



**UNIVERSITI MALAYSIA TERENGGANU**

---

**CSM3023 WEB BASED APPLICATION DEVELOPMENT**

**BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONORS**

**LAB 02**

**SEMESTER I 2023/2024**

---

**Prepared for:**

**DR MOHAMAD NOR BIN HASSAN**

**Prepared by:**

**NUR ATIRAH BINTI SHAMSUDDIN (S67397)**

## Table of Contents

Task 1: Data Sharing in Servlet.....	5
Task 2: Creating A Table in MySQL Database.....	15
Task 3: Setting the Environment of Web Application for Database Connection .....	20
Task 4: Using Servlets for Database CRUD Operations.....	23

**Arahan:**

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Pusat Pengajian Informatik dan Matematik Gunaan (PPIMG), Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual. Tandakan (/) setiap langkah yang telah selesai dibuat dan tulis kesimpulan bagi setiap aktiviti yang telah selesai dijalankan.

**Instruction:**

*This laboratory manual is for use by the students of the School of Informatics and Applied Mathematics (PPIMG), Universiti Malaysia Terengganu (UMT)only. It is not permissible to print and distribute this manual without the official authorisation of the author.*

*Please follow step by step as described in the manual. Tick (/) each step completed and write the conclusions for each completed activity.*

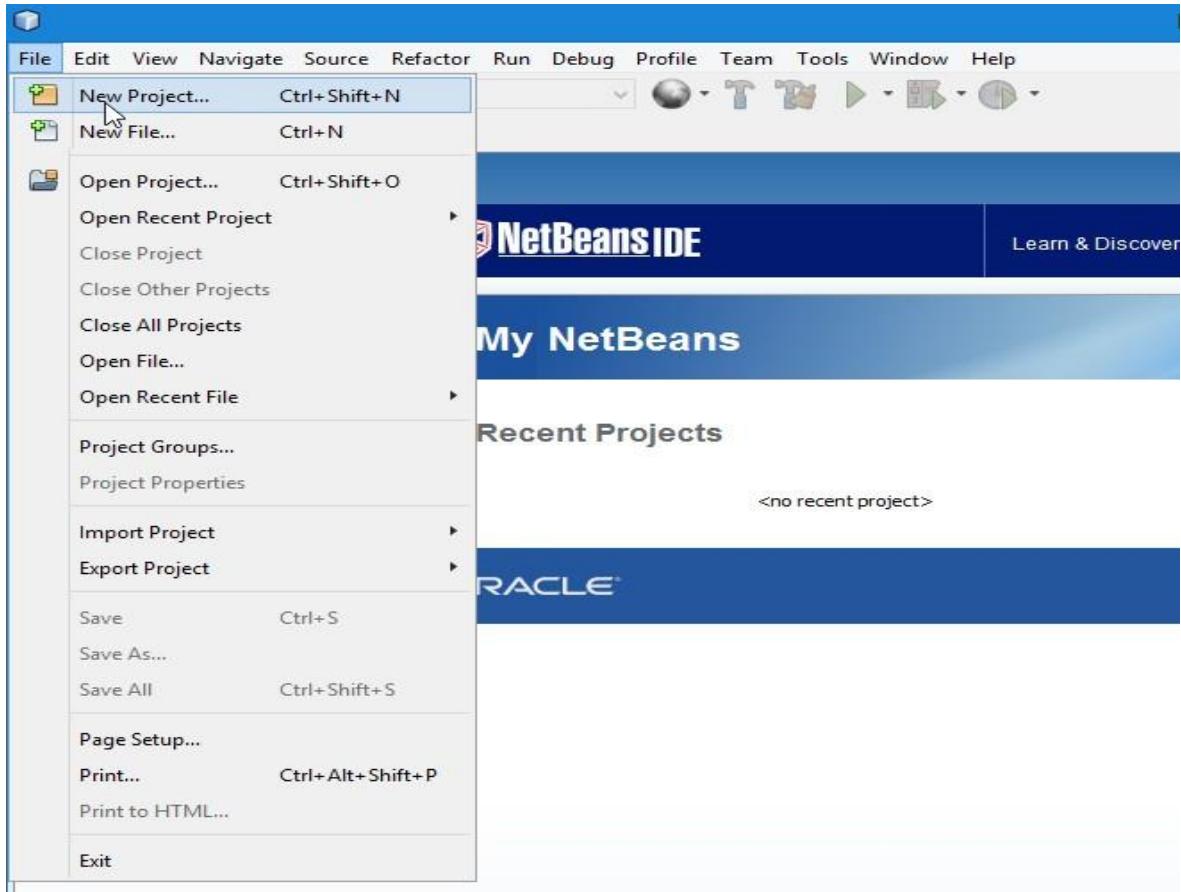
## Task 1: Data Sharing in Servlet

**Objective:** To use servlet for request forwarding and data sharing

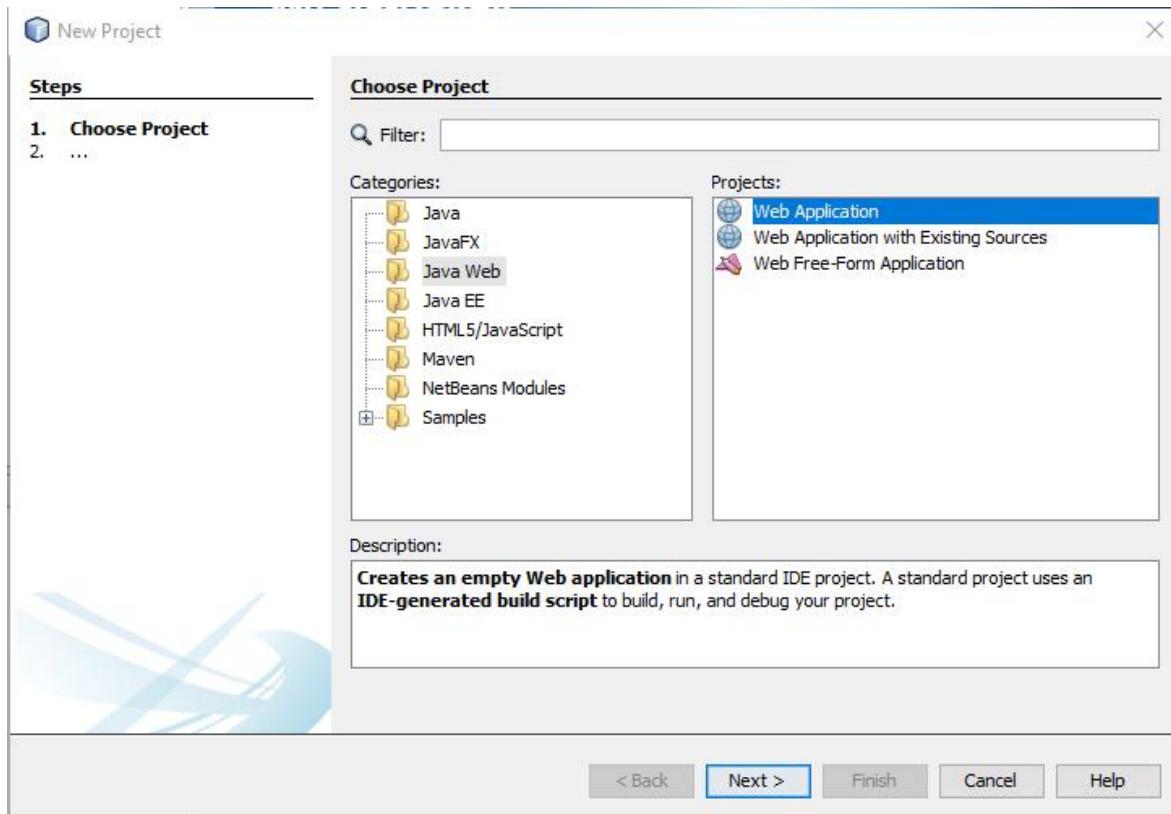
**Problem Description:** Write a login form and a servlet to authenticate a user.

**Estimated time:** 30 minutes

1. Create a directory F:\CSF3107-SXXXXX (only if you have not created it yet). Note: Replace XXXXX with your matric number.
2. Go to F:\CSF3107-SXXXXX\ directory and create sub-directory and name it as *Lab 2*.
3. Open your NetBeans IDE.
4. Go to File -> New Project

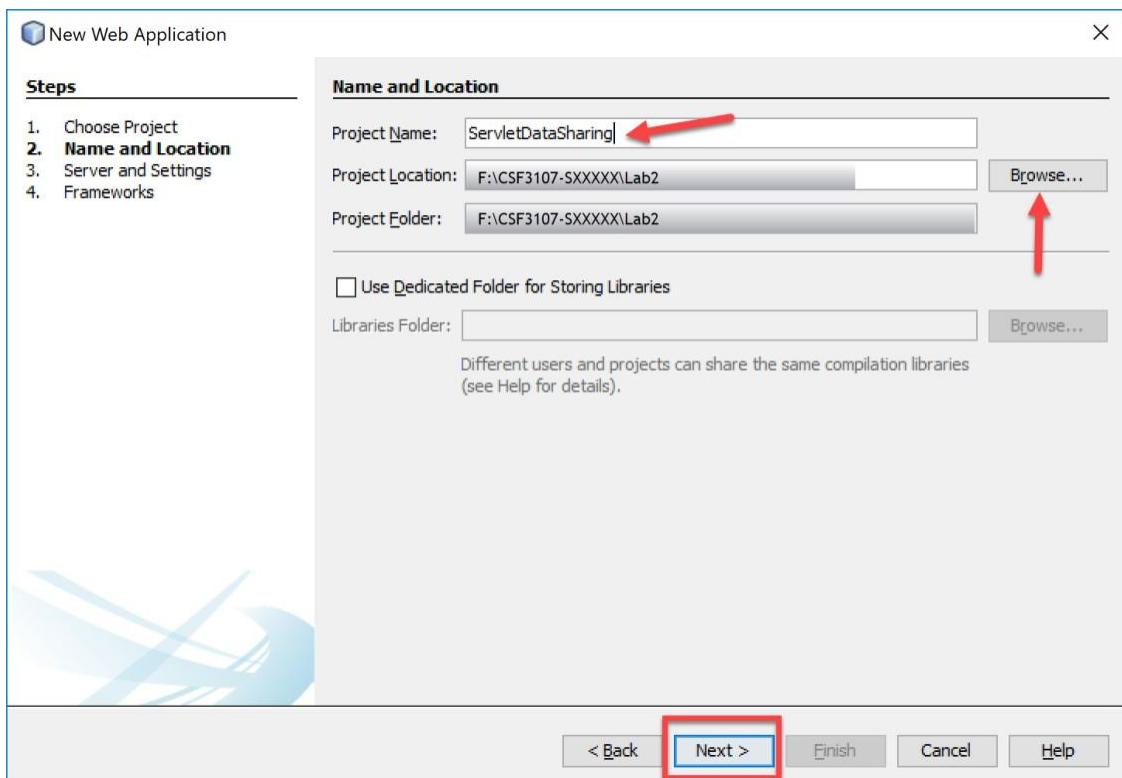


5. Select Java Web -> Web Application and click Next.

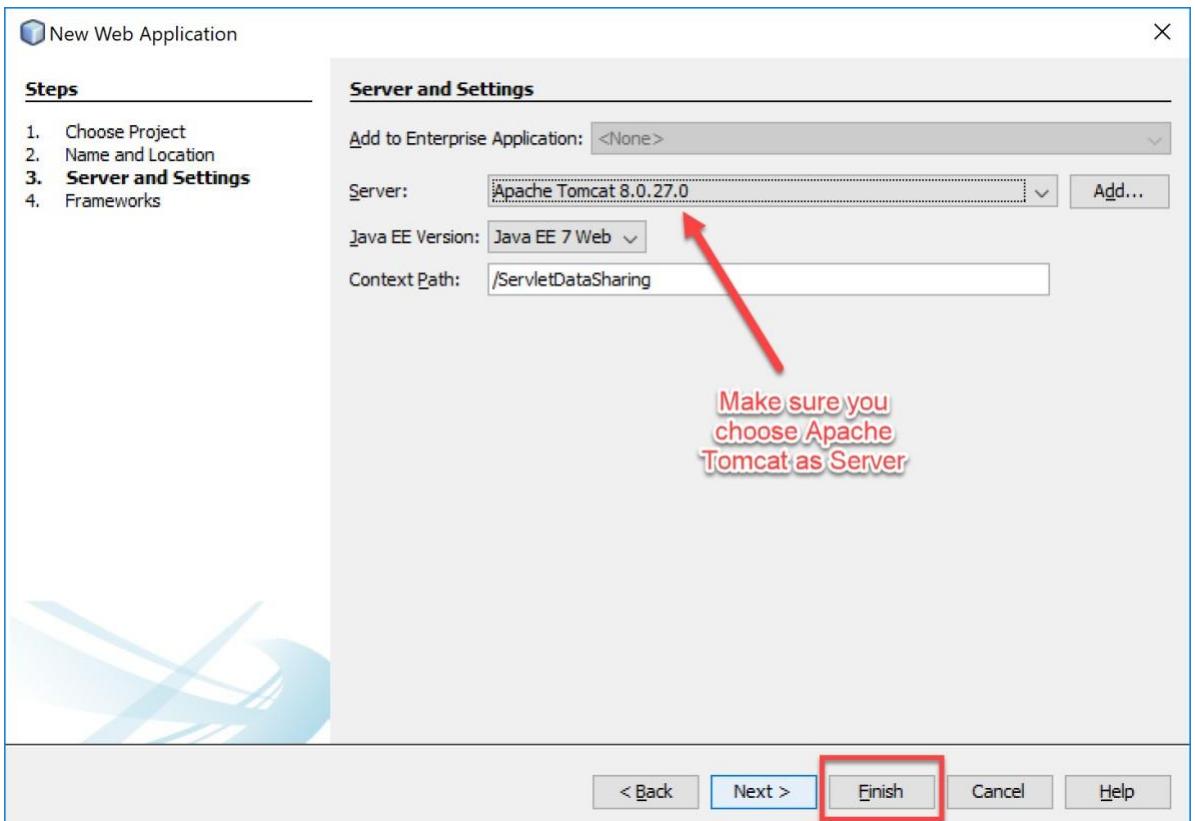


6. Type Project Name: *ServletDataSharing*.

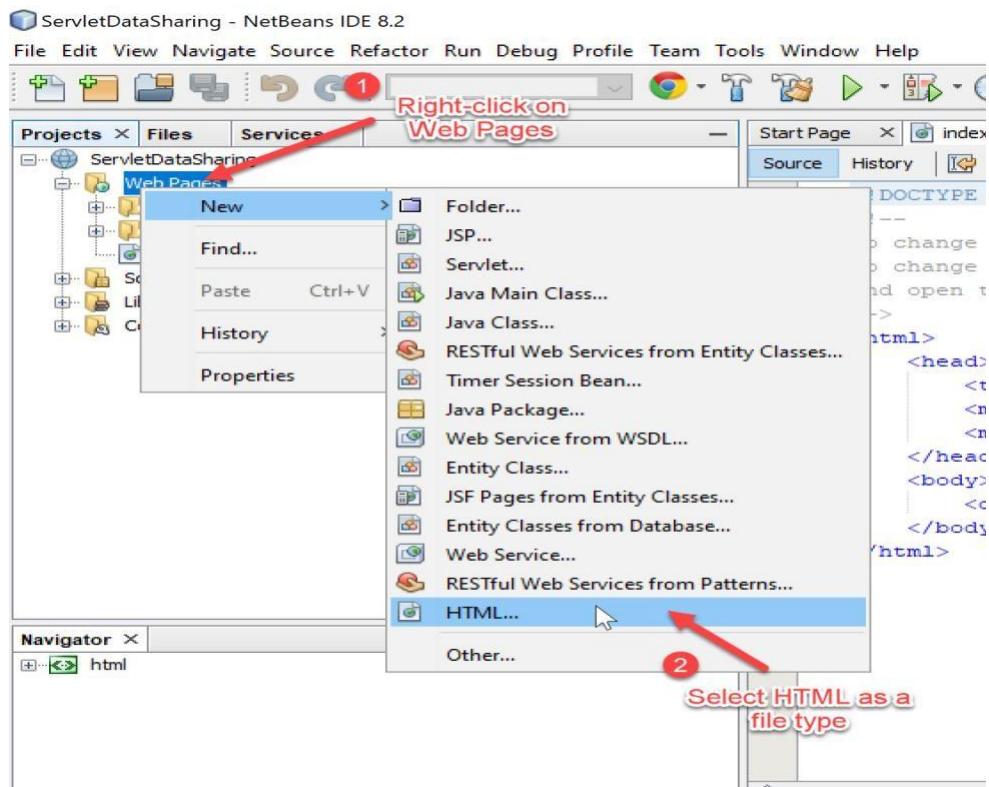
7. Click Browse and choose Project Location: F:\CSF3107-SXXXXX\Lab2. Then click the Next button.



