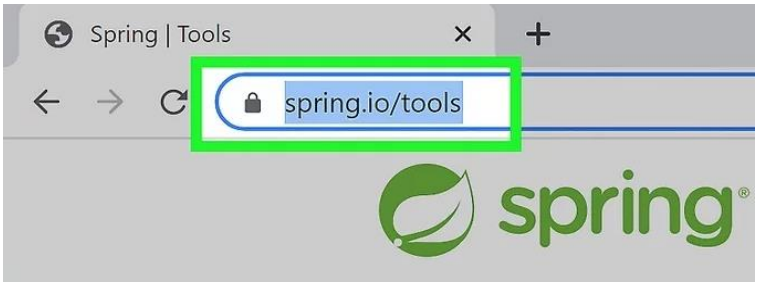
Step 9:

Relaunch Eclipse. After installing Spring Tools in Eclipse, close Eclipse and then launch it again. Spring Tools is now installed and ready to use.

Method 2: Installing the Spring Tools Suite Distribution of Eclipse

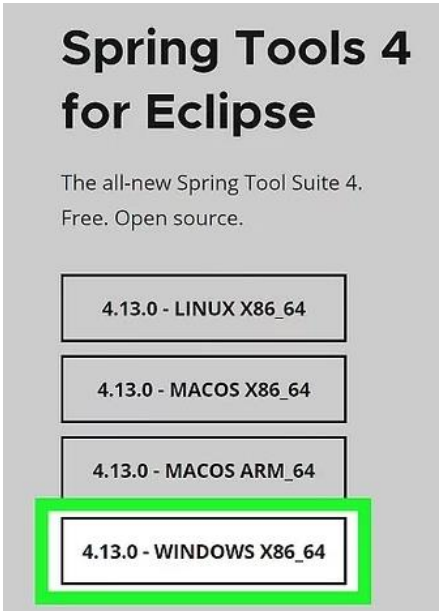
Step 1:

Go to https://spring.io/tools in a web browser. This is the website where you can download Spring Tools Suite. Spring Tools Suite is an Eclipse distribution that comes with Spring Tools (Spring Boot) already pre-installed.



Step 2:

Click the Spring Tools for Eclipse download link for your operating system. There are four download links below "Spring Tools for Eclipse." Click the Windows x86_64 link if you are using Windows. Click the MacOS x86_64 link if you are using an Intel-based Mac. Click MacOS ARM_64 if you are using an ARM-based Mac. Click Linux x86_64 if you are using Linux.



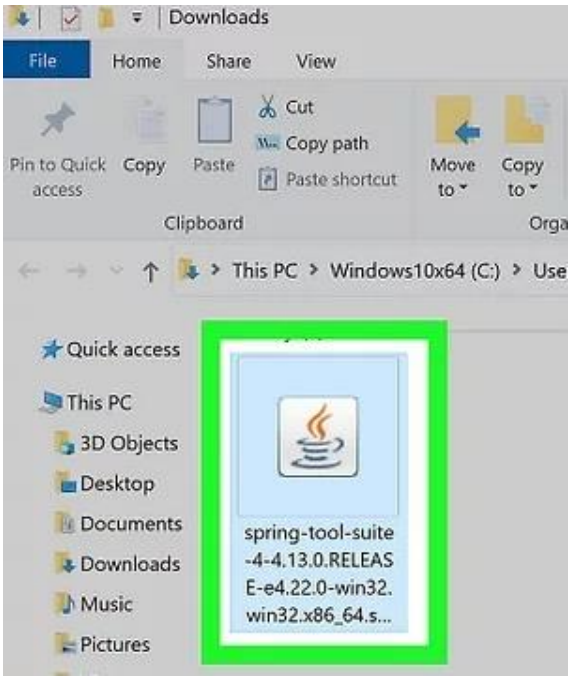
Step 3:

Open the installation file. Click the installation file in your web browser or Downloads folder. This will install Spring Tools Suite automatically. The way it does this is different, depending on which operating system you are using.

If you are using Windows, you'll need to install the latest version of Java in order to install Spring Tools Suite. The JAR installation file will install Spring Tools Suite at whichever location you launch the file from. You may want to copy and paste the installation file to whichever location you want to install Spring Tools Suite at before launching the file.

If you are using a Mac, open the installation DMG file. Spring Tools Suite will start installing automatically. Be sure to drag the Spring Tools Suite app icon to the Applications folder once the installation is complete.

If you are using Linux, you will need to extract the contents of the downloaded tar.gz file to the location you want to install Spring Tools Suite at. It contains the Spring Tools Suite executable file.

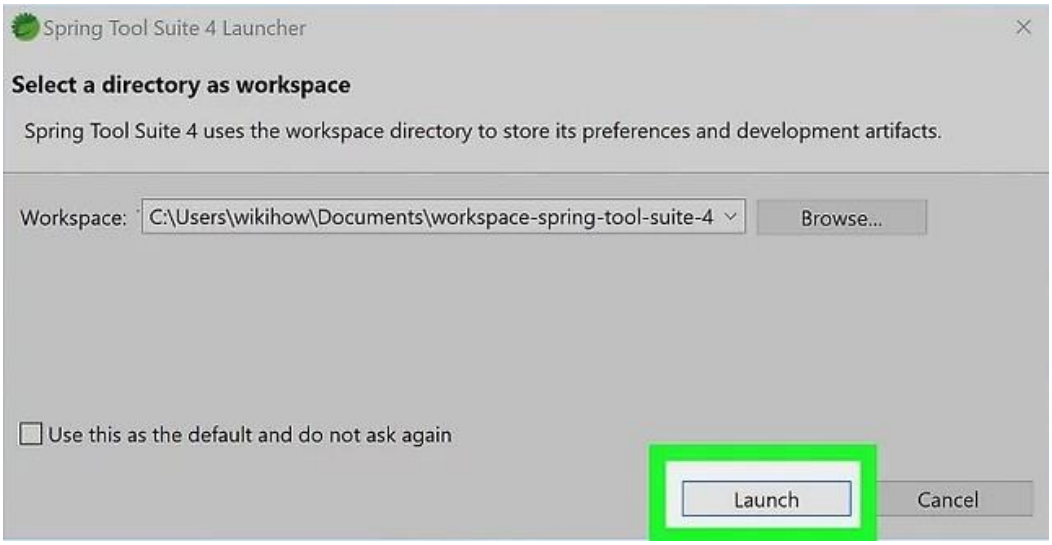


Step 4:

Launch Spring Tools Suite. If you are using Windows or Linux, navigate to the folder that you installed or extracted the Spring Tools Suite installation file to. Then click the Spring Tools Suite executable file. If you are using Mac, navigate to the Applications folder and click the Spring Tools Suite app file.

Step 5:

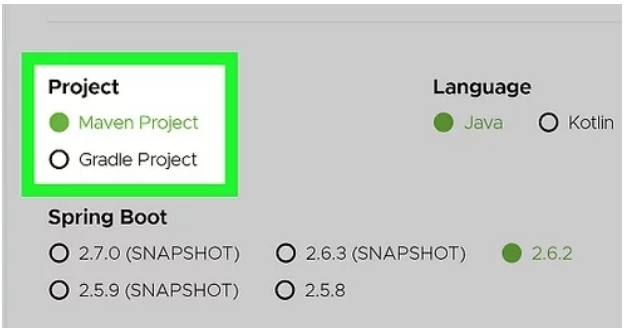
Select a workspace folder. The first time you run Eclipse, you will need to select a workspace folder. Click Launch to use the default workspace folder location. If you want to select a different location to use as your workspace folder, click Browse and select the location you want to use as your workspace. Then click Open. This launches Eclipse with Spring Tools (Spring Boot) already installed.



Method 3: Creating a Spring Boot Project

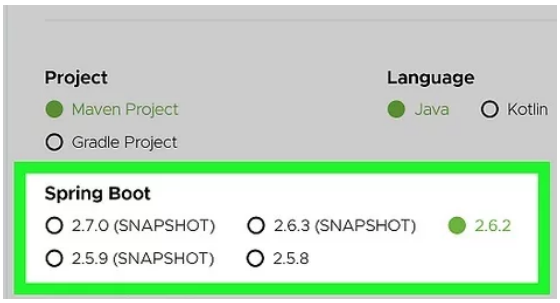
Step 1: Go to https://start.spring.io/ in a web browser. This website allows you to enter the information and select the dependencies of your Spring project. It will then generate a zip file containing all the files needed to start your Spring project.

Step 2: Select which build tool you want to use. Click the radio option next to which build tool you want to use below "Project" in the upper-left corner. You can select a Maven project or a Gradle project.



Step 3: Select which programming language you want to use. Click the radio option next to which programming language you want to use in the upper-left corner. You can select Java, Kotlin, or Groovy.

Step 4: Select which version of Spring Boot you are using. Click the radio option next to the version of Spring Boot you are using. If you are not sure, use the default setting.



Step 5: Enter your project metadata. Use the form below "Metadata" to enter your project's metadata. You'll need to provide the following information:

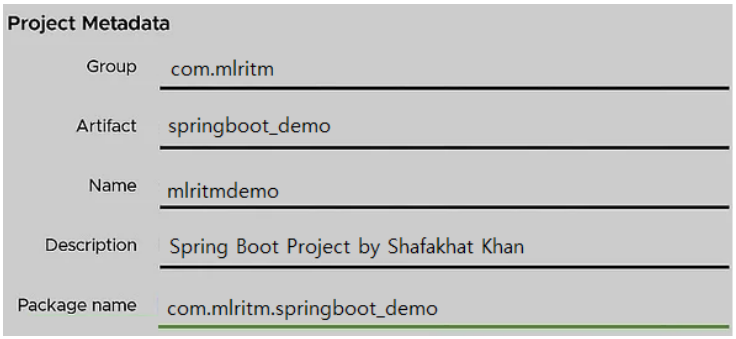
Group: This will be the groupID attribute for your project.

Artifact: This is generally the name of the project. This will be the artifactID attribute

Name: This is usually the same as the Artifact name. If the name of your project is different than the artifactID, you can enter the name here.

Description: Use this space to enter a brief description of your project.

Package name: This is the root package name. This is generally the same as the Group name. If you want to use a different root package name, you can enter it here.



Step 6: Select your project packaging. You can package your project as a JAR file or a WAR file:

JAR: JAR files are self-contained, executable Java programs. They can contain compiled Java code, manifest files, XML configuration data, JSON configuration data, as well as images and audio.

WAR: WAR files contain files related to a web project. They may contain XML, JSP, HTML, CSS, and JavaScript files that can be deployed on any servlet.

Step 7: Select which version of Java you are using. Click the radio option next the version of Java you want to use. You can use java 8, Java 11, or Java 17.

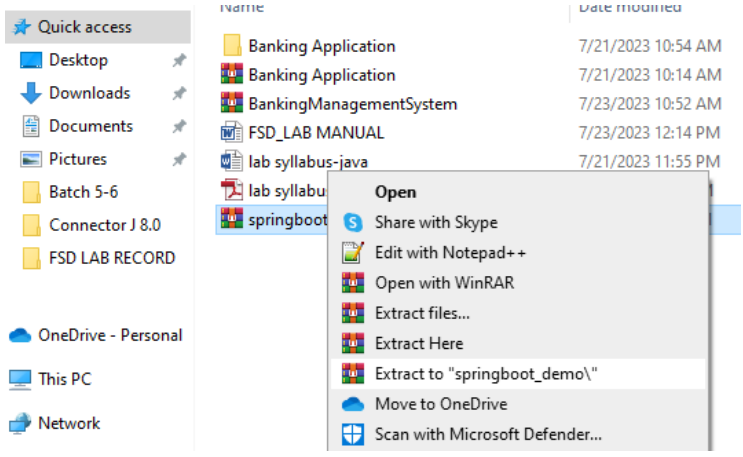
Step 8: Add dependencies to your project. Spring Boot allows you to add a variety of dependencies to your project. Use the following steps to add dependencies to your project:

- Click Add Dependencies in the upper-right corner.
- Use the search bar at the top to search for dependencies.
- Click a dependency to add it.

Step 9: Click Generate. It's in the lower-left corner at the bottom of the screen. This will generate and download a zip file containing all the files needed to start your Spring Boot project in Eclipse.

Alternatively, you can click Explorer to view all the different files in your project. You can view the source code and download individual files. Click Share to get a link to your project that you can copy and send to other people to view.

Step 10: Extract the zip file. Once you download the zip file with your project files extract it. It's best to extract it to your workplace folder or a location you will remember.

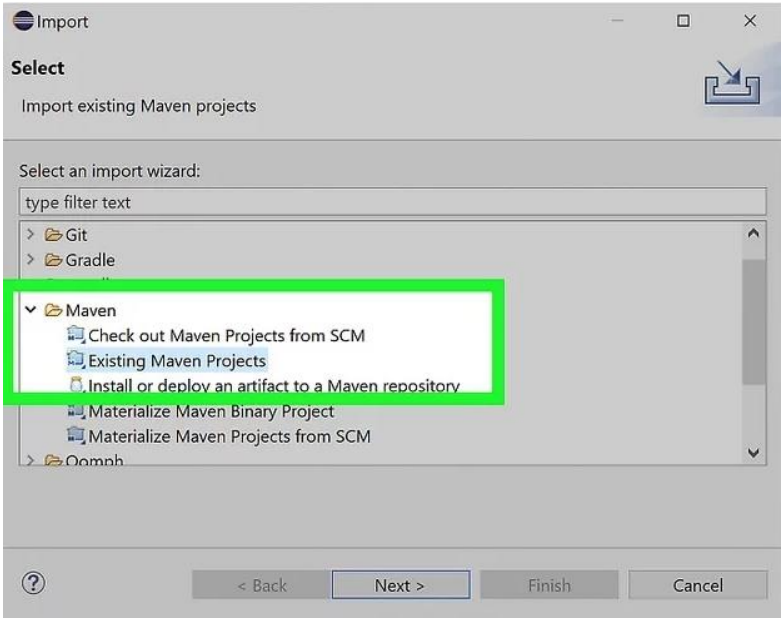


Step 11: Open Eclipse. Make sure you have Spring Boot installed in Eclipse or you are using the Spring Tools distribution of Eclipse. Click the Eclipse icon to launch Eclipse.

If Eclipse doesn't automatically open to your project, you'll need to open it, or create a new project.

Step 12: Import your Spring Boot project. This will import the Spring Boot files you created, downloaded, and extracted into Eclipse so that you can begin coding your project. Use the following steps to import your Spring Boot project files:

- Click File in the menu bar at the top.
- Click Import.
- Expand the "Maven" or "Gradle" folder.
- Click Existing Maven Project or Existing Gradle Project
- Click Next.
- Click Browse in the upper-right corner.
- Select the folder containing the project files that you downloaded and extracted.
- Click Open.
- Click Finish.



Experiment: 7

INSTALLING AND CONFIGURING HIBERNATE TOOLS IN ECLIPSE

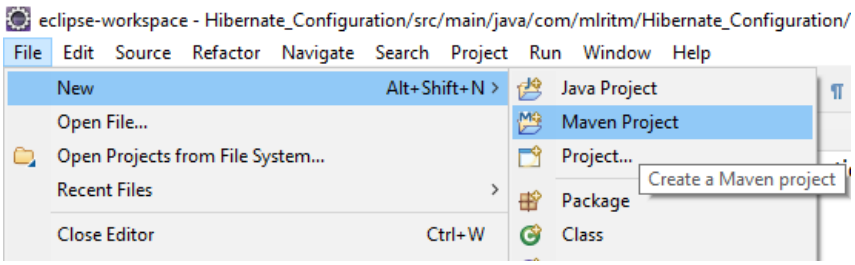
Install and configure databases using Database drivers or hibernate and create customer table in the database using hibernate.

Table should contain the following fields:

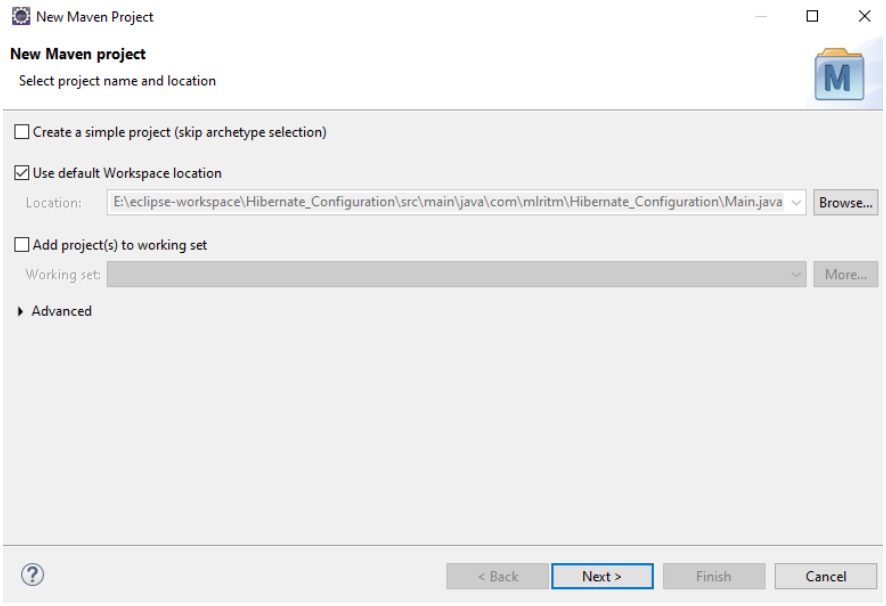
- *account_no (Primary Key)*
- *customer_name*
- *contact*
- *emailId*
- *password*
- *address*
- *withdraw*
- *deposit*
- *balance*

To configure hibernate in eclipse follow the steps below:

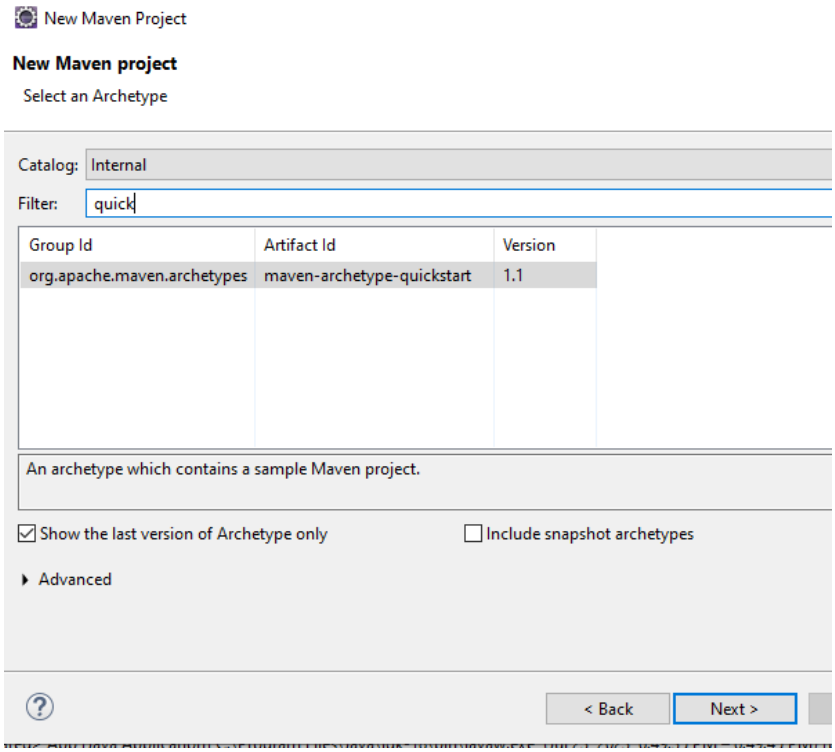
1. Open Eclipse IDE and Create a new Maven Project