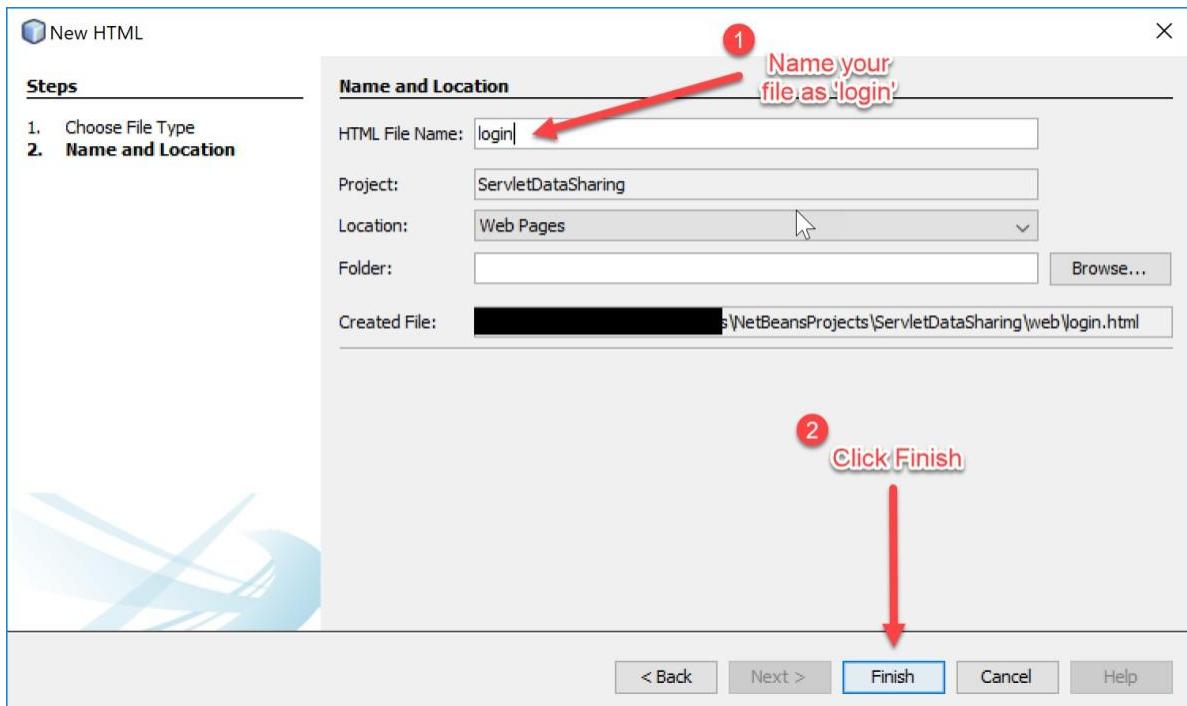
8. Then, choose Apache Tomcat as the server and click Finish.



9. You will see your project structure on the left side of your Netbeans editor. Next, you will create an HTML file for a login page. Follow the instructions as shown on the following screenshot.



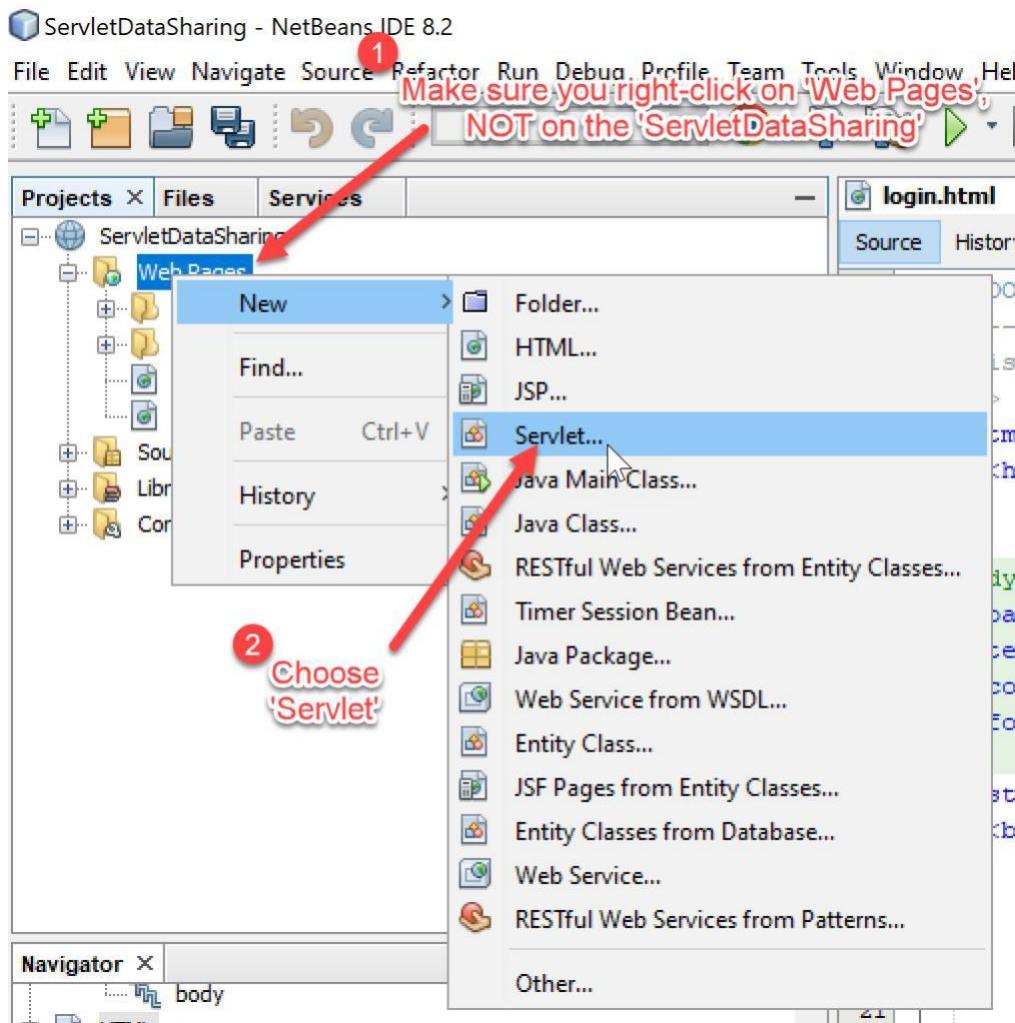
10. Name your HTML file as ‘login’, then click Finish.

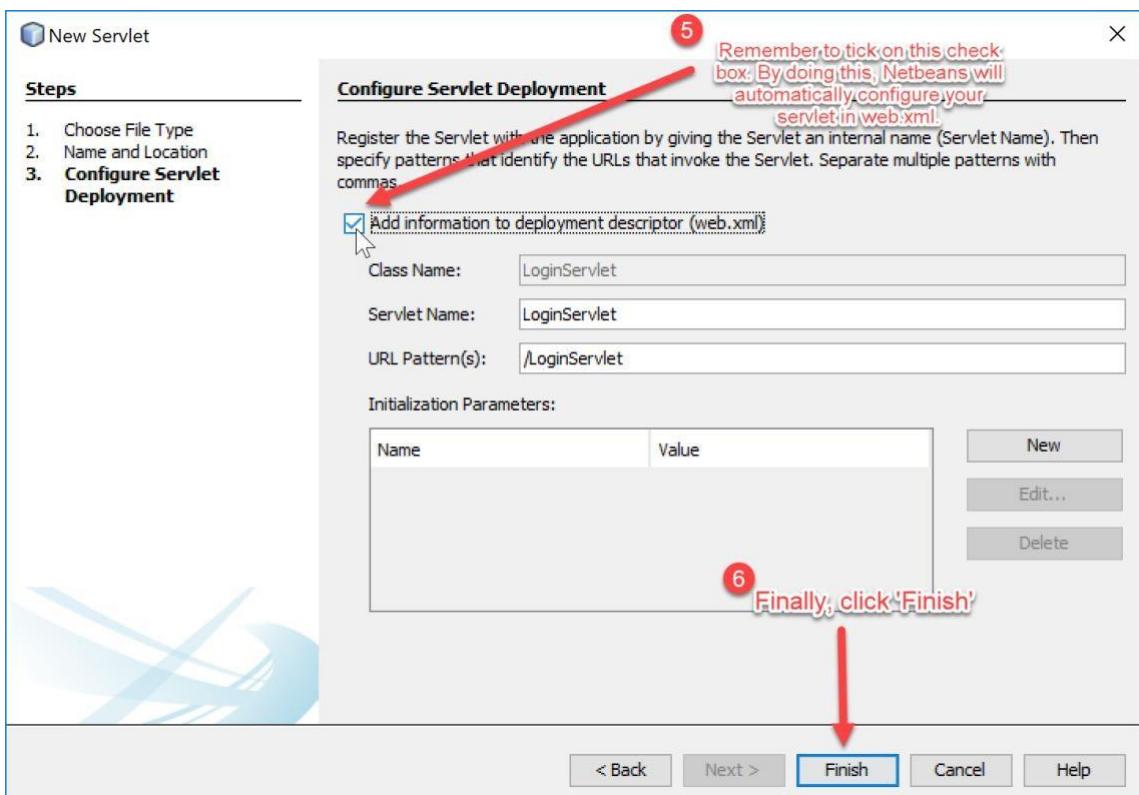
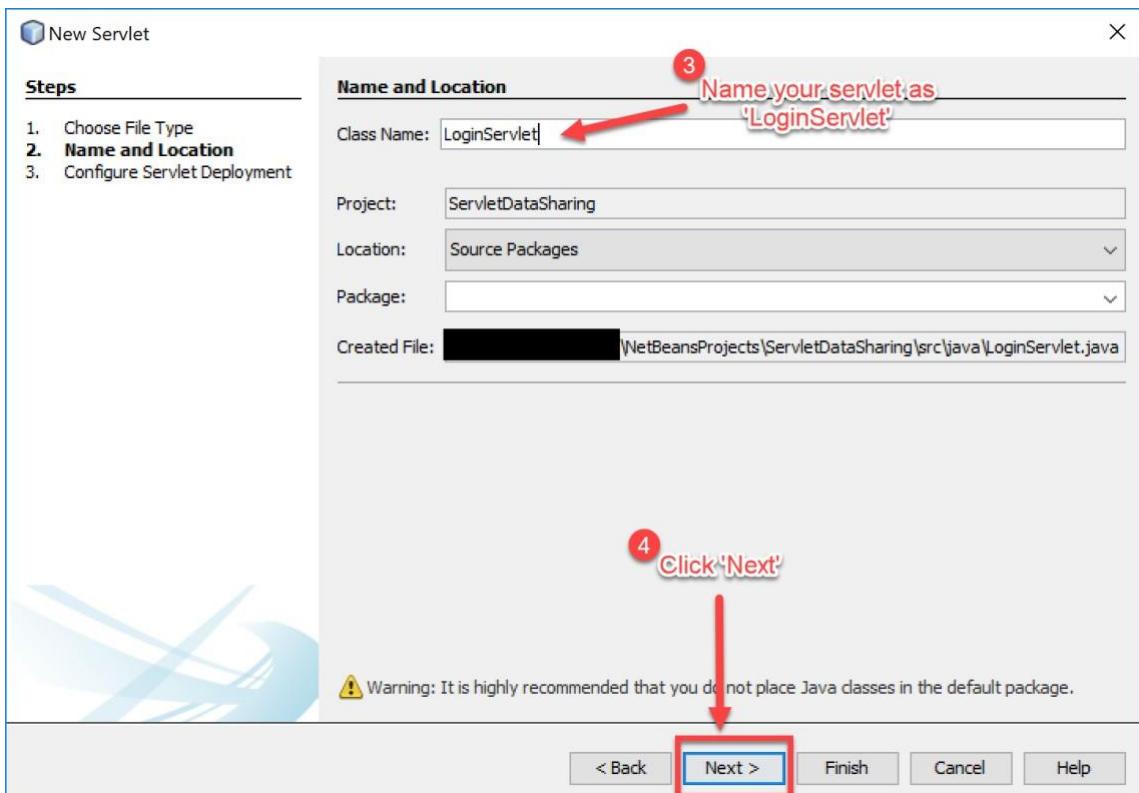


11. Type the following HTML markups into your login.html.

```
<!DOCTYPE html>
<!-- This is a login page.
--&gt;
&lt;html&gt;
    &lt;head&gt;
        &lt;title&gt;Login Page&lt;/title&gt;
        &lt;style&gt;
body {
    background-color: black;
    text-align: left;
    color: white;
    font-family: Arial, Helvetica, sans-serif;
}
&lt;/style&gt; &lt;/head&gt;
&lt;body&gt;
    &lt;h1&gt;Welcome to CSF3107&lt;/h1&gt;
    &lt;p&gt;Please insert your username and password&lt;/p&gt;
    &lt;form name="login" id="login" action="LoginServlet" method="POST" autocomplete="off"&gt;
        Username:&lt;input name="txtUsername" type="text"&gt; &lt;br&gt;
        Password:&lt;input name="txtPassword" type="text"&gt;&lt;br&gt;
        &lt;br&gt;
        &lt;input name="btnLogin" value="Login" type="button"&gt;
        &lt;input name="txtReset" value="Reset" type="reset"&gt;&lt;br&gt;
    &lt;/form&gt;
    &lt;p&gt;&lt;br&gt;
    &lt;/p&gt;
    &lt;/body&gt;
&lt;/html&gt;</pre>
```

12. Next, we are going to create a servlet to process the username and password insert by the user on the login form. We start it by creating a new file for our servlet. Follow the steps below:





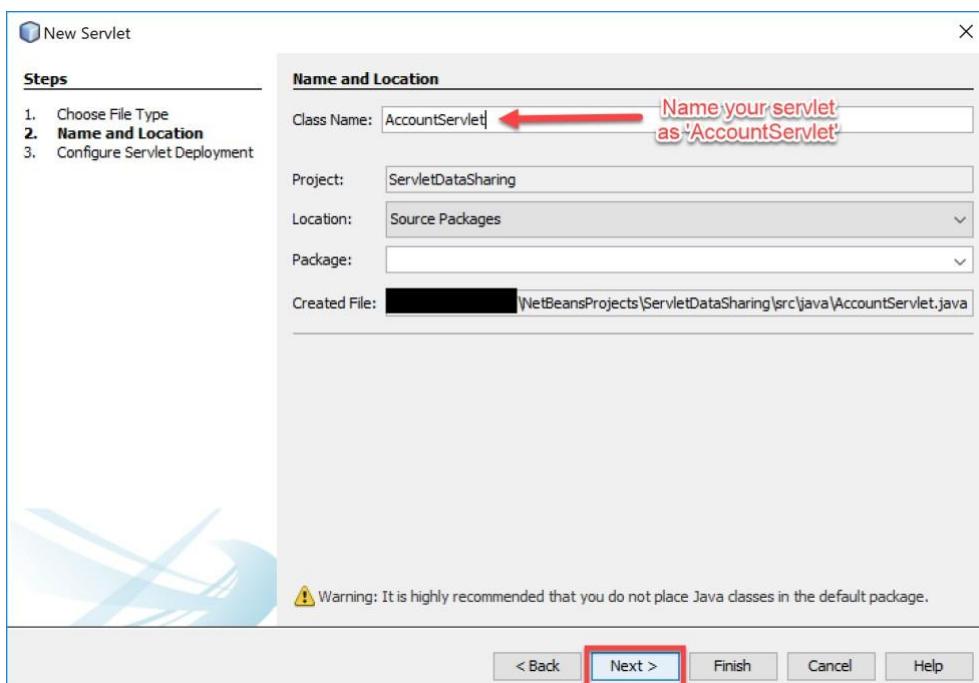
```

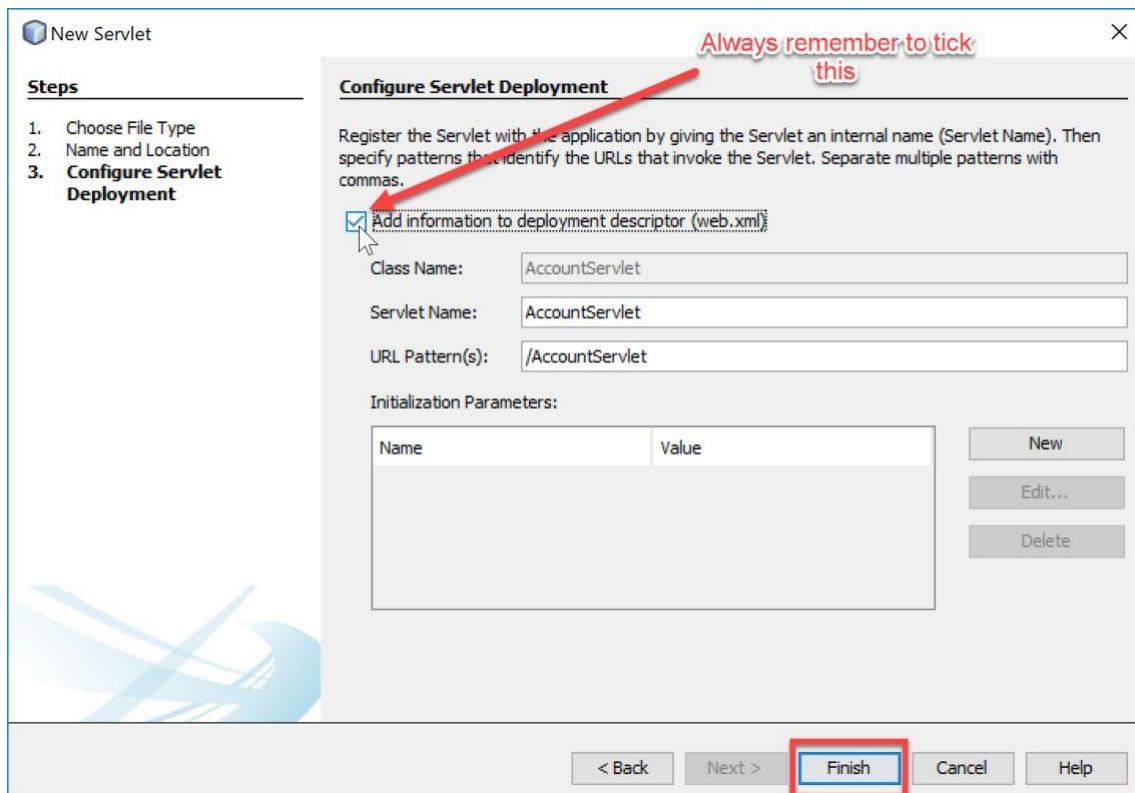
7 import java.io.IOException;
8 import java.util.HashMap; ← We will use HashMap to save the
9 import javax.servlet.ServletException; username and password, remember to
10 import javax.servlet.http.HttpServlet; import it from the library.
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import javax.servlet.RequestDispatcher;
14 import javax.servlet.ServletContext;

15 /**
16  * ... 4 lines ...
17  */
18 public class LoginServlet extends HttpServlet {
19
20     HashMap <String, String> users = new HashMap();
21
22     @Override
23     public void init() throws ServletException{
24         super.init();
25         users.put("Ali", "1234");
26         users.put("Ahmad", "4567");
27         users.put("Muthu", "8910");
28     }
29
30     /**
31      * Processes requests for both HTTP <code>GET</code> and <code>POST</code> ... 9 lines ...
32     */
33     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
34         throws ServletException, IOException {
35         response.setContentType("text/html;charset=UTF-8");
36
37         String username = request.getParameter("txtUsername");
38         String password = request.getParameter("txtPassword");
39
40         if (!username.equals("") && !password.equals("") && users.get(username).equals(password)) {
41             request.setAttribute("userid", username);
42             ServletContext sc = getServletContext();
43             RequestDispatcher rd = sc.getRequestDispatcher("/AccountServlet");
44             rd.forward(request, response);
45         } else {
46             //avoid direct access to the servlet
47             RequestDispatcher rd = request.getRequestDispatcher("/login.html");
48             rd.forward(request, response);
49         }
50     }
51
52 }
53
54
55
56
57
58
59

```

13. Next, we need a servlet to return the information about users' accounts and display it on the browser. Repeat previous step for creating a new servlet and fill the details as shown on the screenshots below.





```

1  ...
2
3
4
5
6
7 import java.io.IOException;
8 import java.io.PrintWriter;
9 import java.util.HashMap;
10 import javax.servlet.ServletException;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14

```

Once again, we need to import HashMap class. This time we use this to store the information about the user accounts.

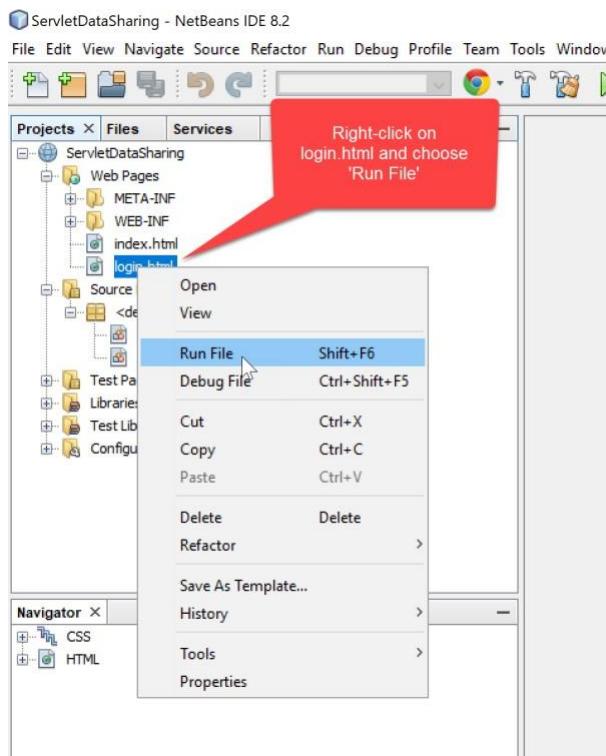
AccountServlet.java - Editor

AccountServlet.java

Source History

```
15  /*...4 lines */
16  public class AccountServlet extends HttpServlet {
17      HashMap<String, String[]> account = new HashMap();
18      @Override
19      public void init() throws ServletException{
20          super.init();
21          account.put("Ali", new String[]{"31/01/2019: 2000.00", "28/02/2019: 3000.00"});
22          account.put("Ahmad", new String[]{"31/01/2019: 100.00", "28/02/2019 5000.00"});
23          account.put("Muthu", new String[]{"31/01/2019: 1000", "28/02/2019 2000"});
24      }
25
26
27  }
28
29  /**
30  * Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
31  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
32      throws ServletException, IOException {
33      response.setContentType("text/html;charset=UTF-8");
34
35      String userid_login = (String)request.getAttribute("userid");
36
37      try (PrintWriter out = response.getWriter()) {
38
39          out.println("<!DOCTYPE html>");
40          out.println("<html>");
41          out.println("<head>");
42          out.println("<title>Servlet AccountServlet</title>");
43          out.println("</head>");
44          out.println("<body>");
45
46          if(account.get(userid_login)==null){
47              out.println("<h1>Sorry, no information found!</h1>");
48          }
49          else{
50              out.println("<h1>Account status for: " + userid_login + "</h1>");
51              for(String tempAcc: account.get(userid_login)){
52                  out.println("<h2>" +tempAcc+"</h2>");
53              }
54          }
55
56          out.println("</body>");
57          out.println("</html>");
58      }
59  }
60
61
62
63
64
65 }
```

Type the rest of the codes into AccountServlet.java



14. The Output will appear in a web browser. Put a Username as ‘Ahmad’ and Password as ‘4567’, then click *Login*.

The screenshot shows a web browser window with the URL `localhost:8084/ServletDataSharing/login.html`. The page has a black background with white text. At the top, it says **Welcome to CSF3107**. Below that, it says **Please insert your username and password**. There are two input fields: one for **Username** containing **Ahmad**, and one for **Password** containing **....**. At the bottom are two buttons: **Login** and **Reset**.

15. If you have followed all the previous steps correctly, you will see the account information of a user named ‘Ahmad’. Repeat step 14 with other users as input and see the result.



#### Reflections:

1. What have you learnt from this exercise?

#### Answer:

- i. How to link the `login.html` to `LoginServlet` and `AccountServlet`.
- ii. Servlet are kind of like a library that store information that will be called in html file.

2. What are the common methods used in Java Servlet?

Answer: `processRequest()` method.

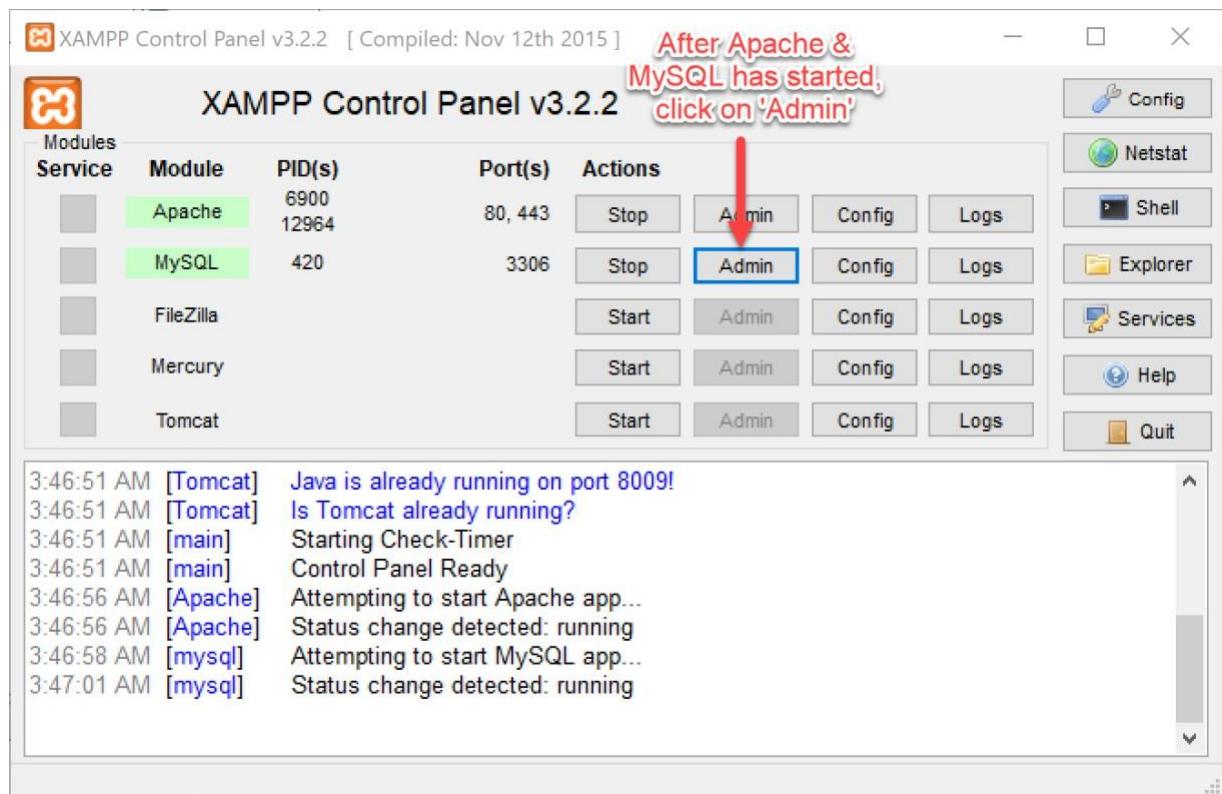
## Task 2: Creating A Table in MySQL Database

**Objective:** To create a MySQL table to store user credentials

**Problem Description:** Prepare a user table to be used in Web Application

**Estimated time:** 30 minutes

### 1. Start Apache and MySQL.



## 2. Create a new database.

The screenshot shows the 'Databases' section of the phpMyAdmin interface. A red arrow points to the 'New' button in the sidebar, labeled 'Click on 'New' to create a new Database'. Another red arrow points to the 'Create database' input field, labeled 'Name your database as 'CSF3107''. A third red arrow points to the 'Create' button, labeled 'Click 'Create''. The database list table shows the newly created database 'CSF3107' along with other standard MySQL databases.

## 3. After the database has been created, now we are going to create a table named 'users'. This table has four columns.

The screenshot shows the 'Structure' screen for the 'csf3107' database. A red arrow points to the 'csf3107' database link in the sidebar, labeled 'Click on your database link, CSF3107'. Another red arrow points to the 'Name:' input field in the 'Create table' dialog, labeled 'Create a table, name it as 'users''. A third red arrow points to the 'Go' button in the 'Create table' dialog, labeled 'Click 'Go''. The message 'No tables found in database.' is displayed above the table creation form.

4. Setup the table ‘users’ as below:

Table name: users

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
<input type="text" value="id"/>	<input type="button" value="INT"/>	<input type="text" value=""/>	<input type="button" value="None"/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PRIMARY
<input type="text" value="username"/>	<input type="button" value="VARCHAR"/>	<input type="text" value="100"/>	<input type="button" value="None"/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="text" value="password"/>	<input type="button" value="VARCHAR"/>	<input type="text" value="225"/>	<input type="button" value="None"/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="text" value="roles"/>	<input type="button" value="VARCHAR"/>	<input type="text" value="10"/>	<input type="button" value="None"/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Storage Engine: InnoDB

PARTITION definition: Partition by: ( Expression or column list )  
Partitions:

Fill in the table's schema as given.

Click on A\_I to enable autoincrement for field 'id'.

Click Save.

5. After you tick for auto-increment, this dialogue box will appear, leave the settings as it is and click ‘Go’.

Add index

Index name:

Index choice:

+ Advanced Options

Column	Size
<input type="text" value="id [int]"/>	<input type="text" value=""/>

Click 'Go' → Go Cancel

6. Your newly created table can be seen on the left phpMyAdmin.

The screenshot shows the phpMyAdmin interface. On the left, the database tree is visible with the 'users' table selected under the 'New' database. A red arrow labeled '1 Click on 'users' table' points to the 'users' entry. In the main panel, the 'Structure' tab is active, displaying the table definition. A red box highlights the table header row ('id', 'username', 'password', 'roles'). A red arrow labeled '2 You will see an empty table' points to this row. Below the table, there is a 'Query results operations' section with a 'Create view' button.

7. You may insert some data into the table by following the steps on the screenshot.

The screenshot shows the phpMyAdmin interface with the 'Insert' tab selected for the 'users' table. A red arrow labeled '1 Click on 'Insert' link to insert a new data' points to the 'Insert' button in the top navigation bar. In the main panel, two data entry forms are shown side-by-side. The first form has an 'Ignore' checkbox checked, which is highlighted with a red arrow labeled '2 Untick the 'Ignore' checkbox'. The second form has the 'Ignore' checkbox unchecked. Both forms contain the following data:

Column	Type	Value
id	int(11)	
username	varchar(100)	Ali
password	varchar(225)	1234
roles	varchar(10)	admin

At the bottom of the page, a red arrow labeled '3 Fill in the details as shown' points to the 'Insert as new row' dropdown menu. Another red arrow labeled '4 Click 'Go' to save the record' points to the 'Go' button at the bottom right.