2. **Select Project Name and Location** windows appears

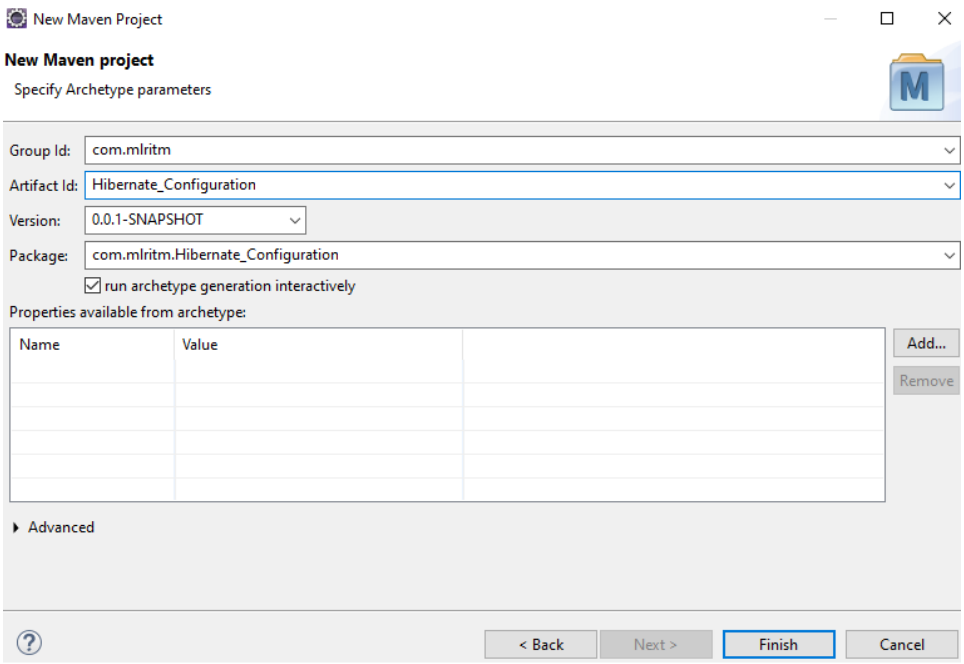


3. Click Next> button and need to **Select the Archetype**

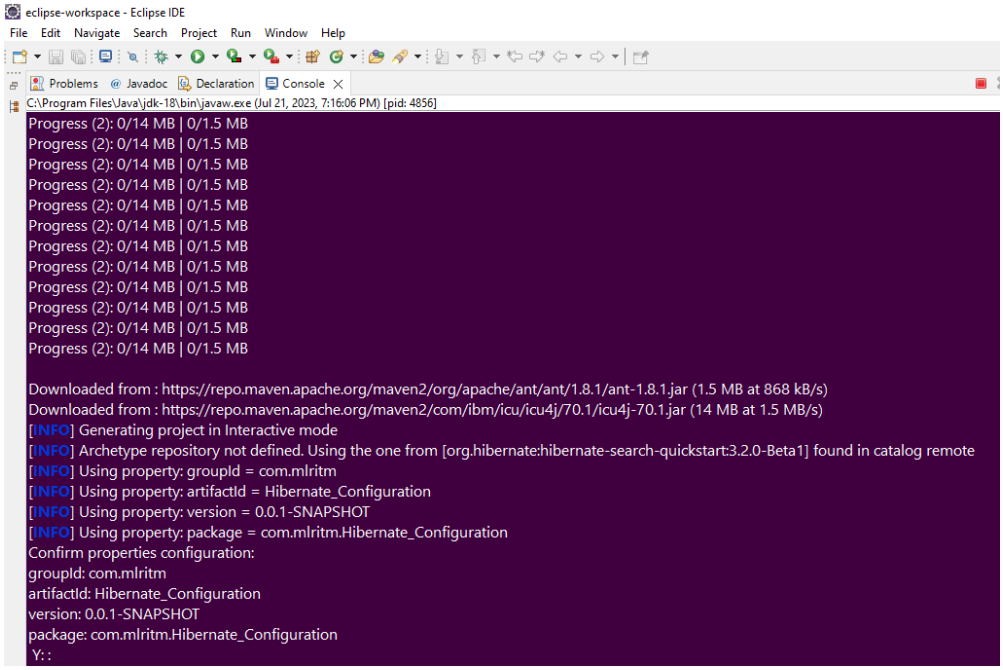


From the hibernate filter list, select Artifact Id **"hibernate-search-quickstart"** and click Next> button.

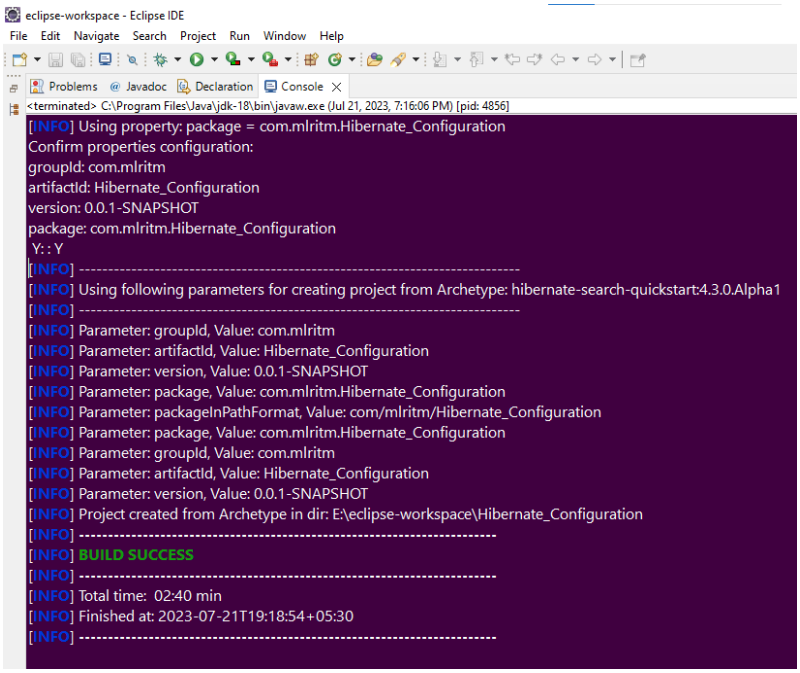
4. Enter Group Id as **com.mlritm**, Artifact Id (Project Name) and hit Finish button.



5. Maven will do the rest for you to download and add the hibernate libraries to Eclipse IDE. It will take few seconds wait for it.



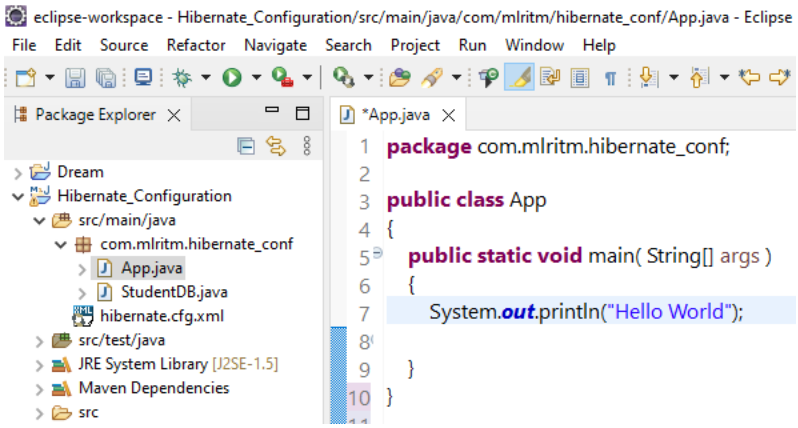
6. Open the console and hit capital letter **'Y'** to complete the download and add Hibernate libraries.



Once the build is success in the package explorer you can view the project you have created.

To implement hibernates concept we user App **class** in **src/main/java and** run the application.

App class:

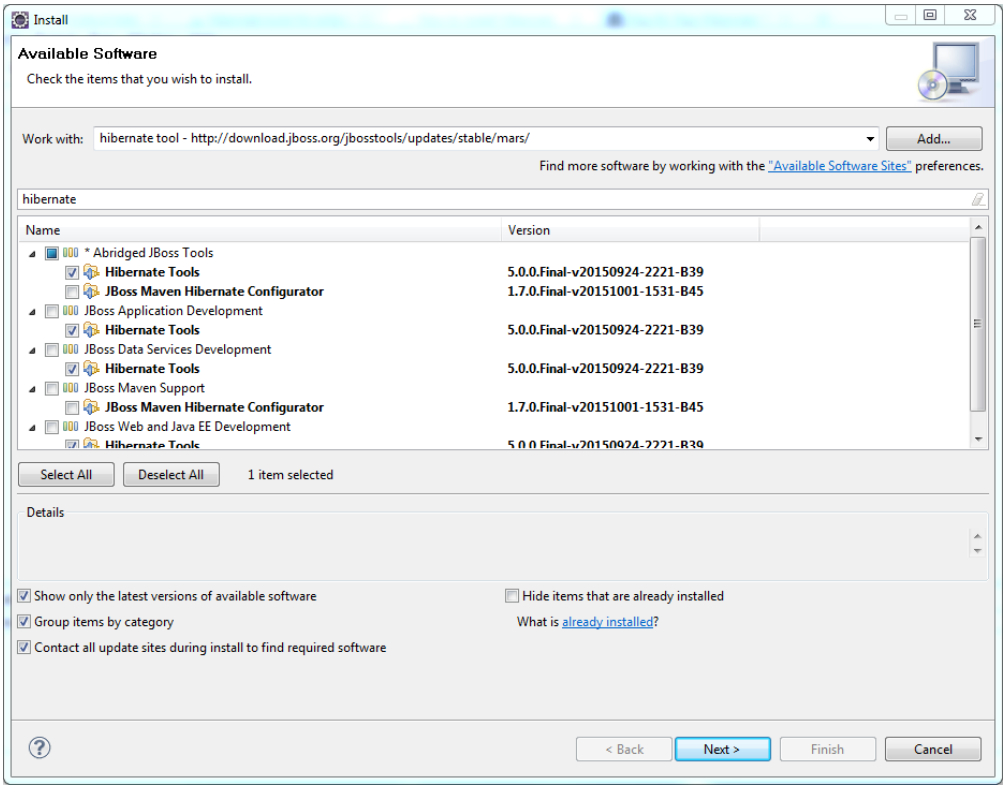


If the console prints **"Hibernate Framework…"** it means, we did the implementation in the correct way.

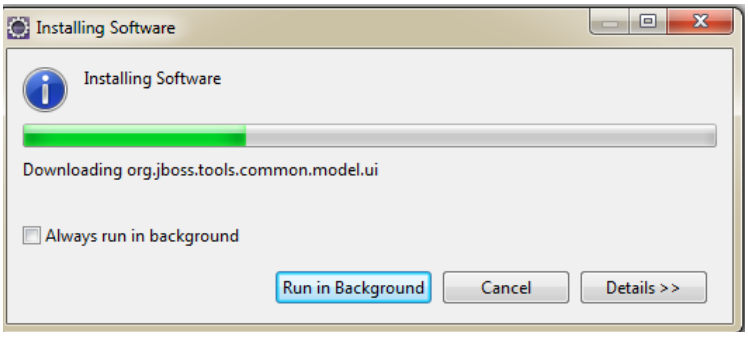
Hibernate Tool Installation in Eclipse IDE:

Method – 1

In Eclipse IDE, menu bar, select **Help >> Install New Software** … put the Eclipse update site URL "download.jboss.org/jbosstools/updates/stable/Eclipse_Version".



Select the tool and click on Next. Do not select all the tools; it will install all the unnecessary tools. **We just need hibernate tools**. Accept license agreement and click finish. It will take some minutes to complete installation process.

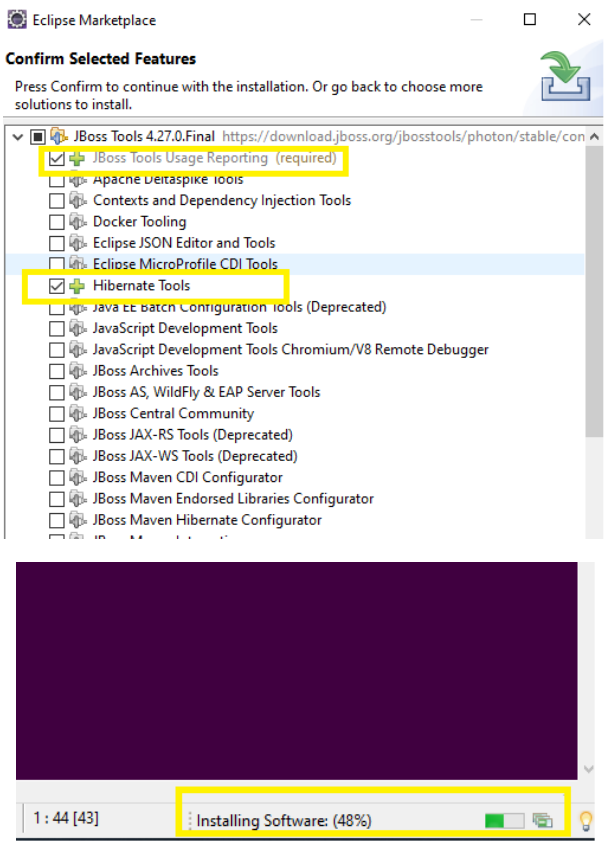


After installation, restart the eclipse to verify whether Hibernate tools is installed properly, we will look at Hibernate Perspective in **Eclipse->>Window->>Open Perspective->>Other**

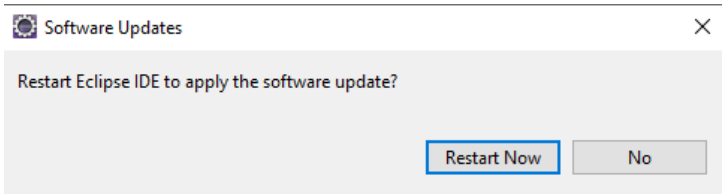
Method – 2

In Eclipse IDE, menu bar, select **Help >> Eclipse Marketplace**. Window appears in the find text box enter "hibernate" and click Go button and wait for few seconds to display hibernate tools. A list of hibernate tools will appear among them we need to select **JBoss Tools 4.27.0 Final**, and click the **install button** associated to it.

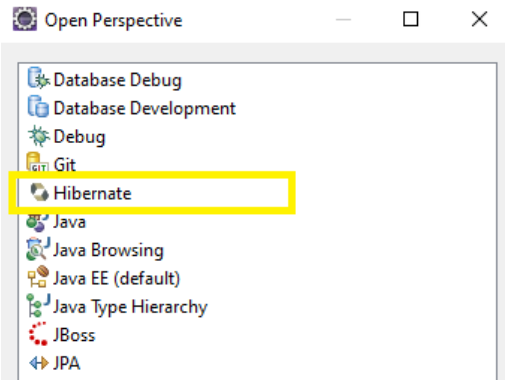
It contains so many tools in it we don't need to install all the tools. **We just need hibernate tools**. Click confirm button to continue with the installation process. Accept license agreement and click finish. It will take some minutes to complete installation process. The installation progress bar appears in the Eclipse IDE at the bottom right hand side.



After the completion of installation process restart Eclipse IDE popup appears, click the **Restart Now** button and let the IDE restart to integrate hibernate with Eclipse IDE.

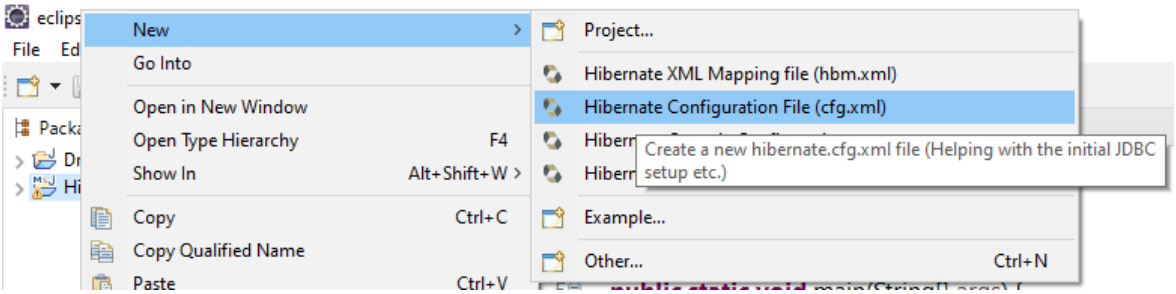


When the Eclipse IDE restarts to view the installed hibernate framework open the perspective window, where the installed framework appears.

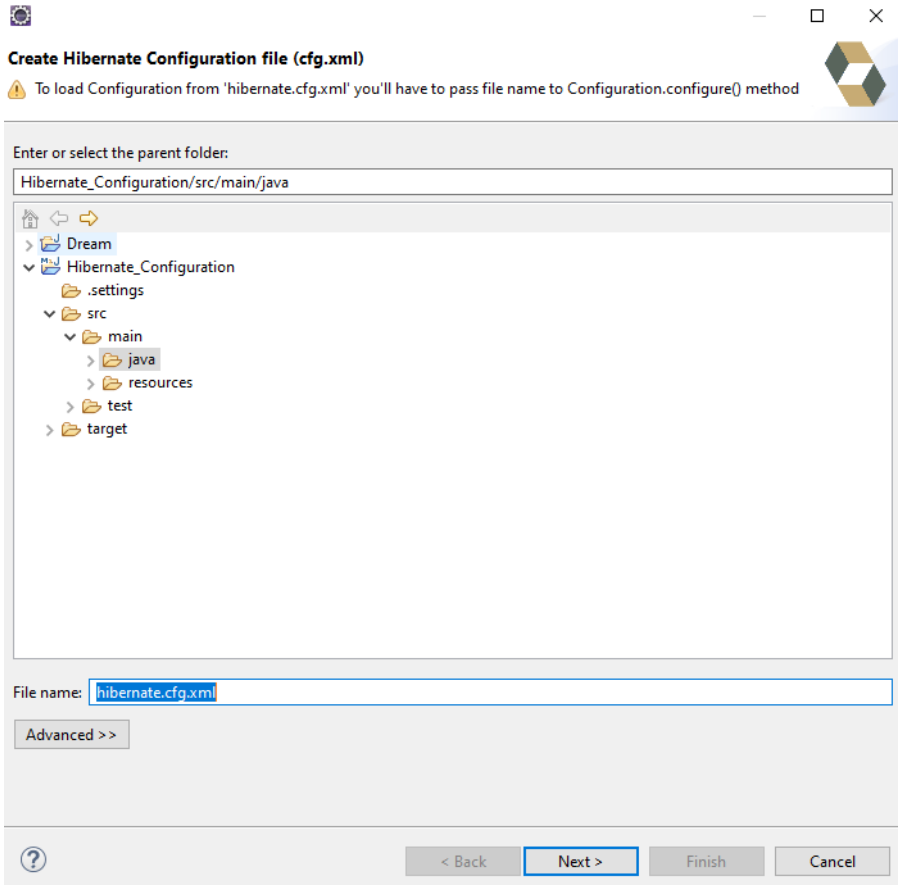


Configuring Database using Hibernate Configuration XML file:

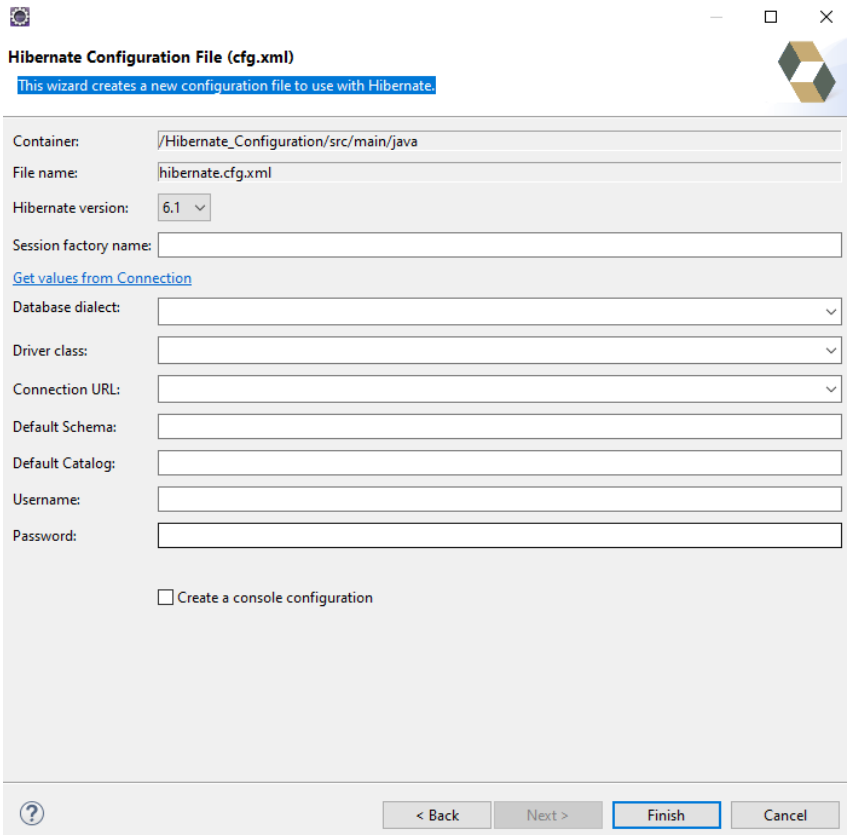
To configure hibernate with the project, select the project, do right click and select **New** menu item and then another popup window appears from that select **hibernate.cfg.xml** file.