8. After you have clicked the ‘Go’ button in the previous step, you will see an autogenerated SQL query on the phpMyAdmin page. To view the data, click Browse.

2 rows inserted.  
Inserted row id: 2

**Click 'Browse' to view the record**

```
INSERT INTO `users` (`id`, `username`, `password`, `roles`) VALUES (NULL, 'Ali', '1234', 'admin'), (NULL, 'Ahmad', '4567', 'user');
```

[Edit inline] [Edit] [Create]

Run SQL query/queries on table csf3107.users:

1. `INSERT INTO `users` (`id`, `username`, `password`, `roles`) VALUES (NULL, 'Ali', '1234', 'admin'), (NULL, 'Ahmad', '4567', 'user');`

Columns

id	username	password	roles

SELECT \*    SELECT    INSERT    UPDATE    DELETE    Clear    Format

Get auto-saved query

Bind parameters

Bookmark this SQL query: [ ]

[ Delimiter : ]  Show this query here again  Retain query box  Rollback when finished  Enable foreign key checks

9. You will see the data of the users. Now, we have finished this task.

**Note:** In a real implementation, you should encrypt the password and never display the original form of it.

Showing rows 0 - 1 (2 total, Query took 0.0020 seconds.)

```
SELECT * FROM `users`
```

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	id	username	password	roles
<input type="checkbox"/>	1	Ali	1234	admin
<input type="checkbox"/>	2	Ahmad	4567	user

Check all With selected: Edit Copy Delete Export

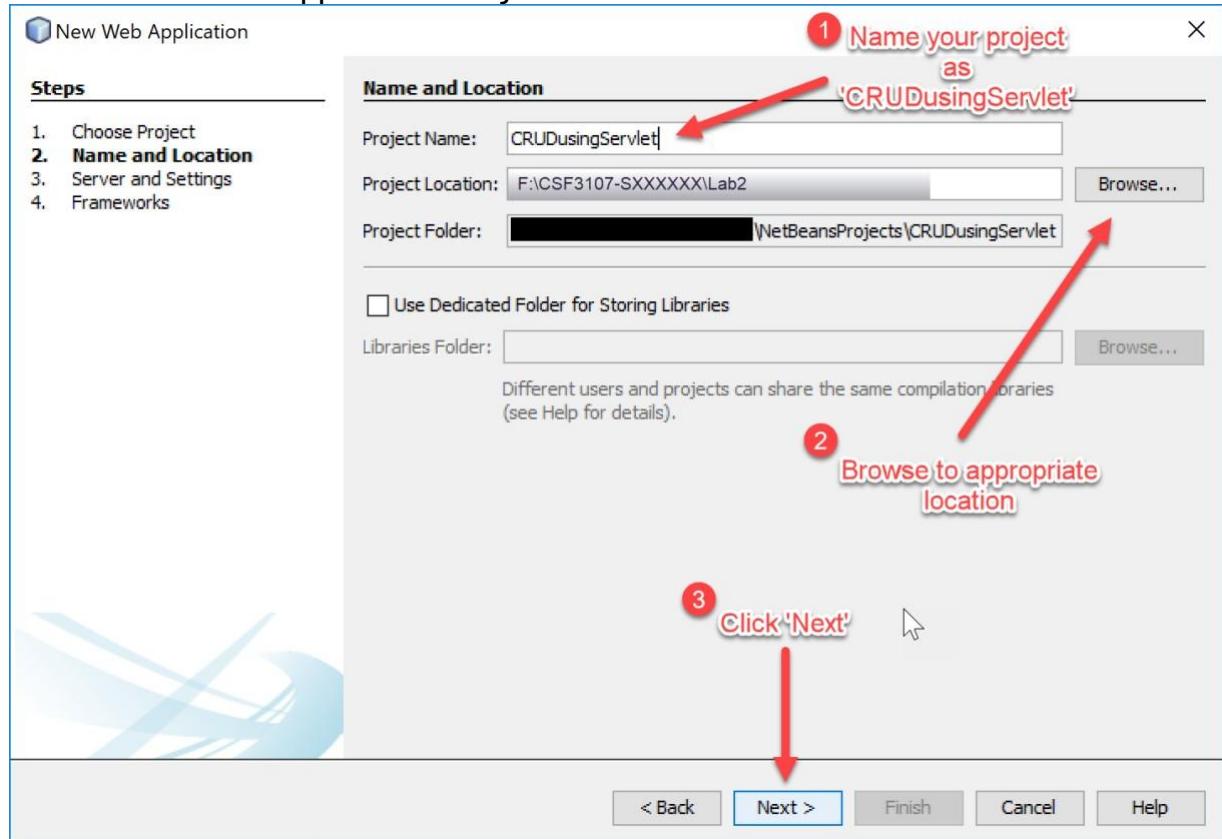
## Task 3: Setting the Environment of Web Application for Database Connection

**Objective:** To set up a proper environment for integrating web application to the database

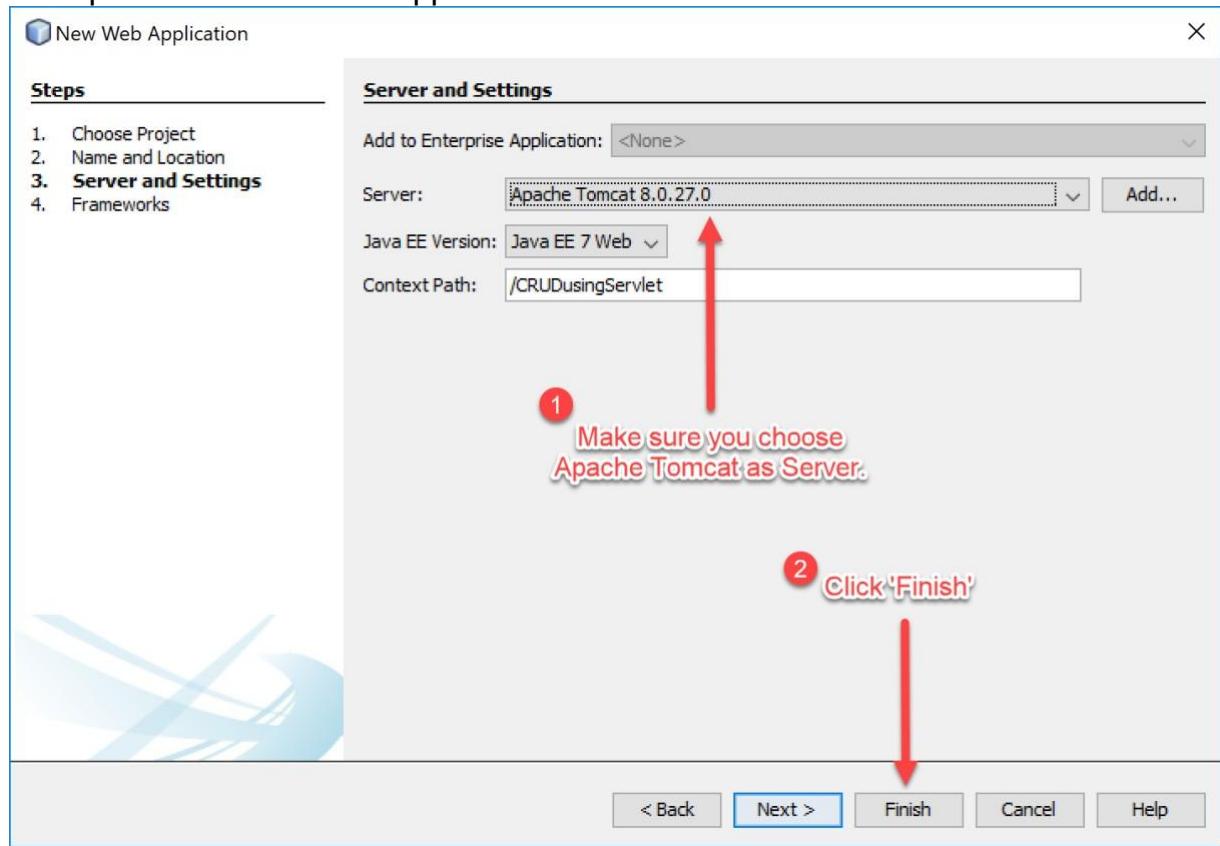
**Problem Description:** Import MySQL JDBC Library to an existing project

**Estimated time:** 5 minutes

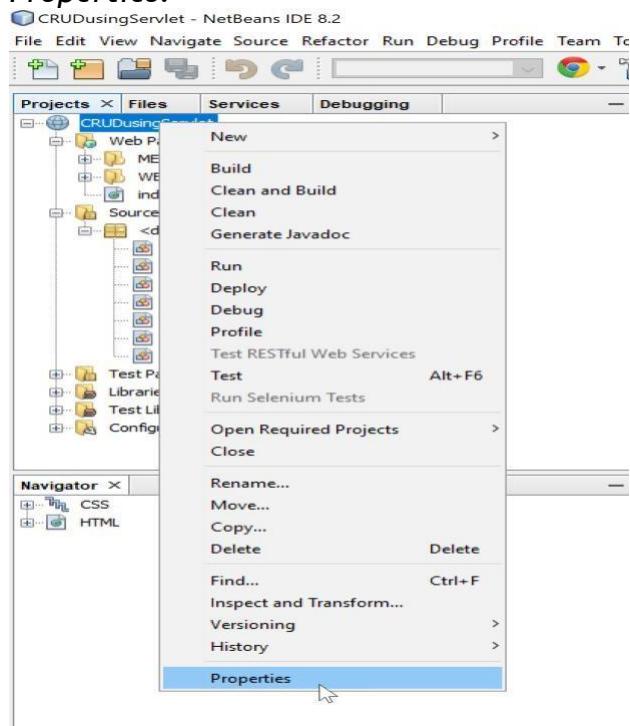
### 1. Create a new Web Application Project.



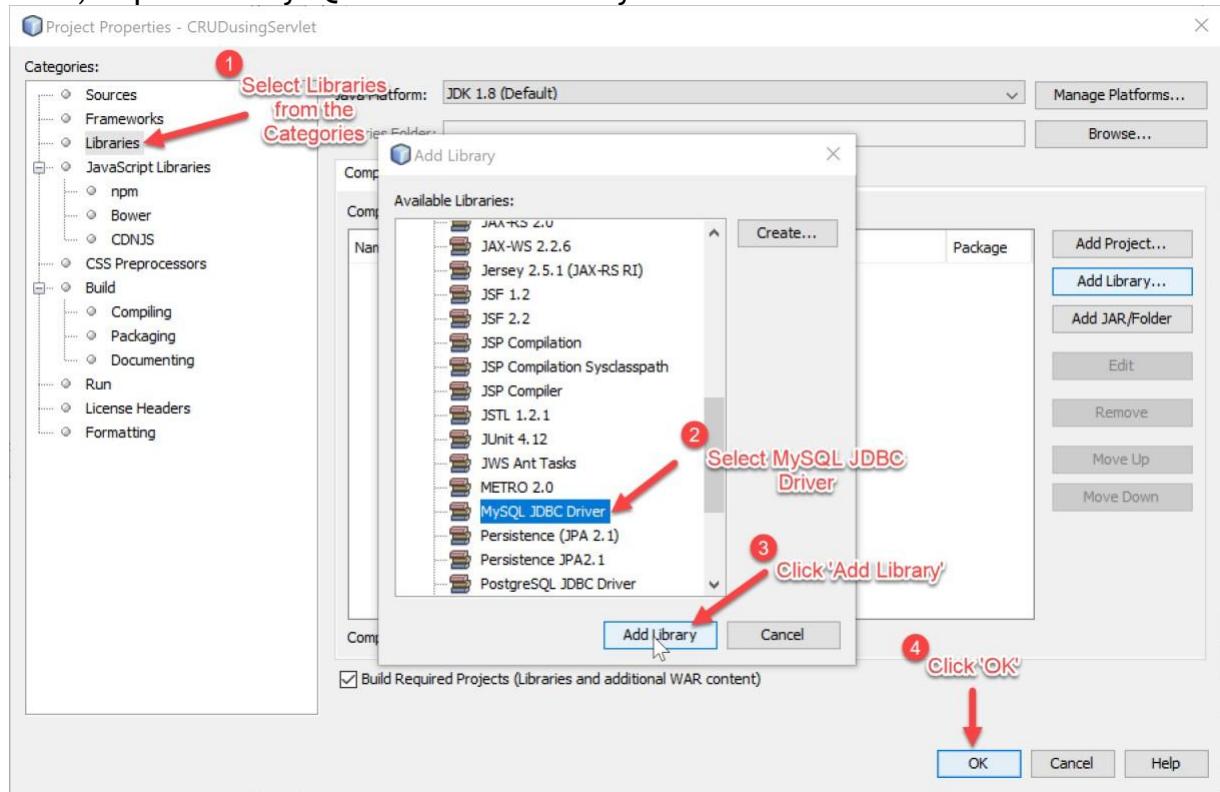
2. Use Apache Tomcat as our application server.



3. Once finished, right-click on Project's name (*CRUDUsingServlet*) and choose *Properties*.



4. Next, import the MySQL JDBC Driver library.



5. Now your application is ready to connect to MySQL database and execute a SQL query.

## Task 4: Using Servlets for Database CRUD Operations

**Objective:** To program multiple servlets for manipulating the database

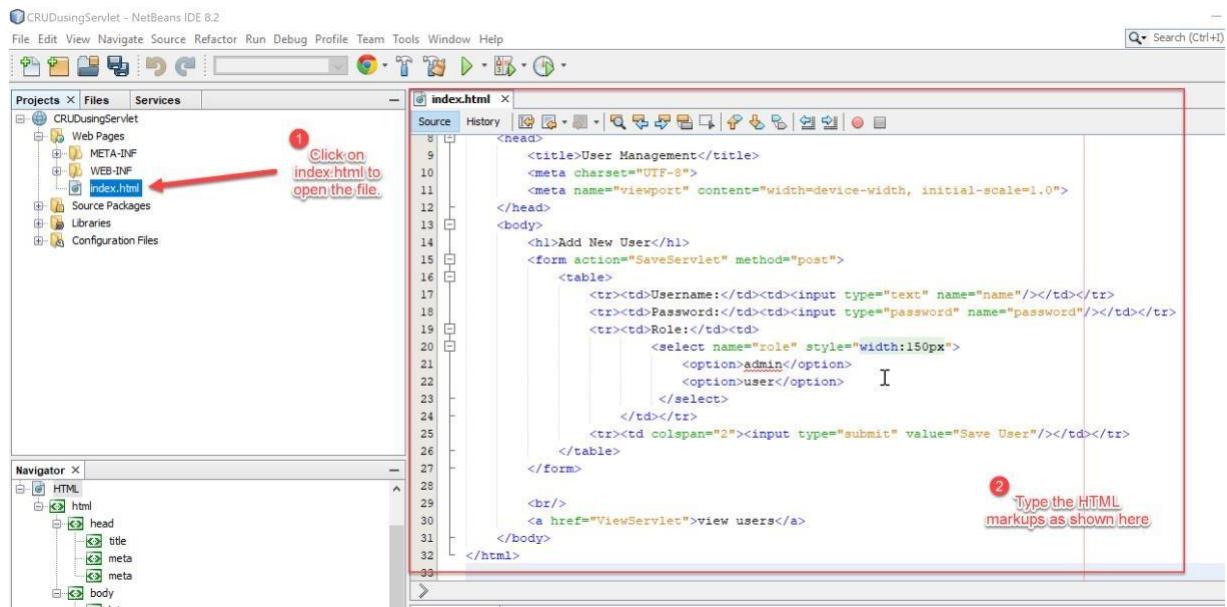
**Problem Description:** Program five different servlets to handle database operations such as insert, update and delete.