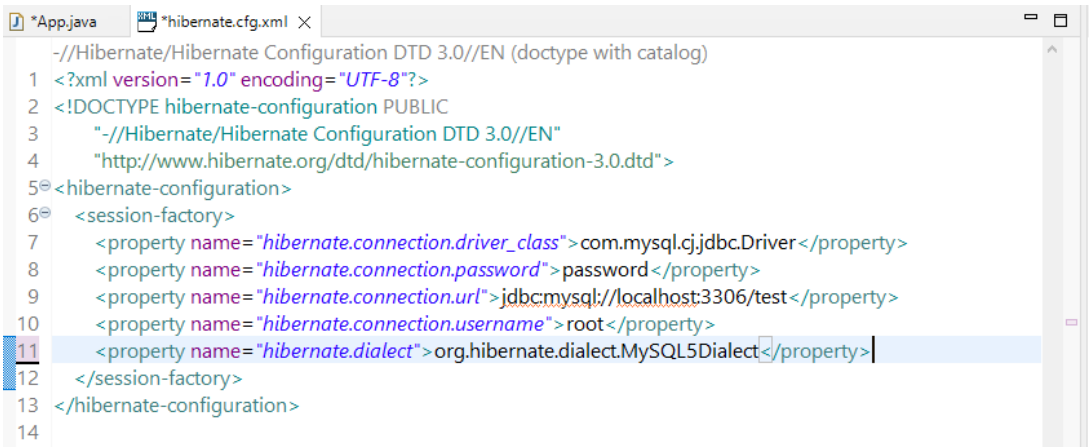
Create hibernate configuration file window will be displayed, set the **File Name** and click **Next** button.



This wizard creates a new configuration file to use with Hibernate window contains the fields related to Database Connectivity where the user needs to provide the details of the database in order to establish connection. Enter the values in the fields provided and click finish button.



Hibernate configuration XML file will be generated as shown below.



Creating Customer table in the database using Hibernate

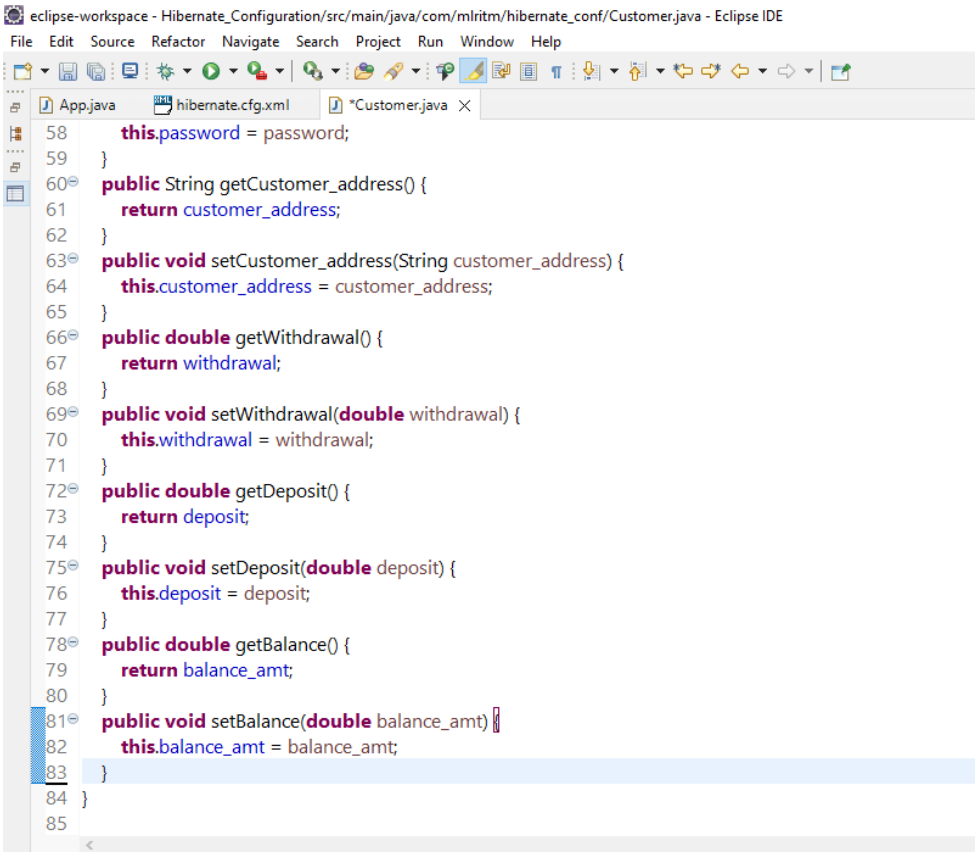
Create customer class in the package **"com.mlritm.hibernate_configuration"**

App.java | hibernate.cfg.xml | Customer.java

```java
package com.mlritm.hibernate_conf;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "student")
public class Customer {
   @Id
   private int account_no;
   @Column
   private String contact_no;
   @Column
   private String customer_name;
   @Column
   private String emailID;
   @Column
   private String password;
   @Column
   private String customer_address;
   @Column
   private double withdrawal;
```

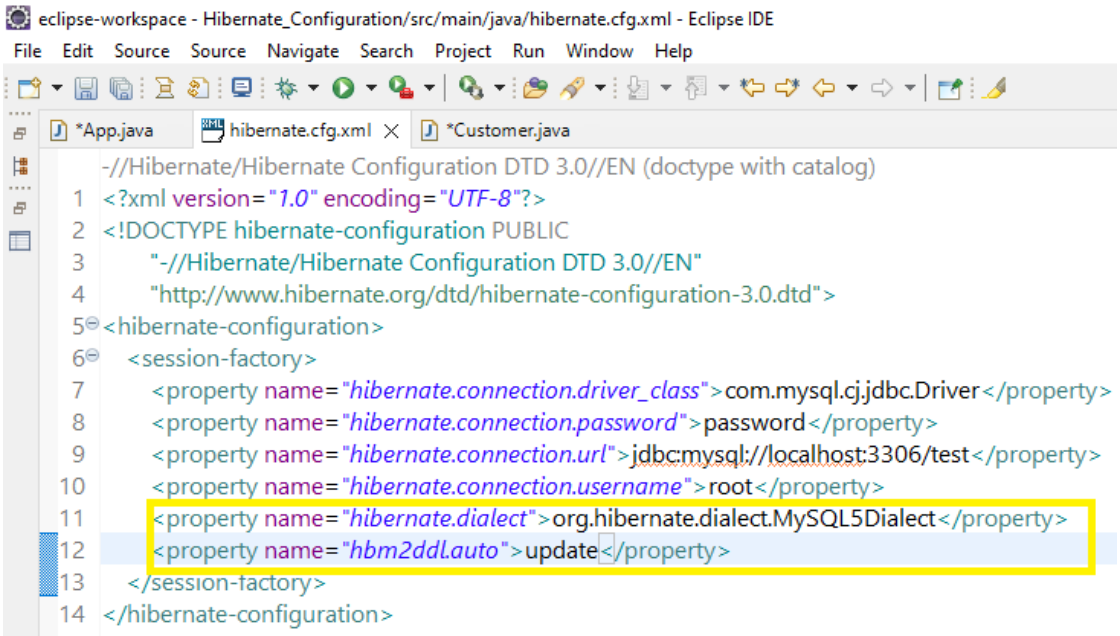
App.java | hibernate.cfg.xml | Customer.java

```java
   public int getAccount_no() {
      return account_no;
   }
   public void setAccount_no(int account_no) {
      this.account_no = account_no;
   }
   public String getContact_no() {
      return contact_no;
   }
   public void setContact_no(String contact_no) {
      this.contact_no = contact_no;
   }
   public String getCustomer_name() {
      return customer_name;
   }
   public void setCustomer_name(String customer_name) {
      this.customer_name = customer_name;
   }
   public String getEmailID() {
      return emailID;
   }
   public void setEmailID(String emailID) {
      this.emailID = emailID;
   }
   public String getPassword() {
      return password;
   }
   public void setPassword(String password) {
```



Write the following code in the App class to create customer table in the database automatically by the ORM and by setting few properties in the configuration file.

hibernate.cfn.xml



App class:

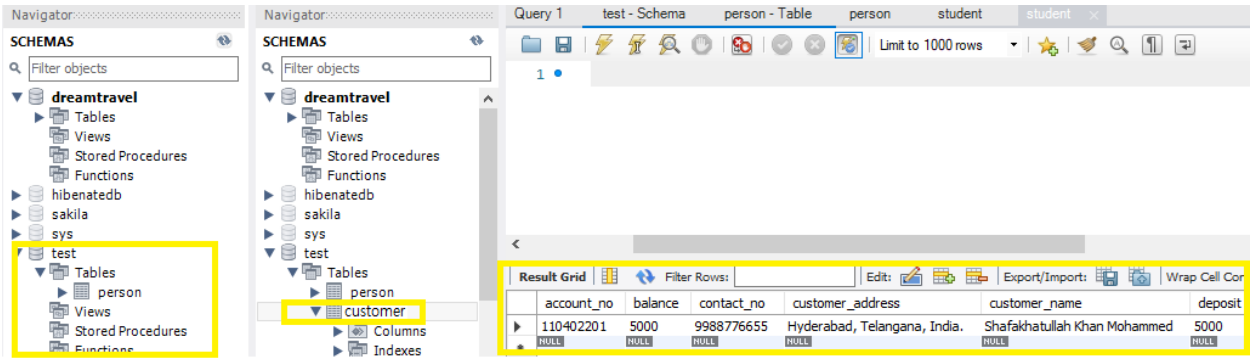
eclipse-workspace - Hibernate_Configuration/src/main/java/com/mlritm/hibernate_conf/App.java - Eclipse IDE

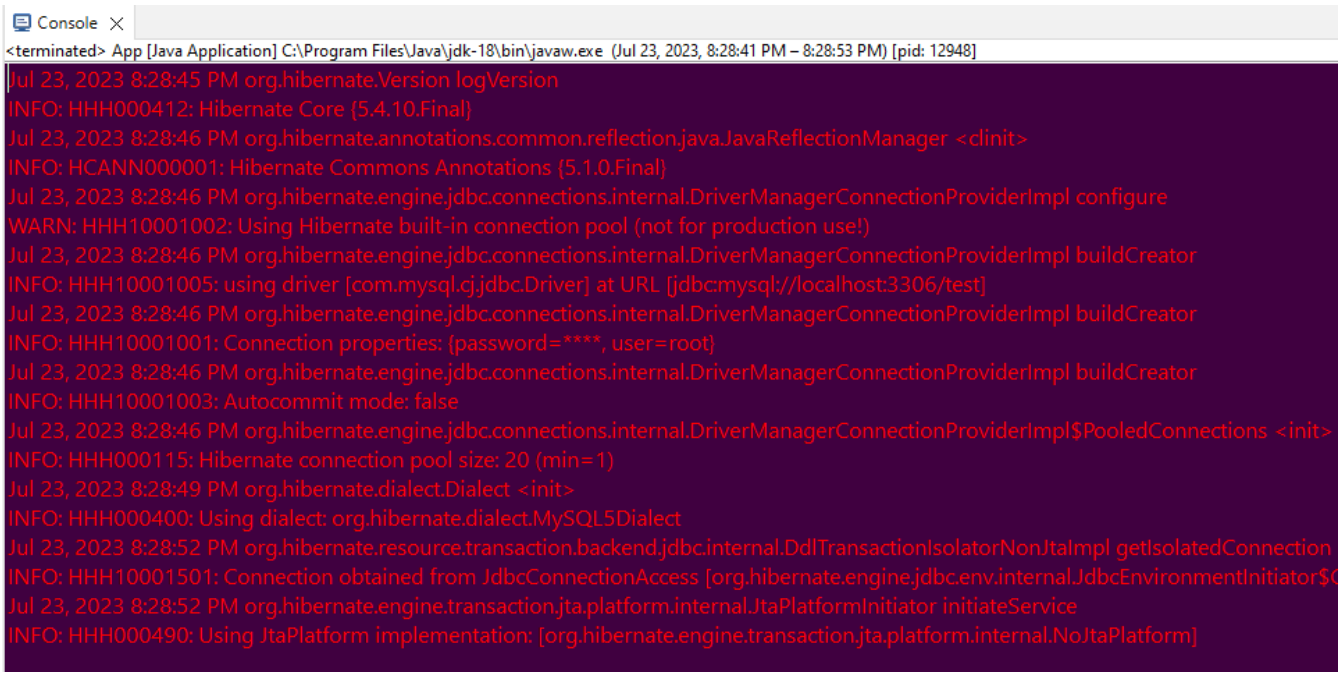
File Edit Source Refactor Navigate Search Project Run Window Help

*App.java | hibernate.cfg.xml | *Customer.java

```
3  import org.hibernate.*;
4  import org.hibernate.cfg.Configuration;
5
6  public class App {
7    public static void main( String[] args ){
8      //creating student object
9      Customer  cus = new Customer();
10     cus.setAccount_no(110402201);
11     cus.setCustomer_name("Shafakhatullah Khan Mohammed");
12     cus.setContact_no("9988776655");
13     cus.setEmailID("shafakhat@mlritm.ac.in");
14     cus.setCustomer_address("Hyderabad, Telangana, India.");
15     cus.setDeposit(5000);
16     cus.setWithdrawal(0);
17     cus.setBalance(5000);
18     try {
19       Configuration con = new Configuration().configure().addAnnotatedClass(Customer.class);
20       SessionFactory sf = con. buildSessionFactory();
21       Session ss = sf.openSession();
22       Transaction trn = ss.beginTransaction();
23       ss.save(cus);
24       trn.commit();
25     }catch(Exception e) {
26       System.out.println(e.getMessage());
27     }
28   }
29 }
30
```

Output:



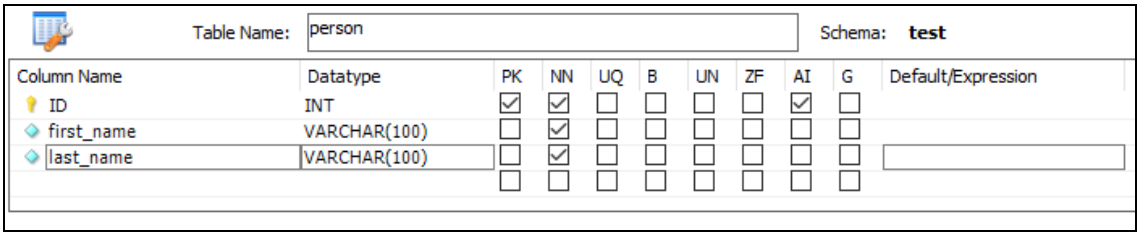

Console
<terminated> App [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe (Jul 23, 2023, 8:28:41 PM – 8:28:53 PM) [pid: 12948]
Jul 23, 2023 8:28:45 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {5.4.10.Final}
Jul 23, 2023 8:28:46 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
Jul 23, 2023 8:28:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Jul 23, 2023 8:28:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/test]
Jul 23, 2023 8:28:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {password=****, user=root}
Jul 23, 2023 8:28:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Jul 23, 2023 8:28:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
Jul 23, 2023 8:28:49 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
Jul 23, 2023 8:28:52 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$
Jul 23, 2023 8:28:52 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]

Experiment: 8

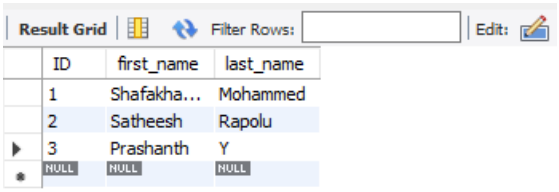
CONNECT JAVA TO A MYSQL DATABASE using JDBC [Maven Project]

Write a java program to Connect with MySQL database using JDBC drivers.

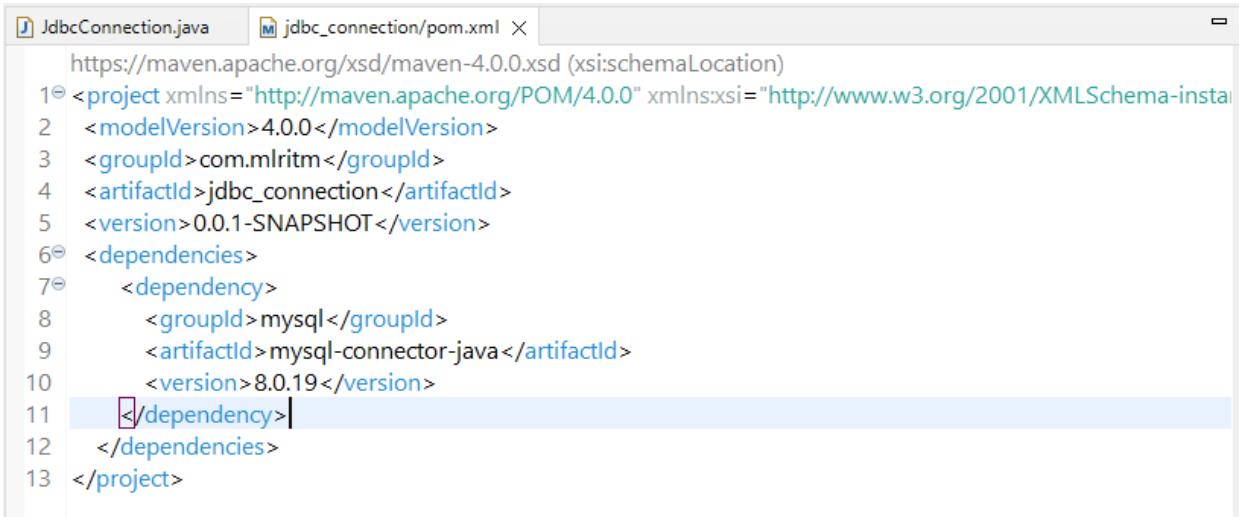
Create a Schema "Test" in MySQL and create the following table in the database "Test".



Once the table is created insert few records in the table.



We'll also need the **mysql-connector-java** artifact which as always is available from Maven Repository:



Connecting Using JDBC

JDBC (Java Database Connectivity) is an API for connecting and executing queries on a database.

Common Properties

During the course of this article, we'll typically use several common JDBC properties:

Connection URL – a string that the JDBC driver uses to connect to a database. It can contain information such as where to search for the database, the name of the database to connect to and other configuration properties:

```
jdbc:mysql://[host][,failoverhost...]
    [:port]/[database]
    [?propertyName1][=propertyValue1]
    [&propertyName2][=propertyValue2]...
```

We'll set this property like so: jdbc:mysql://localhost:3306/test

Driver class – the fully-qualified class name of the driver to use. In our case, we'll use the MySQL driver: com.mysql.cj.jdbc.Driver

Username and password – the credentials of the MySQL account

JDBC Connection Example

Let's see how we can connect to our database and execute a simple **select-all**.

JdbcConnection.java | jdbc_connection/pom.xml

```
1  package jdbc_connection;
2  import java.sql.*;
3
4  public class JdbcConnection{
5     public static void main(String...args) throws Exception{
6        String query = "SELECT * FROM person";
7        String url = "jdbc:mysql://localhost:3306/test";
8        try {
9           Class.forName("com.mysql.cj.jdbc.Driver");
10          Connection conn = DriverManager.getConnection(url, "root", "password");
11          PreparedStatement ps = conn.prepareStatement(query);
12          ResultSet rs = ps.executeQuery();
13          while (rs.next()) {
14             long id = rs.getLong("ID");
15             String name = rs.getString("FIRST_NAME");
16             String lastName = rs.getString("LAST_NAME");
17             System.out.println(id+"      "+name+" "+lastName);
18          }
19       } catch (SQLException e) {
20          System.out.println(e.getMessage());
21       }
22    }
23 }
```

Output:

Error Log | Console

```
<terminated> JdbcConnection [Java Application] C:\Program Files\Java\
ID   FIRST_NAME                LAST_NAME
-----------------------------------------------------
1    Shafakhatullah Khan                Mohammed
2    Satheesh                Rapolu
3    Prashanth                Y
```

Experiment: 9

CONNECT JAVA TO A MYSQL DATABASE using HIBERNATE (ORM)

Write a java program to Connect with MySQL database using hibernate.

Connecting Using ORM

More typically, we'll connect to our MySQL database using an Object Relational Mapping (ORM) Framework.

Native Hibernate APIs

Step 1:

We need to add the *hibernate-core* Maven dependency:

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.10.Final</version>
</dependency>
```

Step 2:

Hibernate requires that an entity class must be created for each table. Let's go ahead and define the *Person* class:

```
@Entity
@Table(name = "Person")
public class Person {
    @Id
    Long id;
    @Column(name = "FIRST_NAME")
    String firstName;

    @Column(name = "LAST_NAME")
    String lastName;

    // getters & setters
}
```

Step 3:

Hibernate resource file, typically named *hibernate.cfg.xml*, where we'll define configuration information:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
```

```
        <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/test </property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>

        <!-- Validate the database schema on startup -->
        <property name="hbm2ddl.auto">validate</property>

        <!-- Names the annotated entity class -->
        <mapping class="Person"/>
    </session-factory>
</hibernate-configuration>
```

Hibernate has many configuration properties. Apart from the standard connection properties, it is worth mentioning the dialect property which allows us to specify the name of the SQL dialect for the database.