- i. SaveServlet.java: to save data into the database
- ii. ViewServlet.java: to view data retrieved from database
- iii. EditServlet.java & EditServlet2.java: to edit existing data
- iv. DeleteServlet.java: to delete existing data

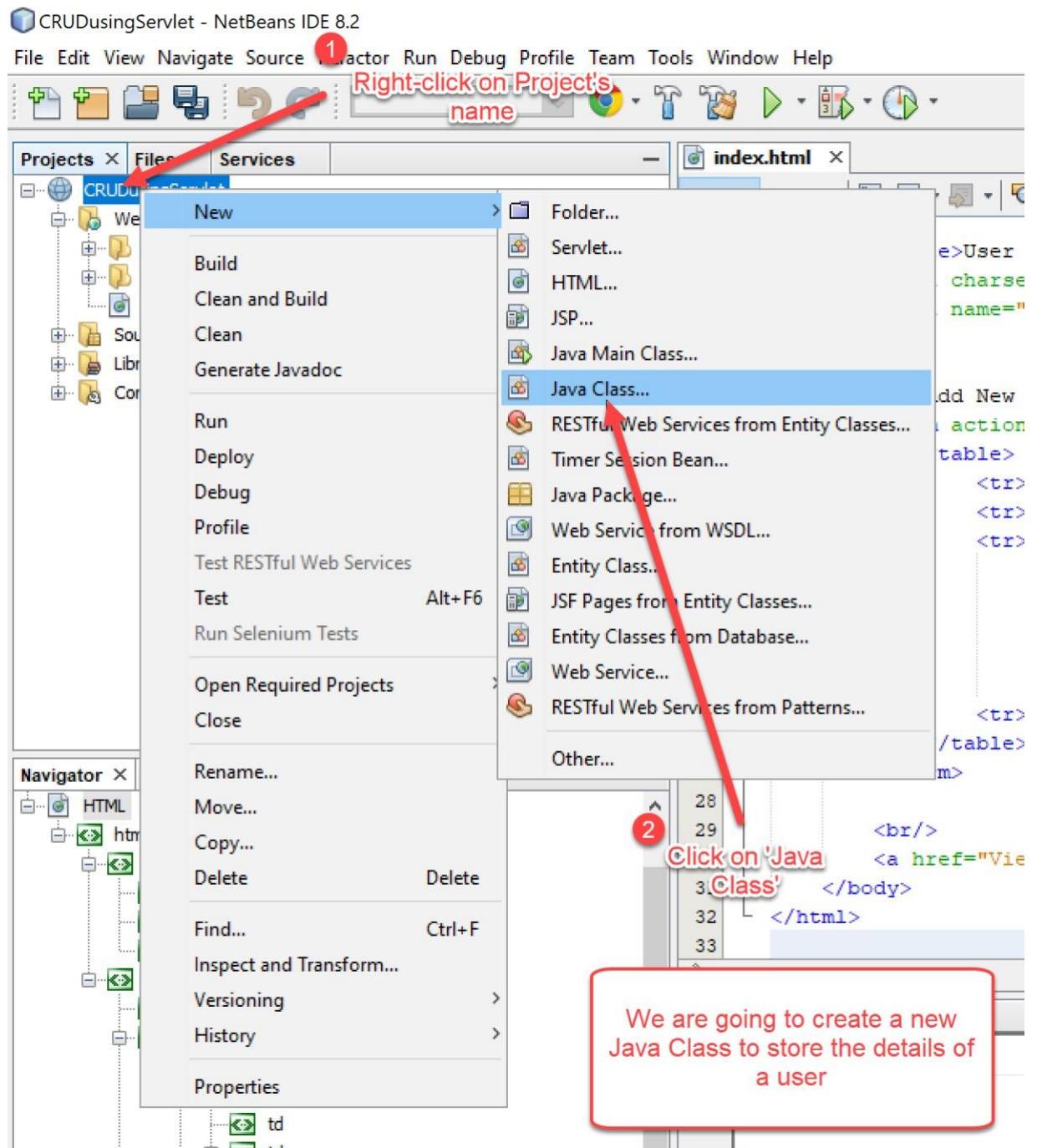
Apart from the servlets, we are going to develop two custom Java class known as JavaBeans and Data Access Object (DAO).

**Estimated time:** 90 minutes

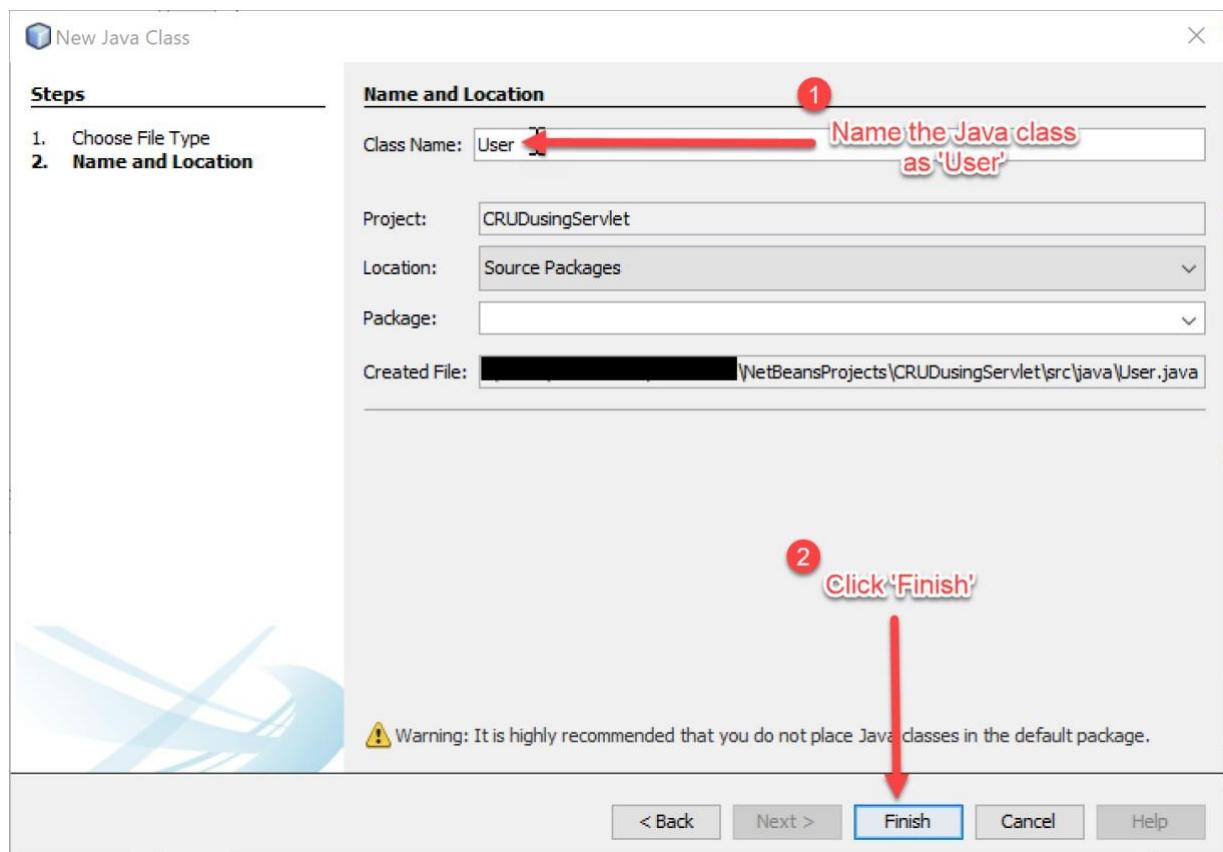
1. Open the Web Application Project (*CRUDusingServlet*) created in Task 3.
2. Open the index.html file and edit as follows.



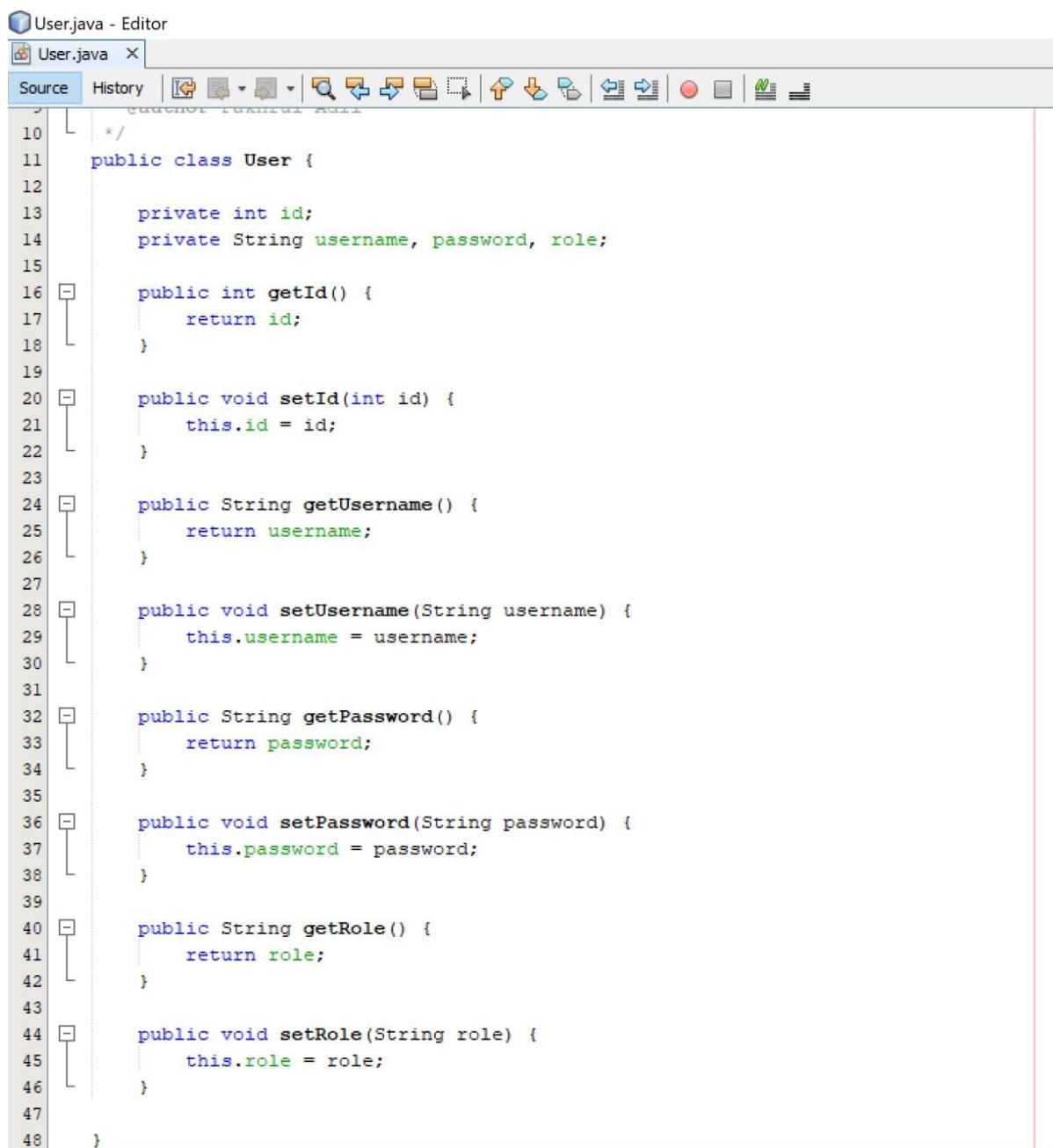
3. Now we are going to create a Java class which is specially used to store the user data.



4. Name the Java class as *User*.



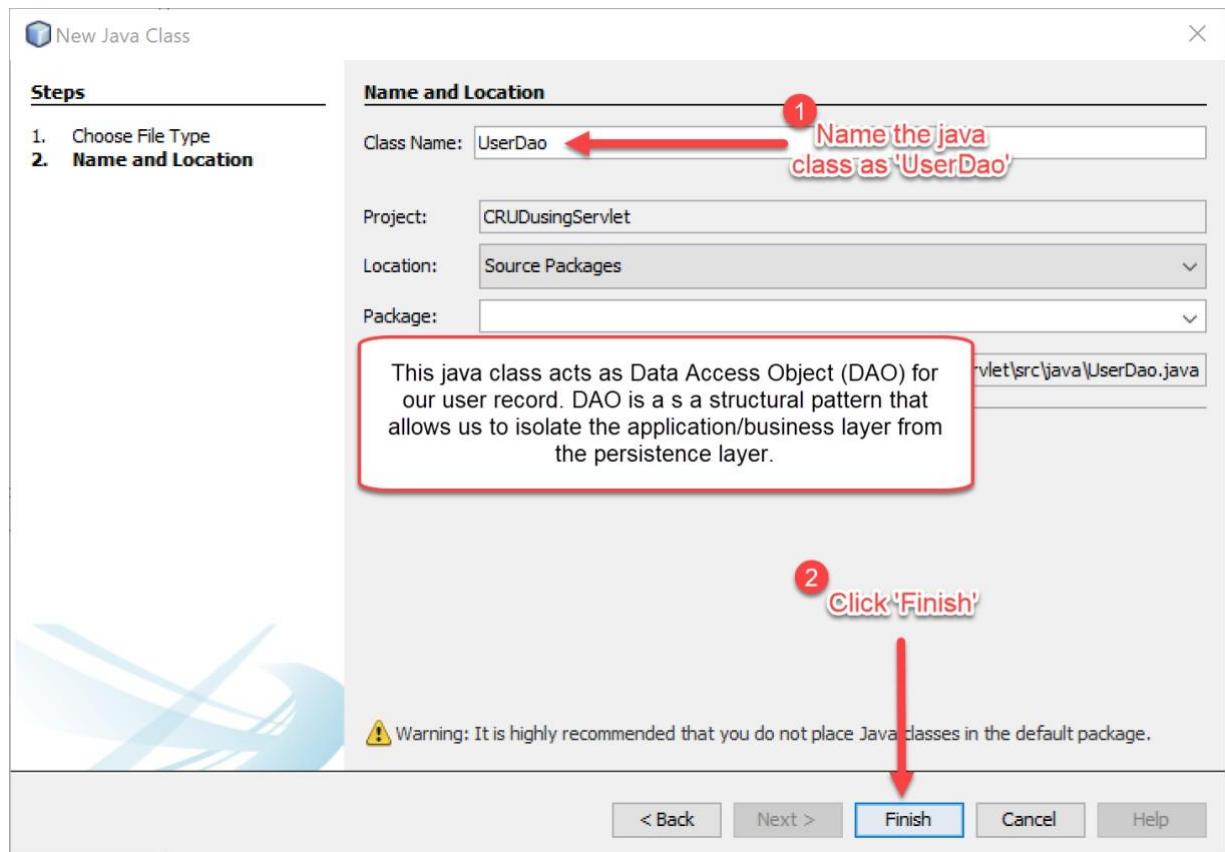
5. Type the following codes in User.java file.



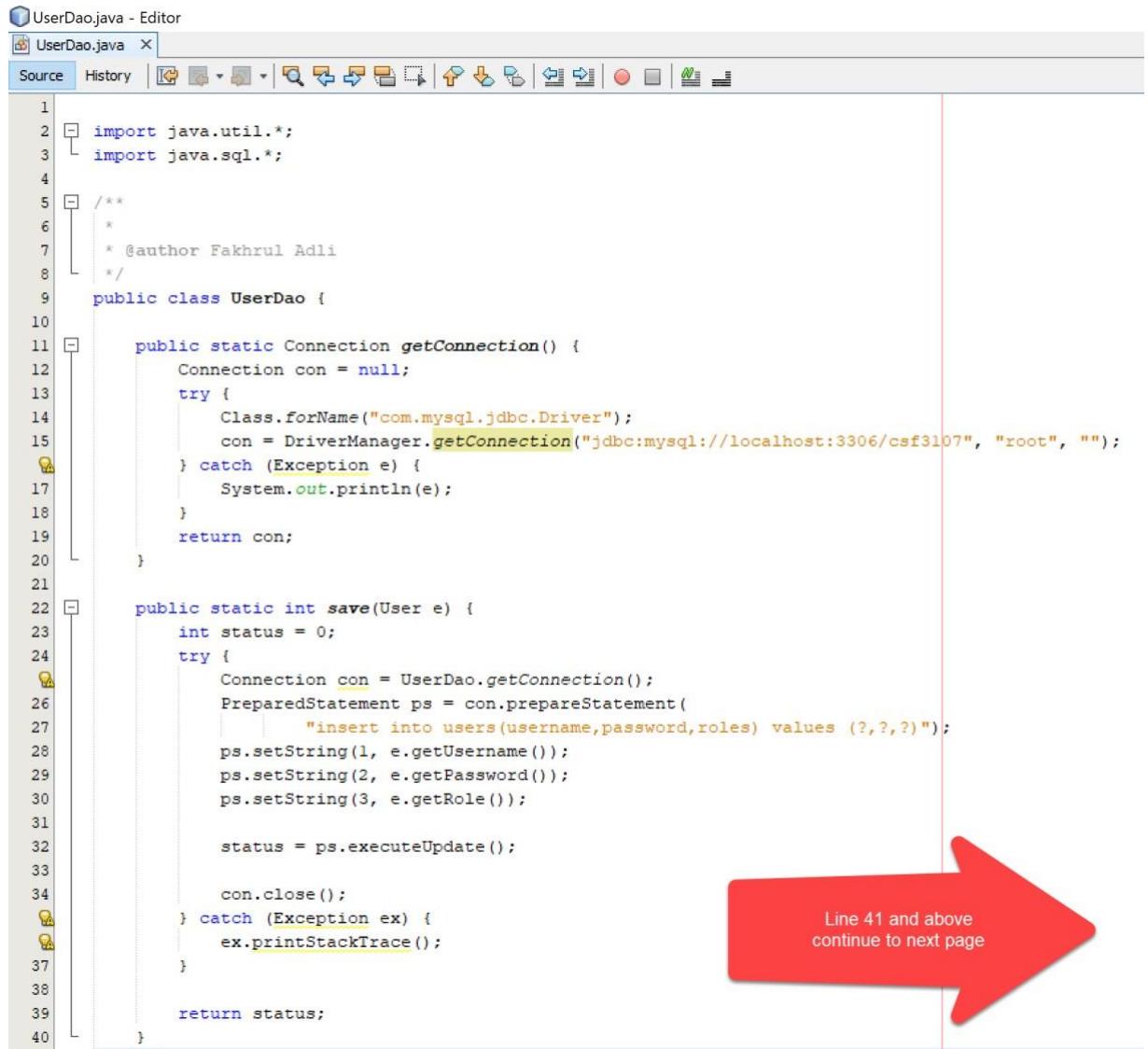
The screenshot shows a Java code editor window titled "User.java - Editor". The tab bar has "User.java" selected. The toolbar includes icons for cut, copy, paste, find, and save. The code area displays the following Java class definition:

```
10  /*
11  * User.java
12  */
13  public class User {
14
15     private int id;
16     private String username, password, role;
17
18     public int getId() {
19         return id;
20     }
21
22     public void setId(int id) {
23         this.id = id;
24     }
25
26     public String getUsername() {
27         return username;
28     }
29
30     public void setUsername(String username) {
31         this.username = username;
32     }
33
34     public String getPassword() {
35         return password;
36     }
37
38     public void setPassword(String password) {
39         this.password = password;
40     }
41
42     public String getRole() {
43         return role;
44     }
45
46     public void setRole(String role) {
47         this.role = role;
48     }
}
```

6. Next, we need to program the Data Access Object (DAO) class which is used for managing database connection and SQL operations.



7. Put the following codes in UserDao.java file. Make sure your database URL is correct, and database user credentials are also valid.



```
1 import java.util.*;
2 import java.sql.*;
3
4 /**
5  * 
6  * @author Fakhrul Adli
7  */
8 public class UserDao {
9
10    public static Connection getConnection() {
11        Connection con = null;
12        try {
13            Class.forName("com.mysql.jdbc.Driver");
14            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/csf3107", "root", "");
15        } catch (Exception e) {
16            System.out.println(e);
17        }
18        return con;
19    }
20
21
22    public static int save(User e) {
23        int status = 0;
24        try {
25            Connection con = UserDao.getConnection();
26            PreparedStatement ps = con.prepareStatement(
27                "insert into users(username,password,roles) values (?, ?, ?)");
28            ps.setString(1, e.getUsername());
29            ps.setString(2, e.getPassword());
30            ps.setString(3, e.getRole());
31
32            status = ps.executeUpdate();
33
34            con.close();
35        } catch (Exception ex) {
36            ex.printStackTrace();
37        }
38
39        return status;
40    }
}
```

Line 41 and above  
continue to next page

## 8. ...continued.

UserDao.java - Editor

UserDao.java

Source History

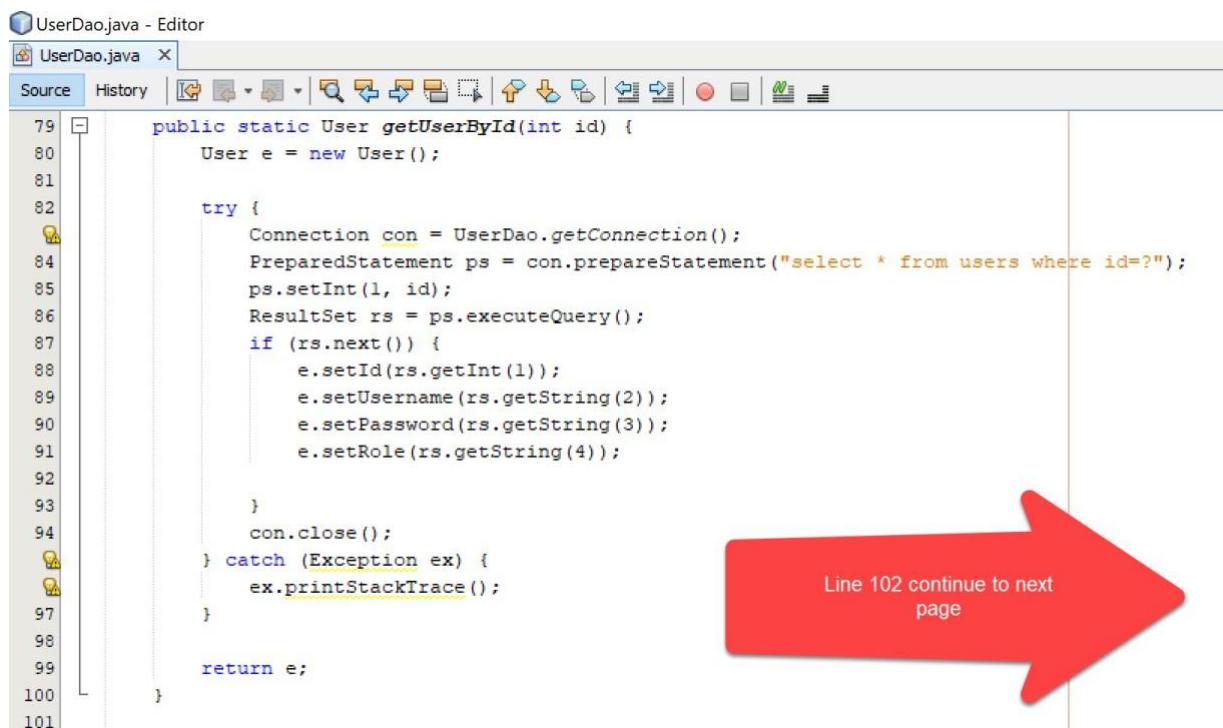
```
42     public static int update(User e) {
43         int status = 0;
44         try {
45             Connection con = UserDao.getConnection();
46             PreparedStatement ps = con.prepareStatement(
47                 "update users set username=?,password=?,roles=? where id=?");
48             ps.setString(1, e.getUsername());
49             ps.setString(2, e.getPassword());
50             ps.setString(3, e.getRole());
51             ps.setInt(4, e.getId());
52
53             status = ps.executeUpdate();
54
55             con.close();
56         } catch (Exception ex) {
57             ex.printStackTrace();
58         }
59
60         return status;
61     }
62
63     public static int delete(int id) {
64         int status = 0;
65         try {
66             Connection con = UserDao.getConnection();
67             PreparedStatement ps = con.prepareStatement("delete from users where id=?");
68             ps.setInt(1, id);
69             status = ps.executeUpdate();
70
71             con.close();
72         } catch (Exception e) {
73             e.printStackTrace();
74         }
75
76         return status;
77     }

```



Line 78 and above continue to next page

## 9. ...continued.



UserDao.java - Editor

UserDao.java

Source History

```
79     public static User getUserById(int id) {
80         User e = new User();
81
82         try {
83             Connection con = UserDao.getConnection();
84             PreparedStatement ps = con.prepareStatement("select * from users where id=?");
85             ps.setInt(1, id);
86             ResultSet rs = ps.executeQuery();
87             if (rs.next()) {
88                 e.setId(rs.getInt(1));
89                 e.setUsername(rs.getString(2));
90                 e.setPassword(rs.getString(3));
91                 e.setRole(rs.getString(4));
92
93             }
94             con.close();
95         } catch (Exception ex) {
96             ex.printStackTrace();
97         }
98
99         return e;
100    }
101 }
```

Line 102 continue to next page

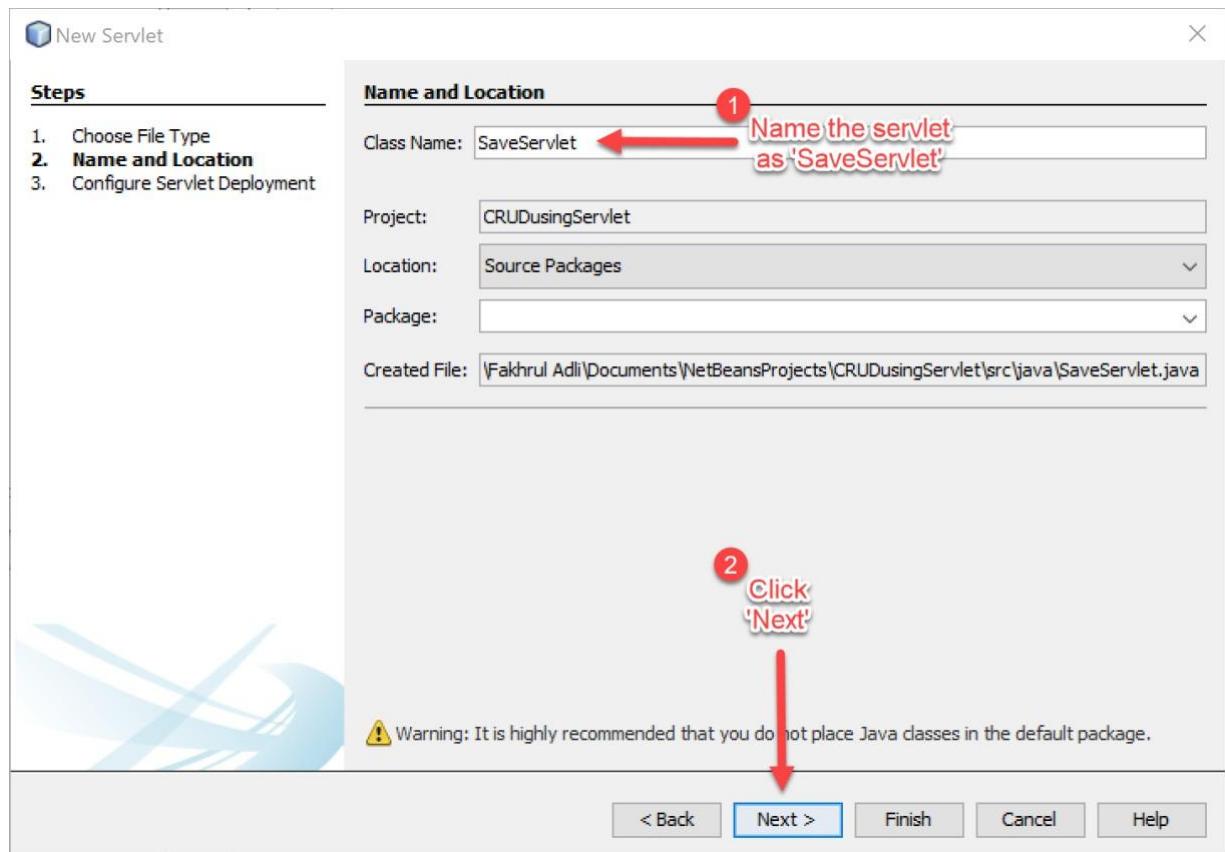
## 10. Save (CTRL+s) UserDao.java file.



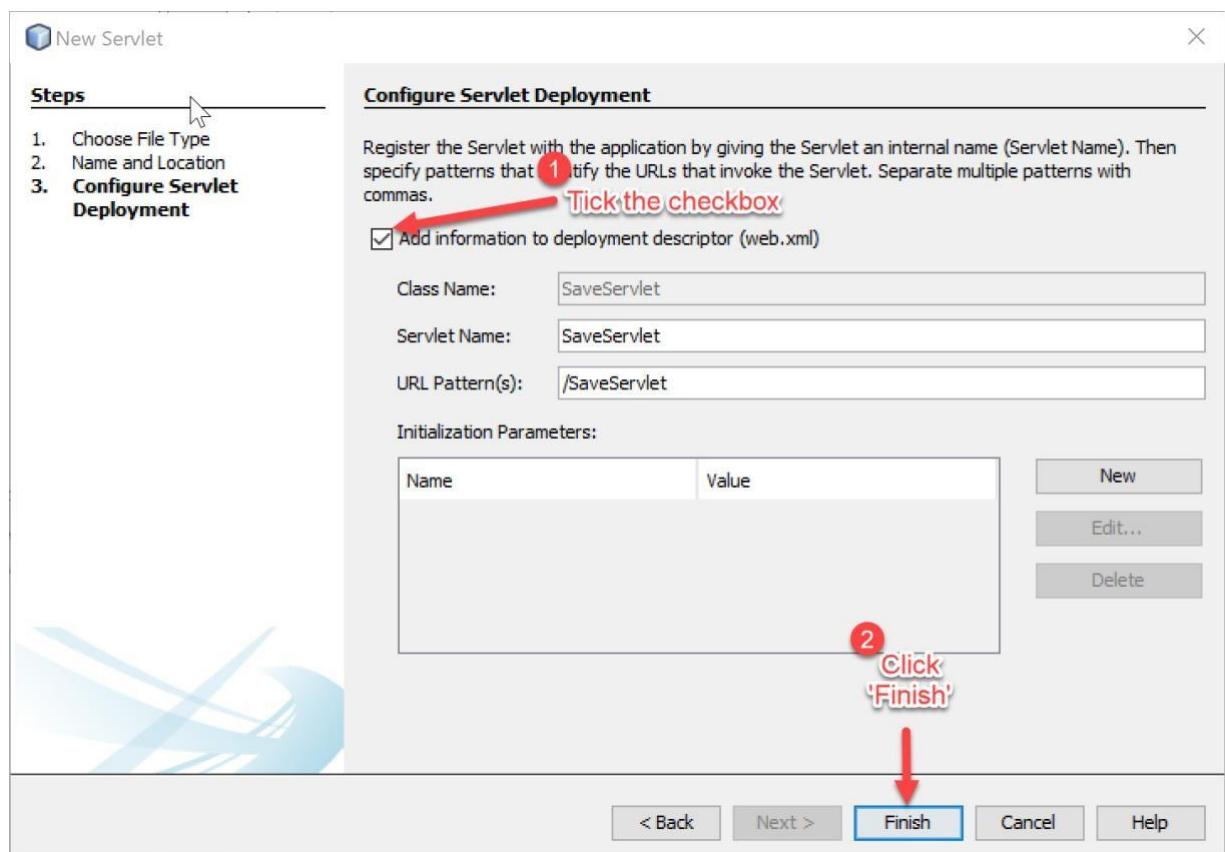
```
101
102     public static List<User> getAllUsers() {
103         List<User> list = new ArrayList<User>();
104
105         try {
106             Connection con = UserDao.getConnection();
107             PreparedStatement ps = con.prepareStatement("select * from users");
108             ResultSet rs = ps.executeQuery();
109             while (rs.next()) {
110                 User e = new User();
111                 e.setId(rs.getInt(1));
112                 e.setUsername(rs.getString(2));
113                 e.setPassword(rs.getString(3));
114                 e.setRole(rs.getString(4));
115                 list.add(e);
116             }
117             con.close();
118         } catch (Exception e) {
119             e.printStackTrace();
120         }
121
122         return list;
123     }
124 }
```

End of UserDao.java

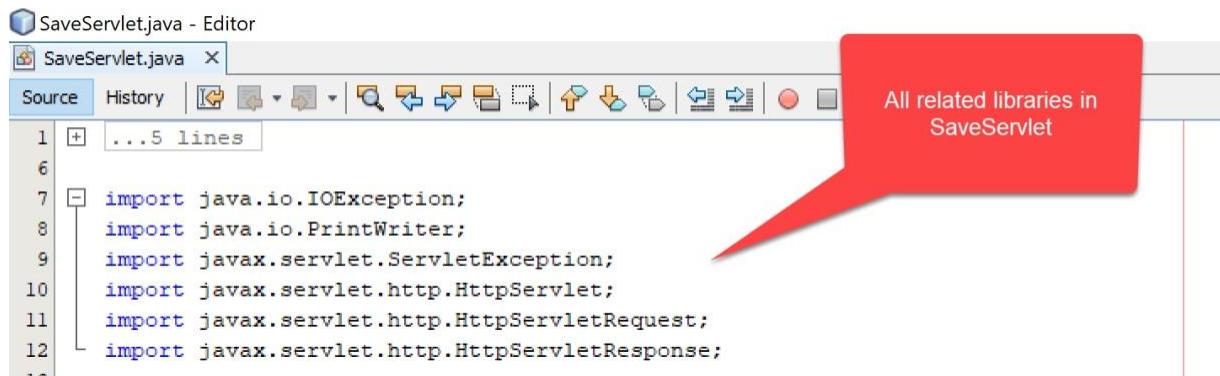
11. After finish coding the UserDao.java, now we are to code the servlet for saving data into the database.



12. Remember to tick the checkbox every time you create a new servlet.



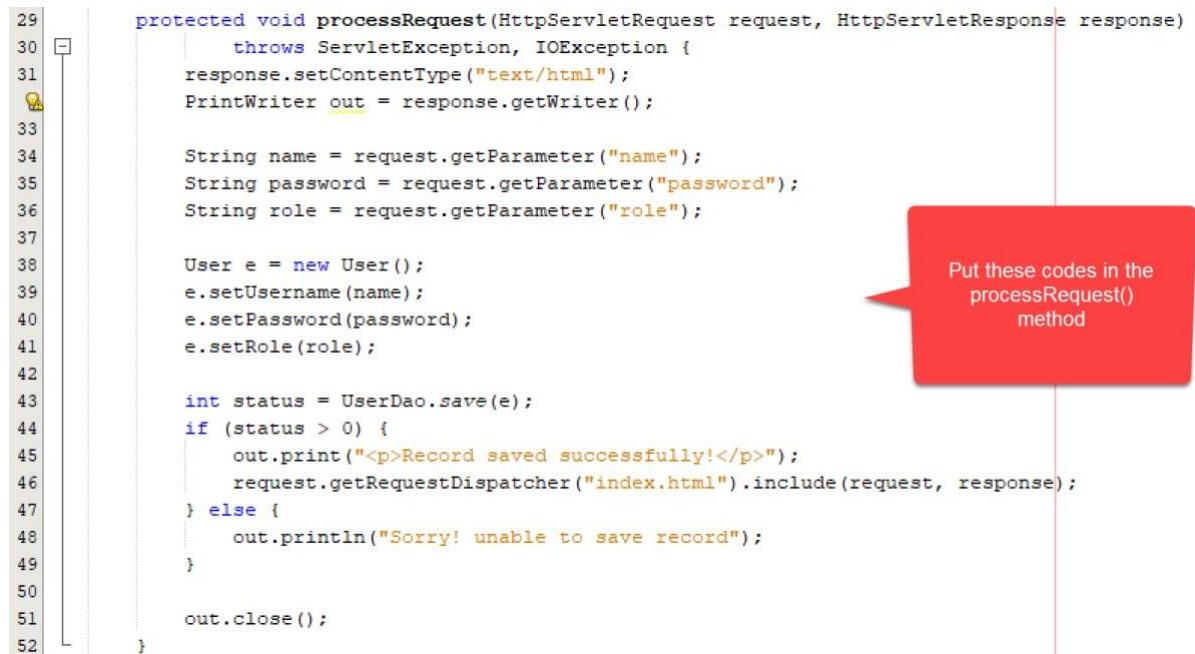
13. Make sure all related libraries are included in the SaveServlet.java file.



```
SaveServlet.java - Editor
SaveServlet.java X
Source History | ... 5 lines | 
1 import java.io.IOException;
2 import java.io.PrintWriter;
3 import javax.servlet.ServletException;
4 import javax.servlet.http.HttpServlet;
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
```

All related libraries in SaveServlet

14. Type the following codes in processRequest() method.



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String name = request.getParameter("name");
    String password = request.getParameter("password");
    String role = request.getParameter("role");

    User e = new User();
    e.setUsername(name);
    e.setPassword(password);
    e.setRole(role);

    int status = UserDao.save(e);
    if (status > 0) {
        out.print("<p>Record saved successfully!</p>");
        request.getRequestDispatcher("index.html").include(request, response);
    } else {
        out.println("Sorry! unable to save record");
    }
    out.close();
}
```

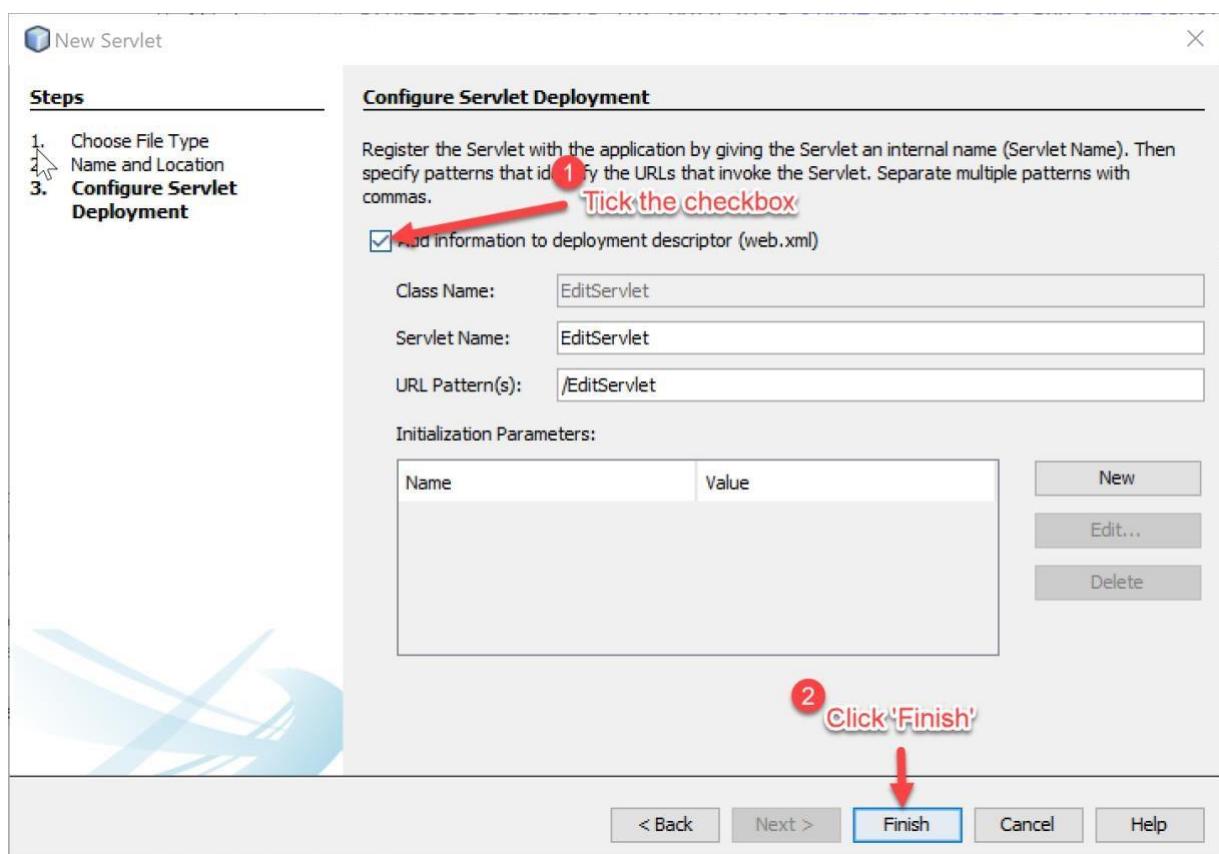
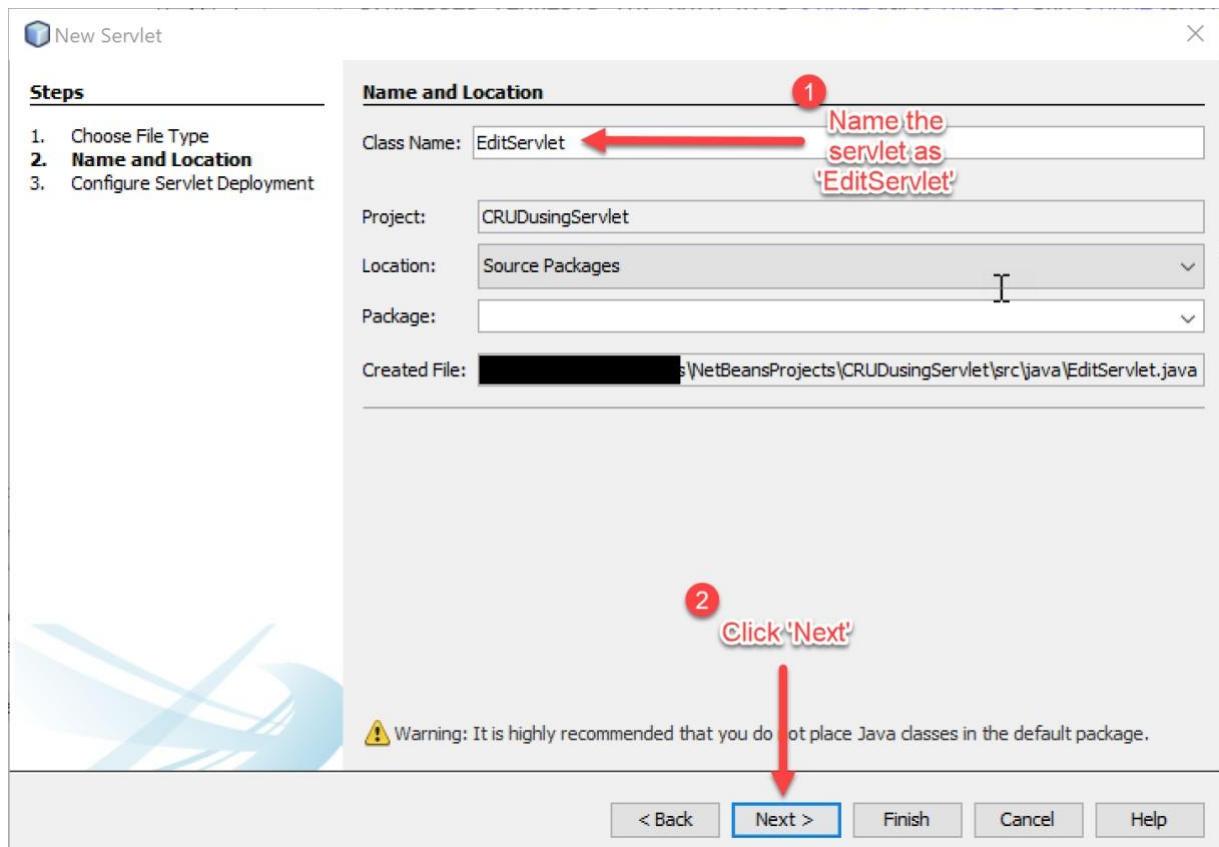
Put these codes in the processRequest() method

15. Save the file after finished.

16. Next, we are going to create a servlet to view our data in database. Create a new servlet *ViewServlet*, tick ‘Add information to deployment descriptor (web.xml)’ and click finish. Insert the following codes to the servlet.

Put these codes in  
processRequest  
method

17. Currently, we have finished creating a servlet for saving and viewing the data in the database. Now, we need a servlet to handle the process of editing the existing data stored in the table in the database.



18. Type the following codes in the servlet you just created.

EditServlet.java - Editor

EditServlet.java

Source History

```

7 import java.io.IOException;
8 import java.io.PrintWriter;
9 import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 /**
15  * Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
16 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
17     throws ServletException, IOException {
18     response.setContentType("text/html");
19     PrintWriter out=response.getWriter();
20     out.println("<h1>Update User</h1>");
21     String sid=request.getParameter("id");
22     int id=Integer.parseInt(sid);
23
24     User e=UserDao.getUserById(id);
25
26     out.print("<form action='EditServlet2' method='post'>");
27     out.print("<table>");
28     out.print("<tr><td></td><td><input type='hidden' name='id' value='"
29             +e.getId()+"'></td></tr>");
30     out.print("<tr><td>Name:</td><td><input type='text' name='username' value='"
31             +e.getUsername()+"'></td></tr>");
32     out.print("<tr><td>Password:</td><td><input type='password' name='password' value='"
33             +e.getPassword()+"'></td></tr>");
34     out.print("<tr><td>Role:</td><td>");
35     out.print("<select name='role' style='width:150px'>");
36     out.print("<option>admin</option>");
37     out.print("<option>user</option>");
38     out.print("</select>");
39     out.print("</td></tr>");
40     out.print("<tr><td colspan='2'><input type='submit' value='Edit & Save' /></td></tr>");
41     out.print("</table>");
42     out.print("</form>");
43
44     out.close();
45 }
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

Make EditServlet.java looks like this

19. The previous servlet for editing data is for displaying the form as below:

← → ⌂ i localhost:8084/CRUDusingServlet/EditServlet?id=1

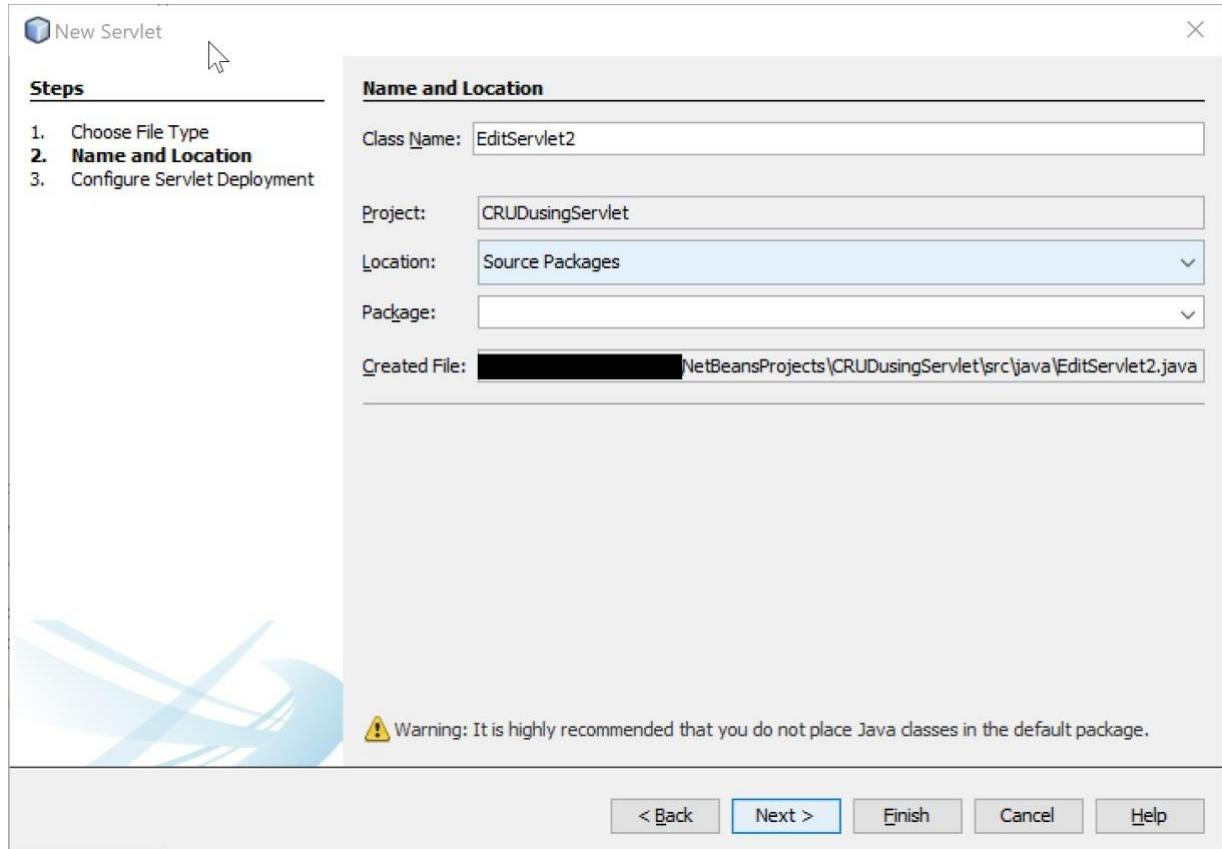
## Update User

Name:

Password:

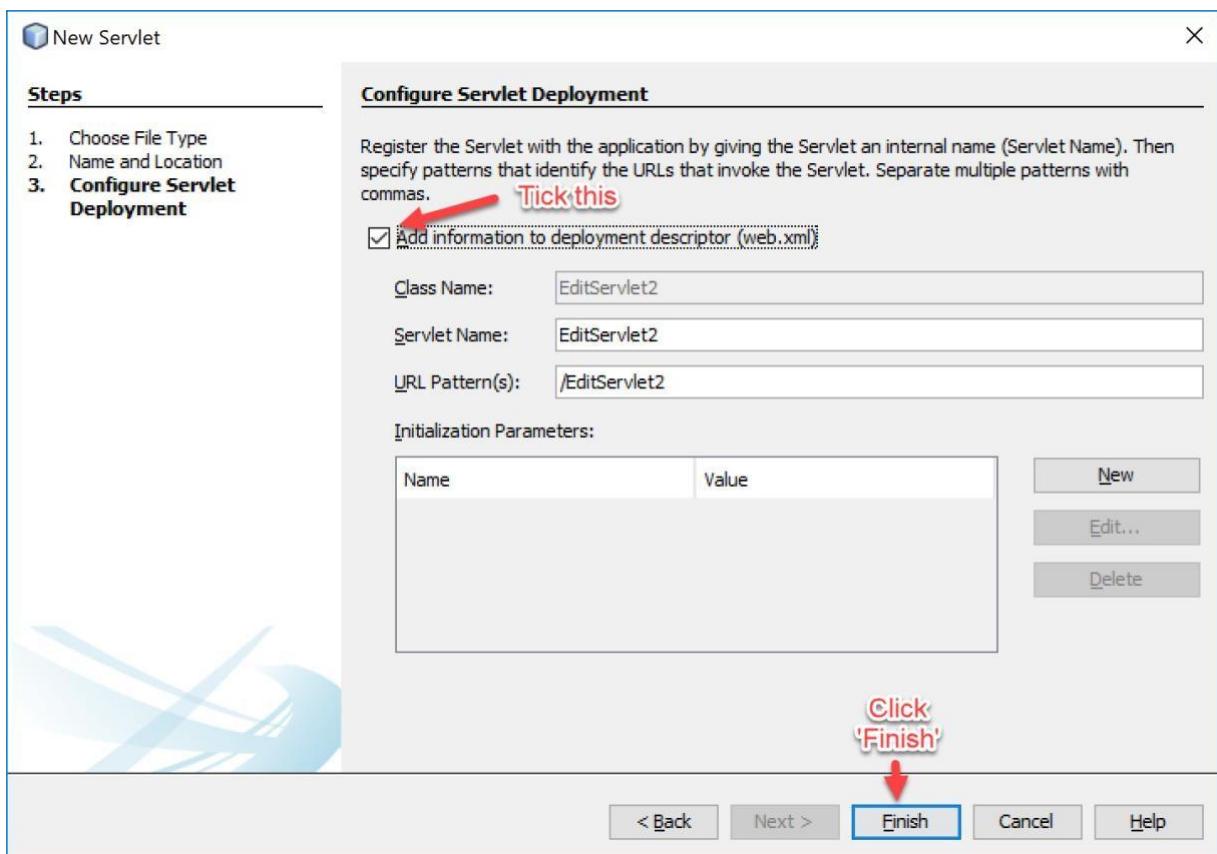
Role:

20. Now, we need another servlet to handle database processing.



21. Remember to tick ‘Add information to deployment descriptor (web.xml)’.

Netbeans will configure the servlet in web.xml on your behalf.



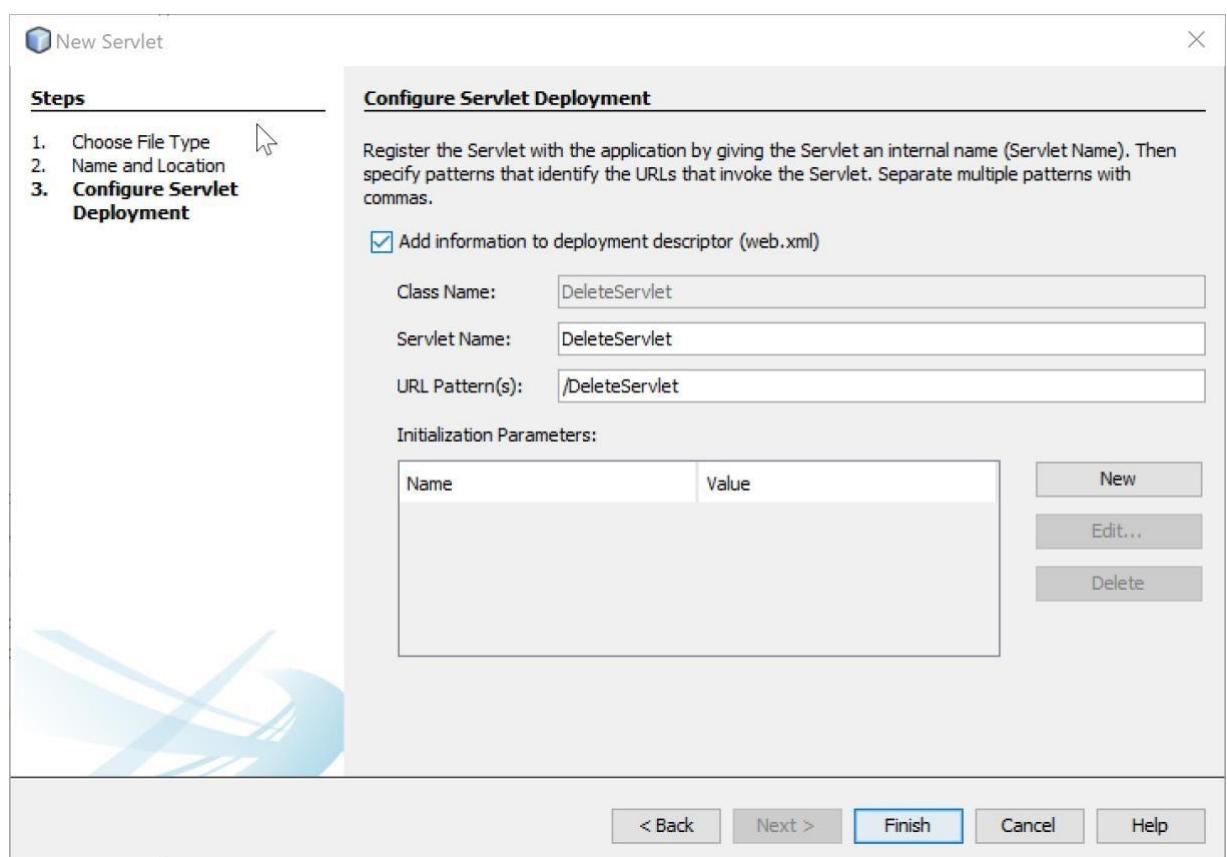
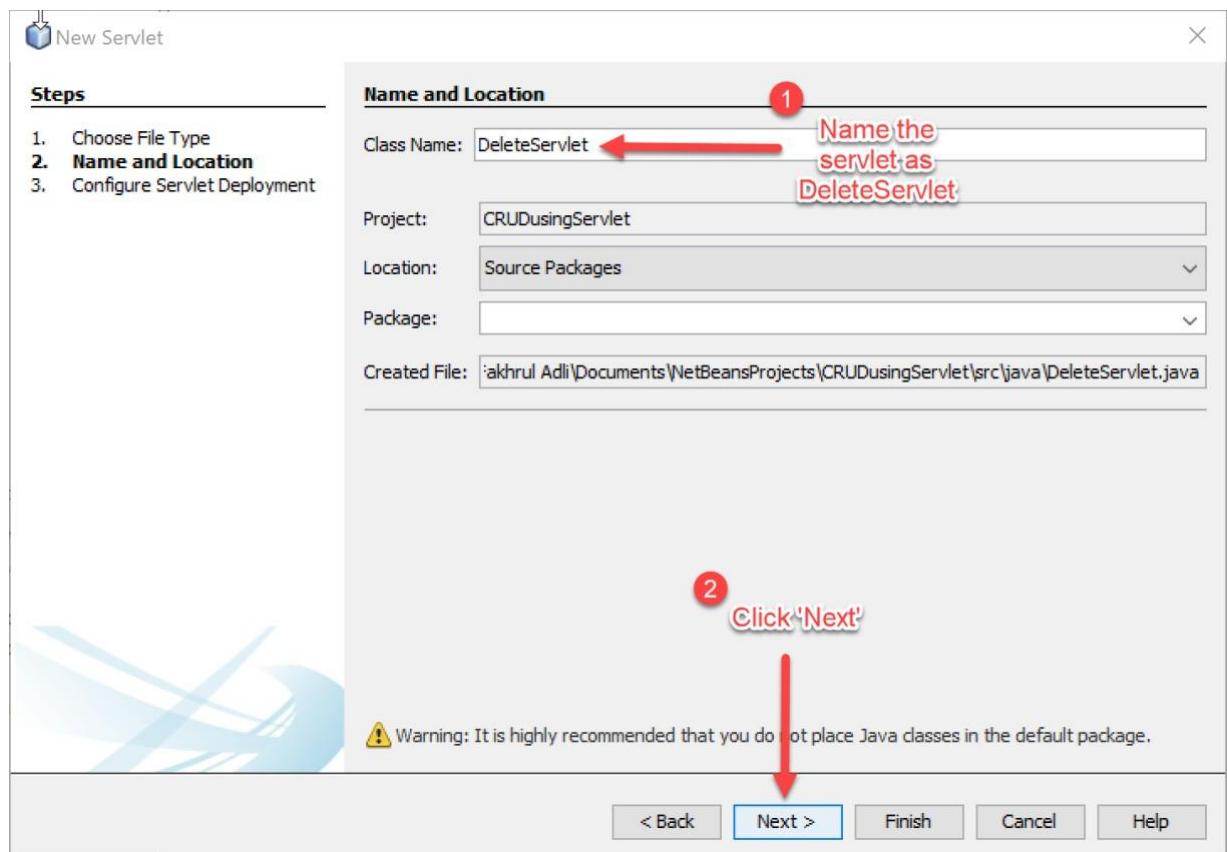
22. Type the following codes in the processRequest() method in

EditServlet2.java. Save the file when you have finished.

```
29 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
30     throws ServletException, IOException {
31     response.setContentType("text/html");
32     PrintWriter out = response.getWriter();
33
34     String name = request.getParameter("name");
35     String password = request.getParameter("password");
36     String role = request.getParameter("role");
37
38     User e = new User();
39     e.setUsername(name);
40     e.setPassword(password);
41     e.setRole(role);
42
43     int status = UserDao.save(e);
44     if (status > 0) {
45         out.print("<p>Record saved successfully!</p>");
46         request.getRequestDispatcher("index.html").include(request, response);
47     } else {
48         out.println("Sorry! unable to save record");
49     }
50
51     out.close();
52 }
```

A red callout bubble on the right side of the code editor says 'Put these codes in the processRequest() method'.

23. Now we already have servlets for edit and saving the data, so the last servlet we need to provide is a servlet for deleting the data.



24. Write the codes below into your DeleteServlet.java file. Save the file when you have done.

25. Run all the CRUD process of your servlet applications. In the end, you should be able to (C)reate, (R)etrieve, (U)pdate and (D)elete the data in the database.

## Reflections:

- What is the name of the Java Library that you need to import before coding the web application with database operations?

## Answer: mySQL connector

2. Which folder keeps the web.xml file? Copy the contents of the file and explain in brief the tags included such as <servlet-name><servlet-class> <servlet-mapping>. etc.

**Answer: WEB-INF folder.**

## **Content of files:**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="6.0" xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
  https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd">
```

```
<!-- Servlet Configuration -->
```

This section defines servlets in the application. Each servlet element consists of a servlet-name and a servlet-class.

```

-->
<servlet>
    < servlet-name>SaveServlet</servlet-name> <!-- Name of the servlet -->
    < servlet-class>com.mycompany.crudusing servlet.SaveServlet</servlet-class> <!--
- Class implementing the servlet -->
</servlet>

<servlet>
    < servlet-name>ViewServlet</servlet-name>
    < servlet-class>com.mycompany.crudusing servlet.ViewServlet</servlet-class>
</servlet>

<servlet>
    < servlet-name>EditServlet</servlet-name>
    < servlet-class>com.mycompany.crudusing servlet.EditServlet</servlet-class>
</servlet>

<servlet>
    < servlet-name>EditServlet2</servlet-name>
    < servlet-class>com.mycompany.crudusing servlet.EditServlet2</servlet-class>
</servlet>

<servlet>
    < servlet-name>DeleteServlet</servlet-name>
    < servlet-class>com.mycompany.crudusing servlet.DeleteServlet</servlet-class>
</servlet>

<!-- Servlet Mapping -->
<!--
    This section maps servlets to URL patterns, defining which servlet
    should handle requests for specific URLs.
-->
<servlet-mapping>
    < servlet-name>SaveServlet</servlet-name> <!-- Servlet name to be mapped -->
    < url-pattern>/SaveServlet</url-pattern> <!-- URL pattern for the servlet -->
</servlet-mapping>

<servlet-mapping>
    < servlet-name>ViewServlet</servlet-name>
    < url-pattern>/ViewServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
    < servlet-name>EditServlet</servlet-name>
    < url-pattern>/EditServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
    < servlet-name>EditServlet2</servlet-name>
    < url-pattern>/EditServlet2</url-pattern>
</servlet-mapping>

```

```

<servlet-mapping>
    <servlet-name>DeleteServlet</servlet-name>
    <url-pattern>/DeleteServlet</url-pattern>
</servlet-mapping>

<!-- Session Configuration -->
<!--
    This section configures session-related settings for the application.
    It sets the session timeout value in minutes.
-->
<session-config>
    <session-timeout>
        30 <!-- Session timeout in minutes -->
    </session-timeout>
</session-config>

</web-app>

```

3. Define the usage of Data Access Object (DAO) servlet. How it ease the business process in your servlet-based web application?

**Answer:** A Data Access Object (DAO) servlet is a design pattern used in web applications to separate the business logic of an application from the persistence layer responsible for interacting with the database. It essentially acts as an intermediary between the servlets (or controllers) and the database, providing a clean interface for performing CRUD (Create, Read, Update, Delete) operations on the database. It helps in Abstraction of Data Access, Encapsulation of Database Operations and Enhanced Security.