Finally, hibernate also needs to know the fully-qualified name of the entity class via the mapping tag. Once we complete the configuration, we'll use the *SessionFactory* class, which is the class responsible for creating and pooling JDBC connections.

Step 4:

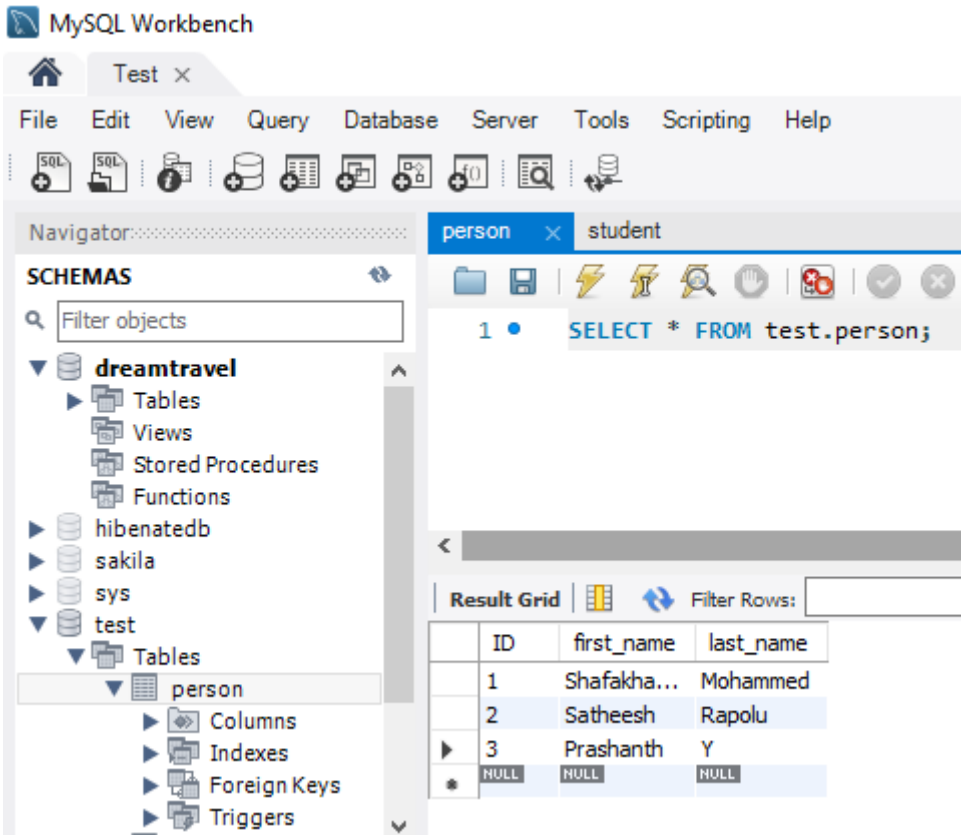
App class body:

```
SessionFactory sessionFactory;
// configures settings from hibernate.cfg.xml
StandardServiceRegistry registry = new
StandardServiceRegistryBuilder().configure().build();
try {
    sessionFactory = new
MetadataSources(registry).buildMetadata().buildSessionFactory();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

Now that we have our connection set up, we can run a query to select all the people from our person table:

```
Session session = sessionFactory.openSession();
session.beginTransaction();

List<Person> result = session.createQuery("from Person", Person.class).list();

result.forEach(person -> {
    //do something with Person instance...
});
session.getTransaction().commit();
session.close();
```

Output:


MySQL Workbench
Test
File Edit View Query Database Server Tools Scripting Help
Navigator:
SCHEMAS
Filter objects
dreamtravel
Tables
Views
Stored Procedures
Functions
hibenatedb
sakila
sys
test
Tables
person
Columns
Indexes
Foreign Keys
Triggers
person
student
1 SELECT * FROM test.person;
Result Grid
Filter Rows:
ID first_name last_name
1 Shafakha... Mohammed
2 Satheesh Rapolu
3 Prashanth Y
NULL NULL NULL

Experiment: 10

CREATE & INSERT DATA IN THE TABLE (SEVER SIDE PROGRAMMING)

Write a java server program to add or insert data into created table.

A common requirement when you are developing and testing your application is to use create and drop your database schema at start up. Let's learn how to do that in a simple Spring Boot application.

Configuration steps for automatic Table generation

Here is a sample *application.properties* configuration that will let Hibernate create the Database tables out of your Entity beans:

```
spring.datasource.url=jdbc:mysql://localhost:3306/springdemo_db?useSSL=false

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

spring.jpa.generate-ddl=true

spring.jpa.hibernate.ddl-auto = create
```

Here are the possible values for hibernate.hbm2ddl.auto :

- validate: validate the schema, makes no changes to the database.
- update: update the schema.
- create: creates the schema, destroying previous data
- create-drop: drop the schema at the end of the session

Automatic schema creation does not work?

If you are unable to see the database schema generation, check the following points:

Are you including the configuration file in the right place? In a Maven project, you should place your application.properties (or equivalent) in src/main/resources folder.

Your Entity classes should be in the same or in a sub-package relative to where you have you class with @EnableAutoConfiguration. If that's not trye, your Spring Boot application simply cannot detect them and hence will not create your Database tables.

As workaround, you can use @EntityScan annotation in the main class, specifying the Entity you want to save or package too. See the following example:

```
@SpringBootApplication
@EnableConfigurationProperties
@EntityScan(basePackages = {"com.mlritm.model"})  // force scan JPA entities
public class Application {
  private static ConfigurableApplicationContext applicationContext;
  public static void main(String[] args) {
    Application.applicationContext = SpringApplication.run(Application.class, args);
  }
}
```

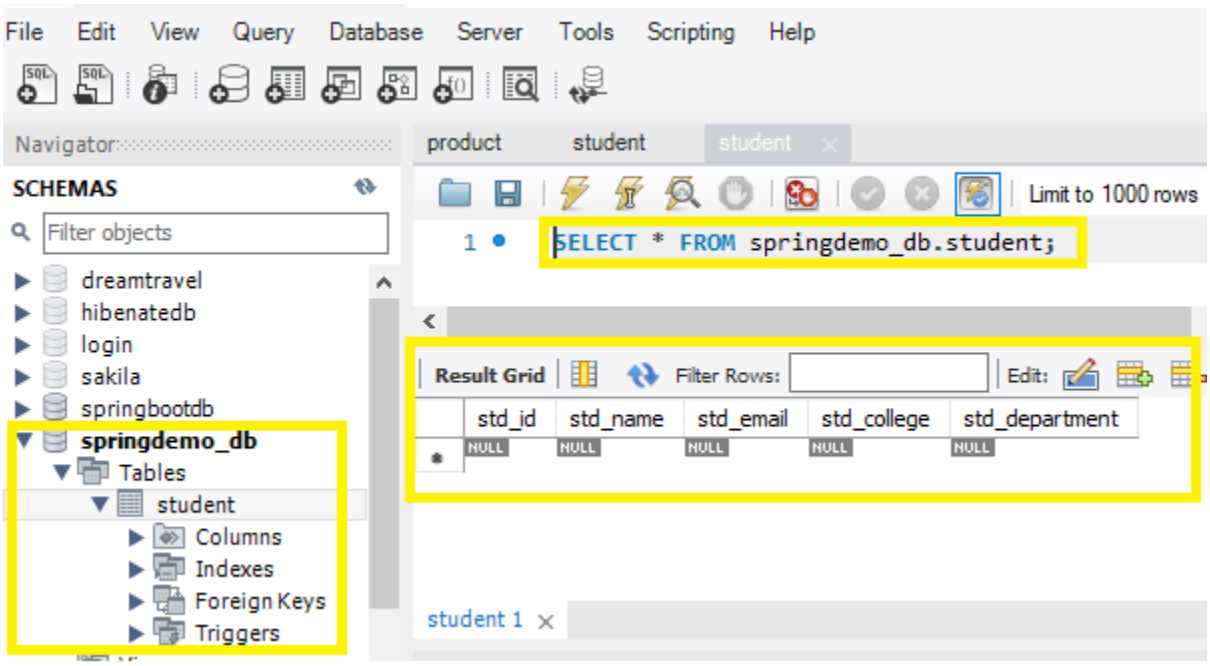
Delegate schema creation to the Database

For some databases, it is possible to delegate the creation of the schema you are using to the Database itself, via the JDBC Driver.

For example, if you are using MySQL Database, you could use the createDatabaseIfNotExist property as follows:

```
spring.datasource.url=jdbc:mysql://localhost:3306/sampledb?createDatabaseIfNotExist=true&amp;amp;useUnicode=true&amp;amp;characterEncoding=utf-8&amp;amp;autoReconnect=true
```

Output:



Experiment: 11

DELETE DATA FROM A TABLE (SERVER SIDE PROGRAMMING)

Write a java server program to delete the data from the Customer table.

To delete data in Spring Boot with JPA and Hibernate

Use built-in Delete APIs of Spring Data JPA repositories

Spring Data Repository provides delete(entity) and deleteById(entityId) APIs to delete data with JPA and Hibernate entity

Suppose we define Customer JPA entity and Spring Data Repository as below

```
@Entity
public class Customer {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;

   ...
}

public interface CustomerRepository extends JpaRepository<Customer, Integer> {
}
```

Then we can directly call the built-in Delete APIs of CustomerRepository to delete data

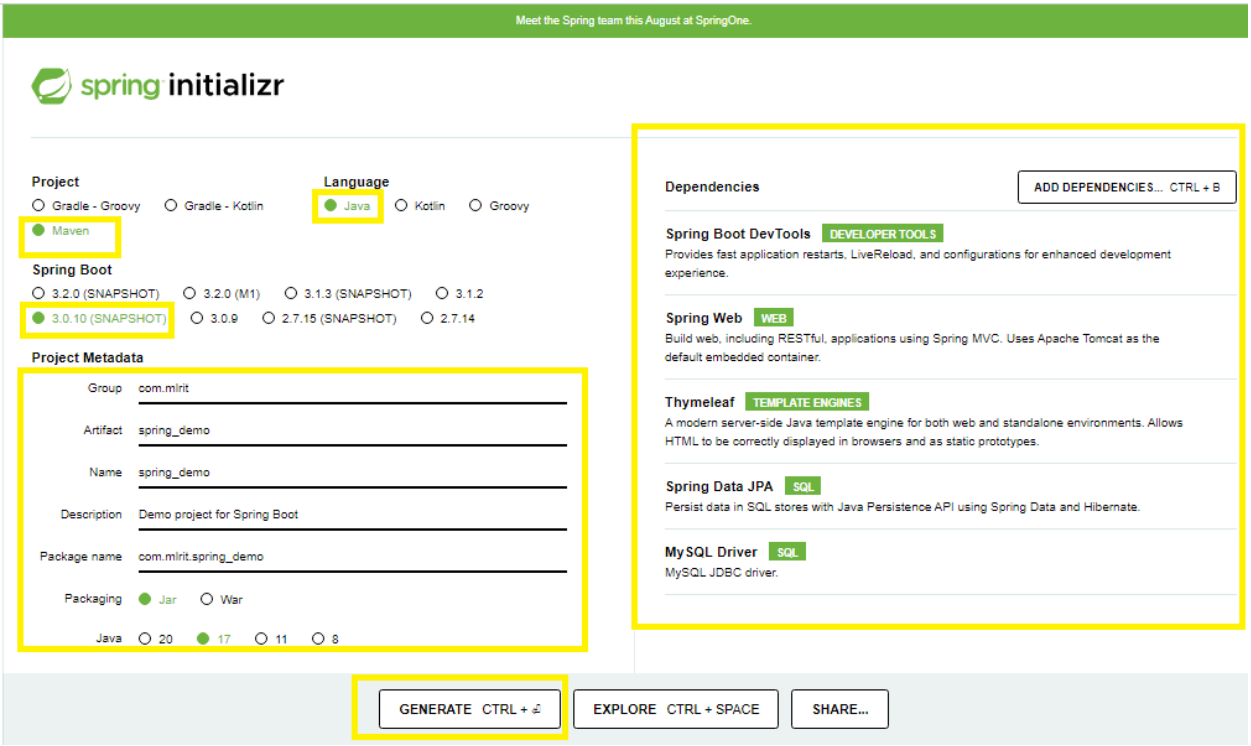
```
@Service
@Transactional
public class CustomerService {
   @Autowired
   private CustomerRepository customerRepository;

   public void delete(Customer customer) {
      customerRepository.delete(customer);
   }

   public void deleteById(Integer id) {
      customerRepository.deleteById(id);
   }

   ...

}
```

Experiment: 12

CRUD OPERATIONS USING SPRING BOOT

Write a java program to perform CRUD operations using spring or spring boot. (Update, Delete)

Create a spring project go to the link - https://start.spring.io/, do all the things highlighted and generate, then a "zip" file will be download as soon as you hit the generate button.



Once the download is complete, Extract the zip file Import the project folder into the Eclipse IDE.

Implementation of CRUD operations in Springboot

When you import the downloaded spring boot application by default *SpringDemoApplication.java* is created which consists of the basic template for the application.

SpringDemoApplication.java

```
package com.mlrit.spring_demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringDemoApplication {
	public static void main(String[] args) {
		SpringApplication.run(SpringDemoApplication.class, args);
	}
}
```

After successfully importing the project files, Open the MySQL Workbench and open the connection and create a database *"springbootdb"* inside that create a table product with attributes *id, version, name, price, and product_id*.
Then, in the Eclipse IDE you have to Create the package *com.mlritm. service*. Inside the package, you have to create an interface *ProductService* and a class *ProductServiceImpl*.

ProductSercive.java

```
package com.mlritm.service;
import com.mlritm.entity.Product;
public interface ProductService {
   Iterable<Product> listAllProducts();
   Product getProductById(Integer id);
   Product saveProduct(Product product);
   void deleteProduct(Integer id);
}
```

ProductServiceImpl.java

```
package com.mlritm.service;

import com.mlritm.entity.Product;
import com.mlritm.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
/**
 * Product service implement.
 */
@Service
public class ProductServiceImpl implements ProductService {
    @Autowired
    private ProductRepository productRepository;
   @Override
   public Iterable<Product> listAllProducts() {
      return productRepository.findAll();
   }
   @Override
   public Product getProductById(Integer id) {
      return productRepository.findById(id).get();
   }
   @Override
   public Product saveProduct(Product product) {
      return productRepository.save(product);
   }
   @Override
   public void deleteProduct(Integer id) {
      productRepository.deleteById(id);
   }
}
```

After that you have to Create the package *com.mlritm.controller*. Inside the package, you have to create the classes *IndexController* and *ProductController*. This class deals with RESTful APIs for CRUD operations.

IndexController.java

```
package com.mlritm.controller;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;

@RestController
public class IndexController {
   @GetMapping("/")
   String index() {
      return "index";
   }
}
```

ProductController.java

```
package com.mlritm.controller;
import com.mlritm.entity.Product;
import com.mlritm.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PutMapping;

/**
 * Product controller.
 */
@RestController("/products")
public class ProductController {
    @Autowired
    private ProductService productService;

  @GetMapping("/")
   public String list(Model model) {
      model.addAttribute("products", productService.listAllProducts());
      System.out.println("Returning products:");
      return "products";
   }

   @GetMapping("/{id}")
   public String showProduct(@PathVariable Integer id, Model model) {
      model.addAttribute("product", productService.getProductById(id));
      return "productshow";
   }
```

```
    // code to modify the product
    @PutMapping("/edit/{id}")
    public String edit(@PathVariable Integer id, Model model) {
        model.addAttribute("product", productService.getProductById(id));
        return "productform";
    }

    @RequestMapping("product/new")
    public String newProduct(Model model) {
        model.addAttribute("product", new Product());
        return "productform";
    }

    @RequestMapping(value = "product", method = RequestMethod.POST)
    public String saveProduct(Product product) {
        productService.saveProduct(product);
        return "redirect:/product/" + product.getId();
    }

    @DeleteMapping("/{id}")
    public String delete(@PathVariable Integer id) {
        productService.deleteProduct(id);
        return "redirect:/products";
    }

}
```

Then, you must create the package *com.mlritm.repository* inside the package you have to create an EmployeeRepository.java file. *ProductRepository* interface extends the JPARepository interface which is used to perform the CRUD operations in Springboot.

ProductRepository.java

```
package com.mlritm.repository;
import com.mlritm.entity.Product;
import org.springframework.data.jpa.repository.*;

public interface ProductRepository extends JpaRepository<Product, Integer> {

}
```

Then, you must Create the package *com.mlritm.entity* inside the package you have to create the class Product. This class retrieves the information from the database and displays it to the user.

Product.java

```
package com.mlritm.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
import javax.persistence.Version;

import java.math.BigDecimal;

/**
 * Product entity.
 */
@Entity
public class Product {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Integer id;

   @Version
   private Integer version;

   private String productId;
   private String name;
   private BigDecimal price;

   public Product() {

   }

        public Integer getId() {
                return id;
        }

        public void setId(Integer id) {
                this.id = id;
        }

        public Integer getVersion() {
                return version;
        }

        public void setVersion(Integer version) {
                this.version = version;
        }

        public String getProductId() {
                return productId;
        }

        public void setProductId(String productId) {
                this.productId = productId;
        }

        public String getName() {
                return name;
        }
```

```
        public void setName(String name) {
                this.name = name;
        }

        public BigDecimal getPrice() {
                return price;
        }

        public void setPrice(BigDecimal price) {
                this.price = price;
        }

  }
```

After that, you have to create the *index.html* page inside the folder *src/main/resources*. This file is created to create a User Interface to perform CRUD operations in Springboot and MYSQL.

Index.html:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootswatch/4.5.2/cosmo/bootstrap.min.css" />
        <script src= "https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" ></script>
</head>
<body>
<div>
        <h2 >Spring Boot Crud System - Product</h2>
        <table class="table">
        <tr>
                <td>
                <div align = "left" >
                   <h3><a  th:href="@{'/new'}">Add new</a></h3>
          </div>
                </td>
        </tr>
        <tr>
                <div class="col-sm-5" align = "center">
    <div class="panel-body" align = "center" >
```

```
                <thead class="thead-dark">
                <tr>
                <th>ID</th>
        <th>Version</th>
        <th>Name</th>
        <th>Price</th>
                        <th>Product ID</th>
        <th>Edit</th>
         <th>Delete</th>
                </tr>
                </thead>
                <tbody>
                        <tr  th:each="product : ${listproduct}">
                                <td th:text="${product.id}">ID</td>
                                <td th:text="${product.version}">Version</td>
                                <td th:text="${product.name}">Name</td>
                                <td th:text="${product.price}">Price</td>
                                <td th:text="${product.product_id}">Product ID</td>
                                <td>
                                        <a th:href="@{'/edit/' + ${product.id}}">Edit</a>
                                </td>
                                <td>
                                        <a th:href="@{'/delete/' + ${product.id}}">Delete</a>
                                </td>
                                </tr>
                </tbody>


          </div>
        </div>
          </tr>
          </tbody>
          </table>
          <div>
</body>
</html>
```

Inside the *src/main/resources* folder, create another HTML file new.html. This new file is created to open a new page to create a new user.

new.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="utf-8" />
   <title>Create New Product</title>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootswatch/4.5.2/cosmo/bootstrap.min.css" />
   <script src= "https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" ></script>
</head>
<body>
<div align="center">
   <h1>Create New Employee</h1>
   <br />
    <div class="col-sm-4">
   <form action="#" th:action="@{/save}" th:object="${employee}" method="post">
                    <div alight="left">
        <tr>
           <label class="form-label" >Employee Name</label>
           <td>
                    <input type="hidden" th:field="*{id}" />
                    <input type="text" th:field="*{ename}" class="form-control" placeholder="Emp Name" />
           </td>
        </tr>
      </div>
       <div alight="left">
        <tr>
           <label class="form-label" >mobile</label>

           <td>
            <input type="text" th:field="*{mobile}" class="form-control" placeholder="mobile" />
           </td>
        </tr>
      </div>
      <div alight="left">
           <tr>
           <label class="form-label" >salary</label>
           <td><input type="text" th:field="*{salary}" class="form-control" placeholder="salary" /></td>
        </tr>
        </div><br>
        <tr>
        <td colspan="2"><button type="submit" class="btn btn-info">Save</button> </td>
        </tr>
   </form>
</div>
</body>
</html>
```

After that, we have to set the database path and project configuration in the *application.properties* file located in the *src/main/resources* folder.

spring.datasource.url=jdbc:mysql://localhost:3306/springbootdb

server.error.whitelabel.enabled=false

server.port=7070

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.open-in-view=false

spring.thymeleaf.cache=false

api.base.path = http://localhost:7070/

To run our application, go to the browser and check the site *localhost:7070*. Our final application has the features to read the product details, create new product data, update the product details and delete the product data.

Output: