



UNIVERSITI MALAYSIA TERENGGANU

CSM3023 WEB-BASED APPLICATION DEVELOPMENT (K1)

BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONORS

LAB 2 - SERVLET: DATA SHARING AND DATABASE MANAGEMENT

SEMESTER II 2023/2024

Prepared for:

DR. MOHAMAD NOR HASSAN

Prepared by:

LUQMAN HAKIM BIN AZIZ

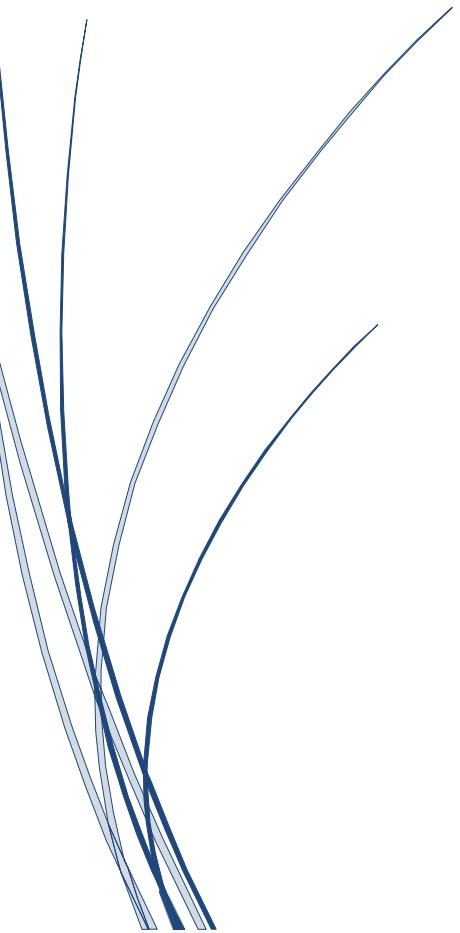
(S66292)



Week 2

Servlet: Data Sharing and Database Management

Web Programming 2



Lecturers

PUSAT PENGAJIAN INFORMATIK DAN MATEMATIK GUNAAN
(PPIMG), UNIVERSITI MALAYSIA TERENGGANU (UMT)

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

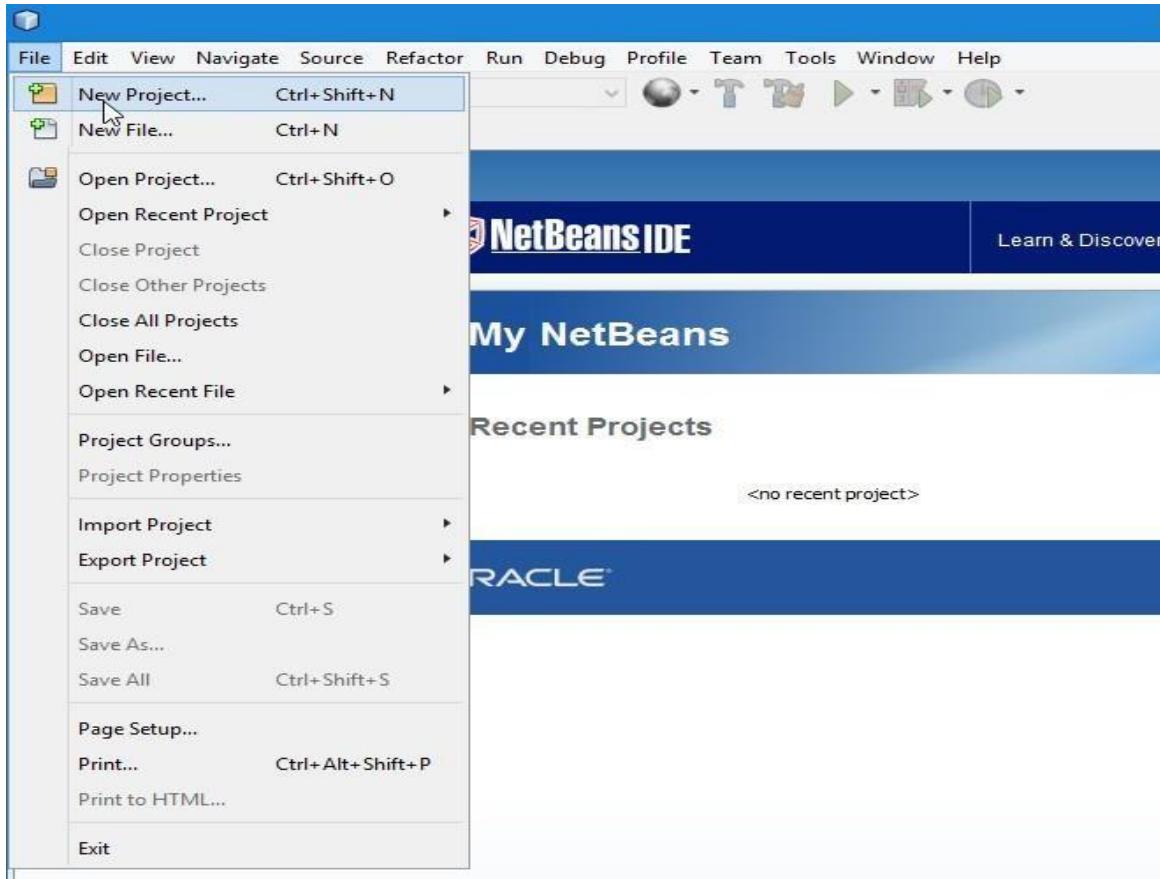
Task 1: Data Sharing in Servlet

Objective: To use servlet for request forwarding and data sharing

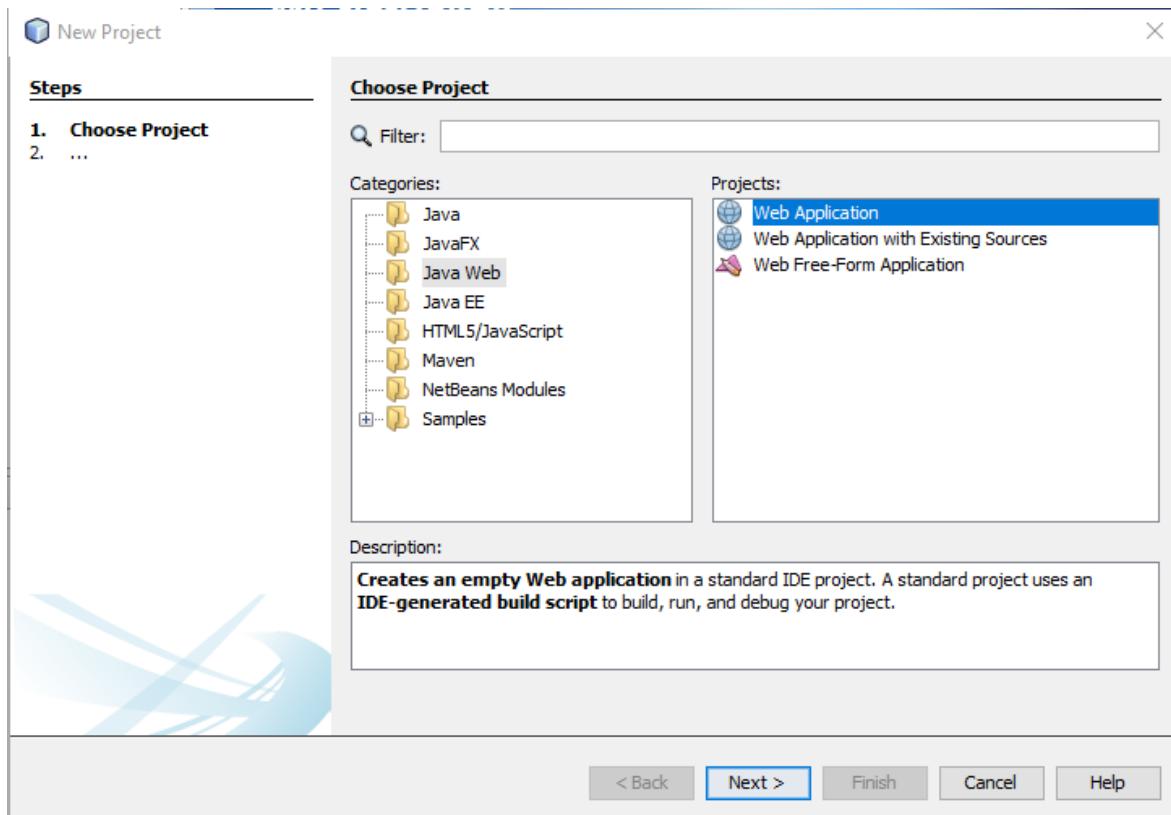
Problem Description: Write a login form and a servlet to authenticate a user.

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

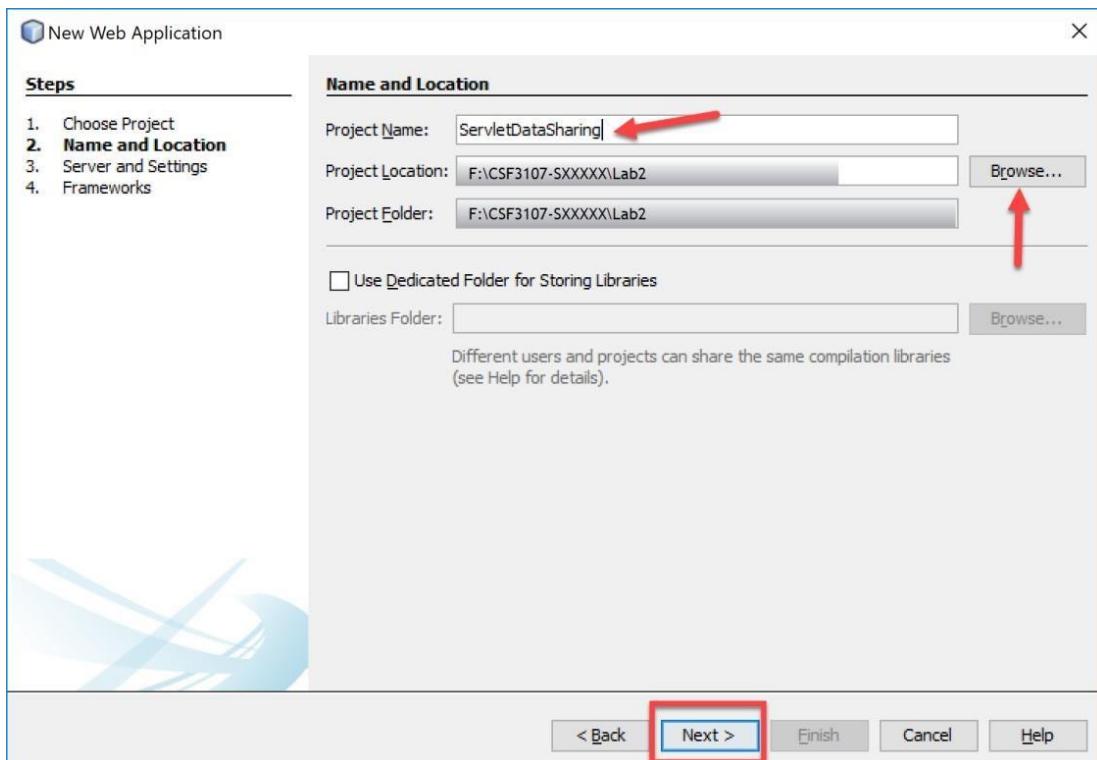


5. Select Java Web -> Web Application and click Next.

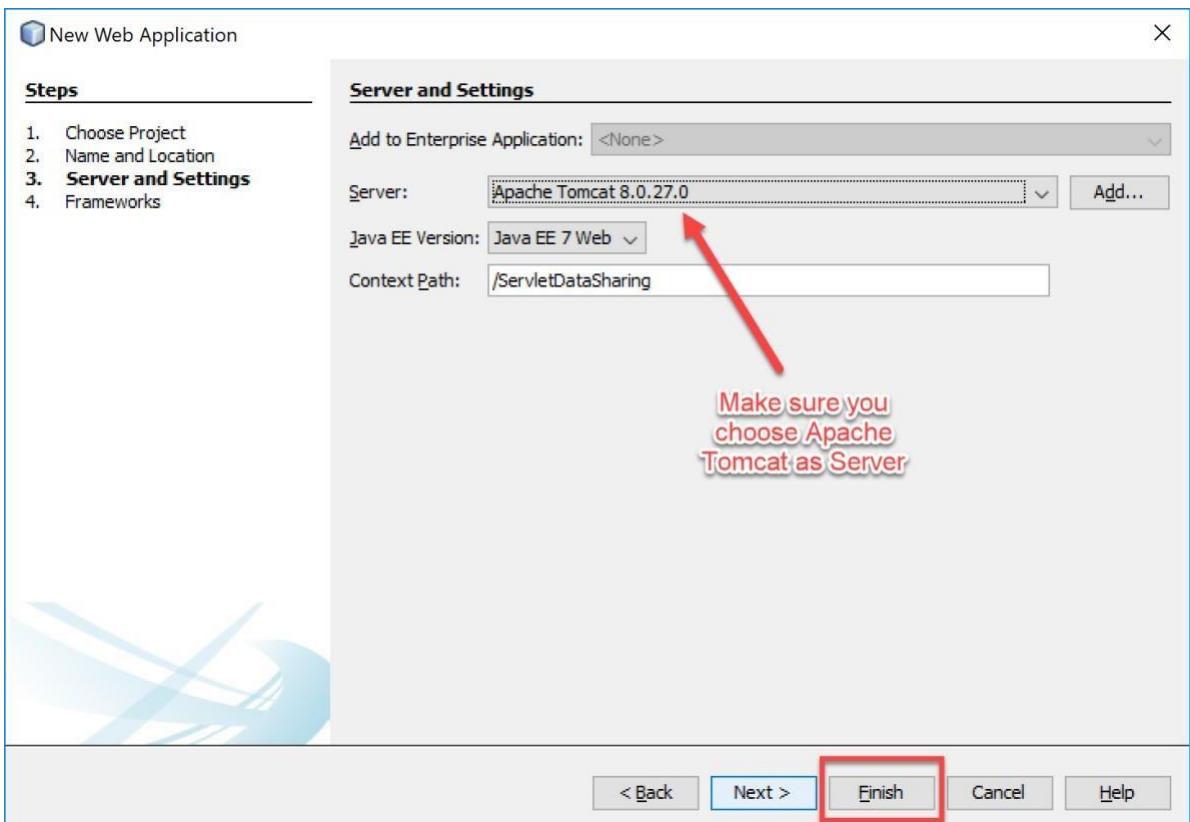


6. Type Project Name: *ServletDataSharing*.

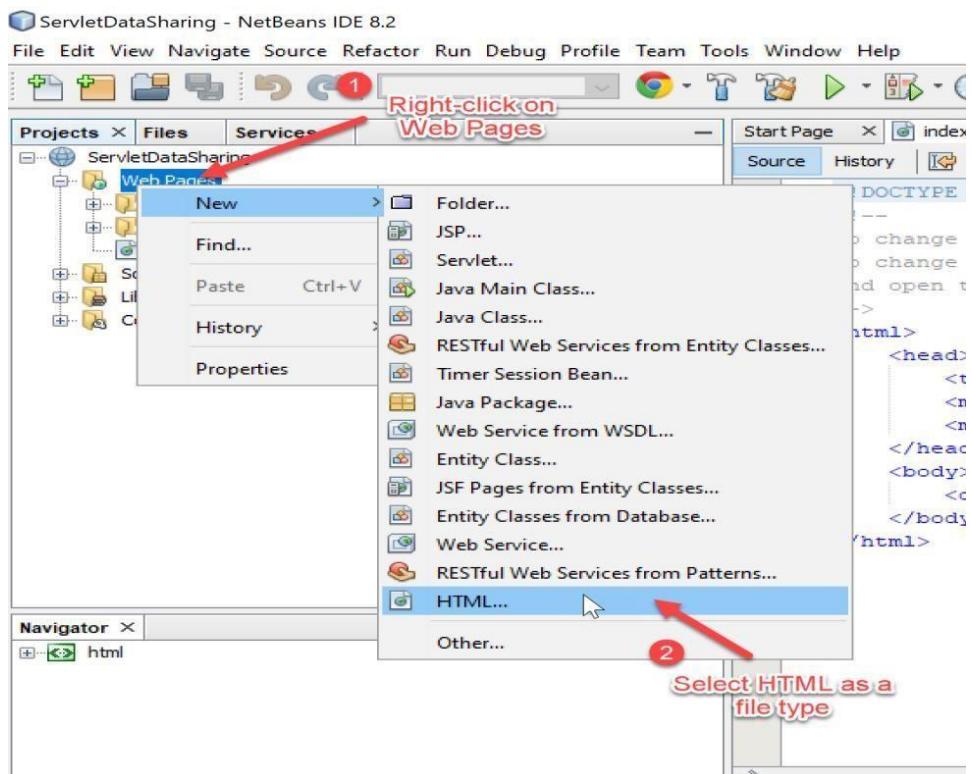
7. Click Browse and choose Project Location: F:\CSF3107-SXXXXX\Lab2. Then click the Next button.



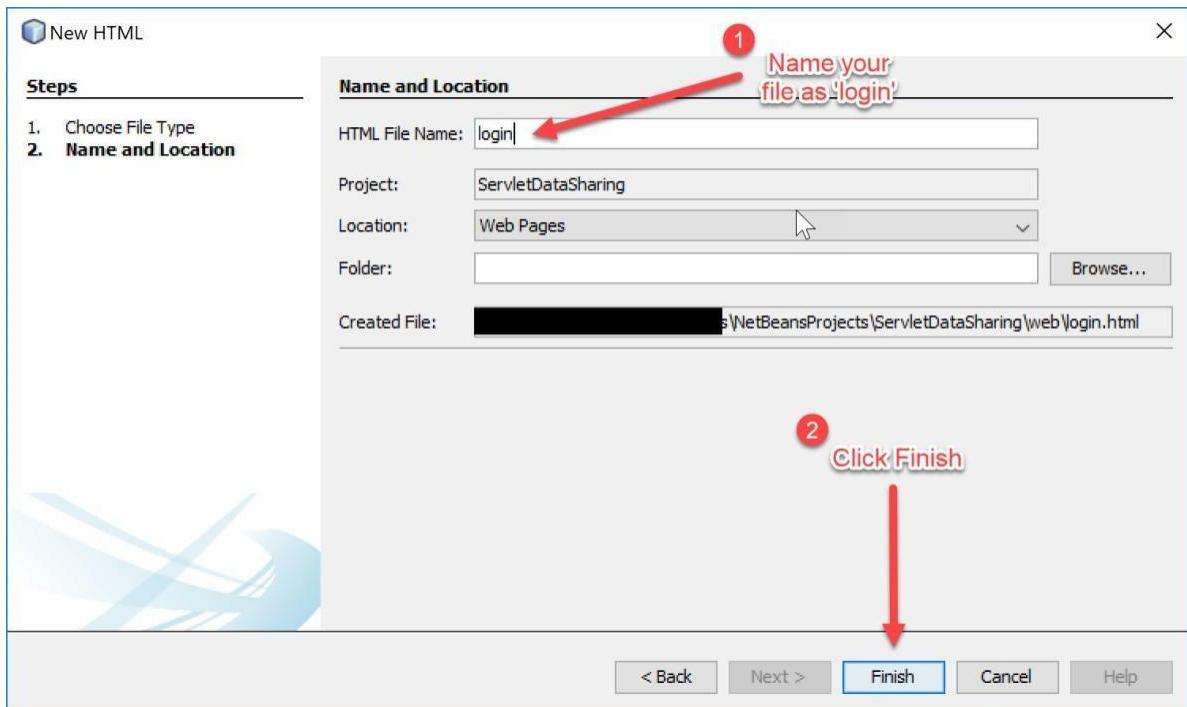
8. Then, choose Apache Tomcat as the server and click Finish.



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.



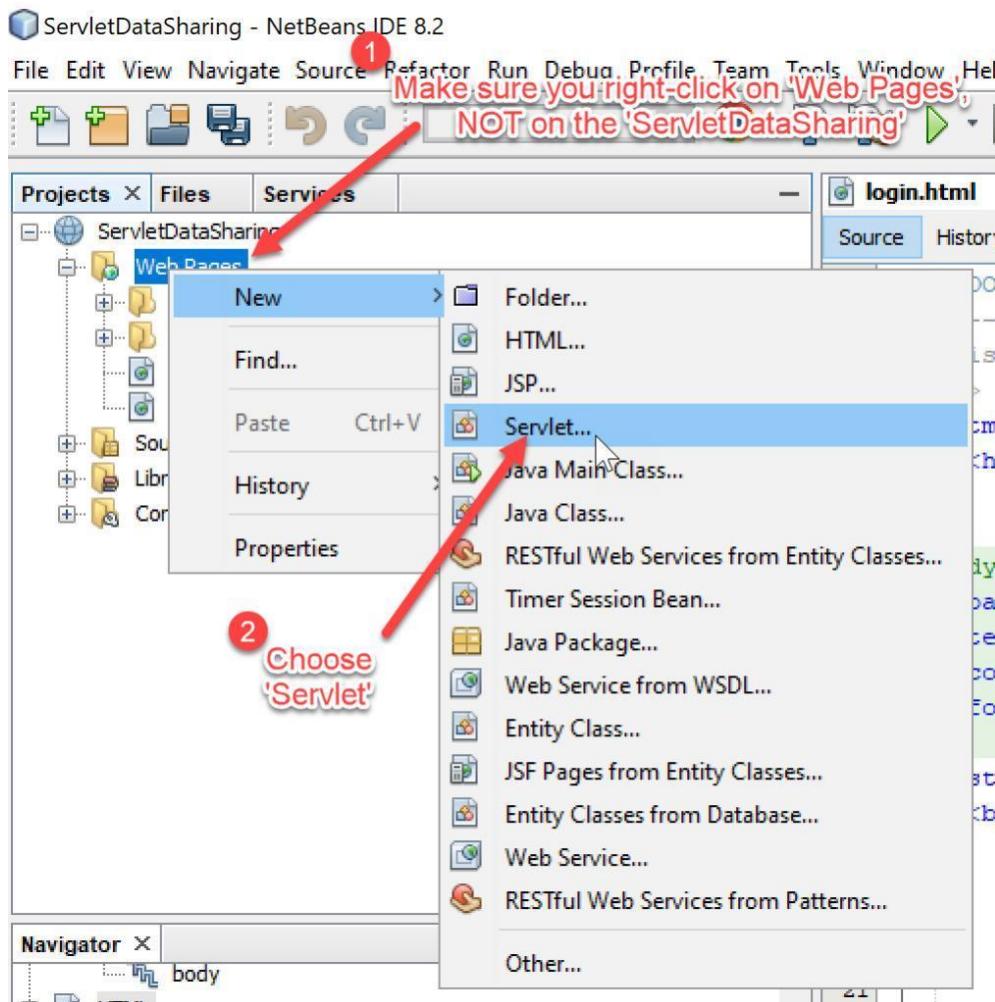
10. Name your HTML file as ‘login’, then click Finish.

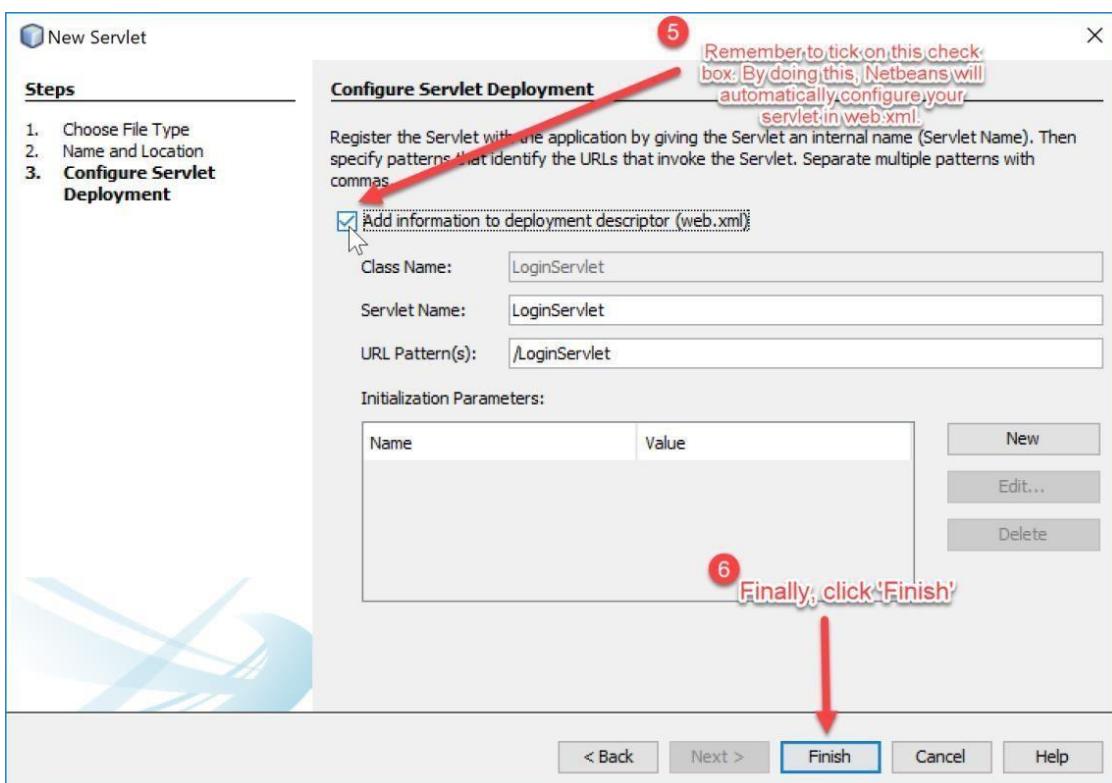
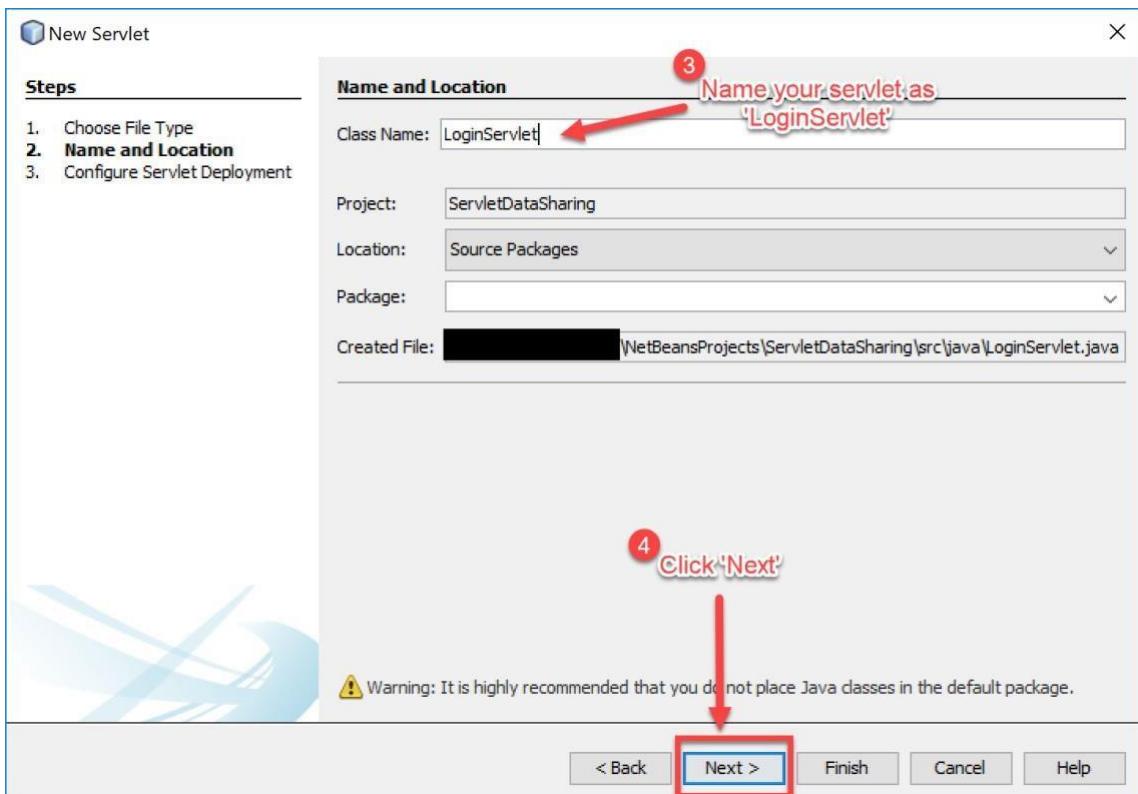


11. Type the following HTML markups into your login.html.

```
<!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:



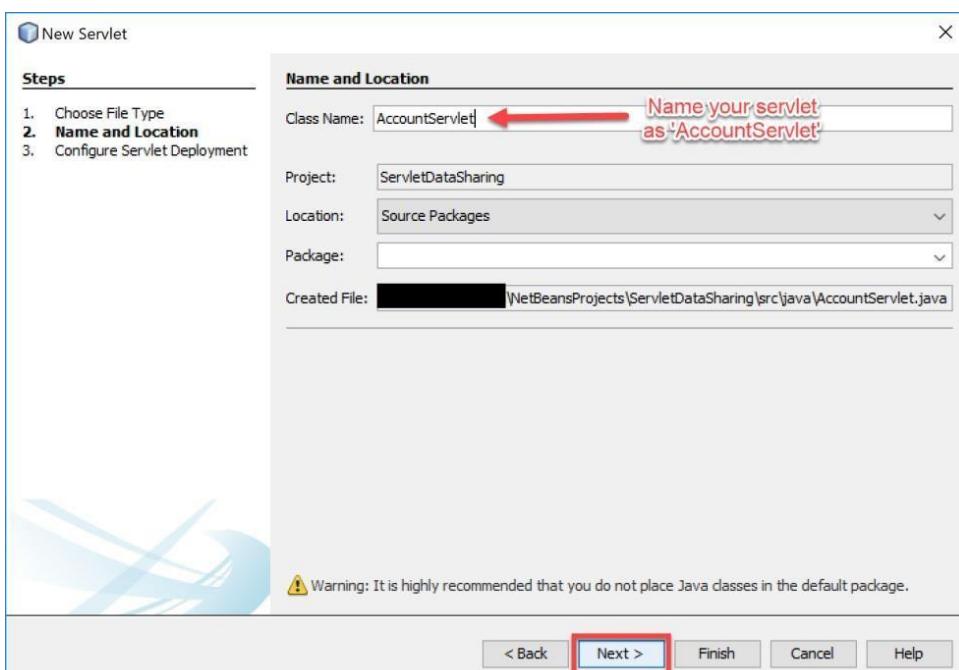


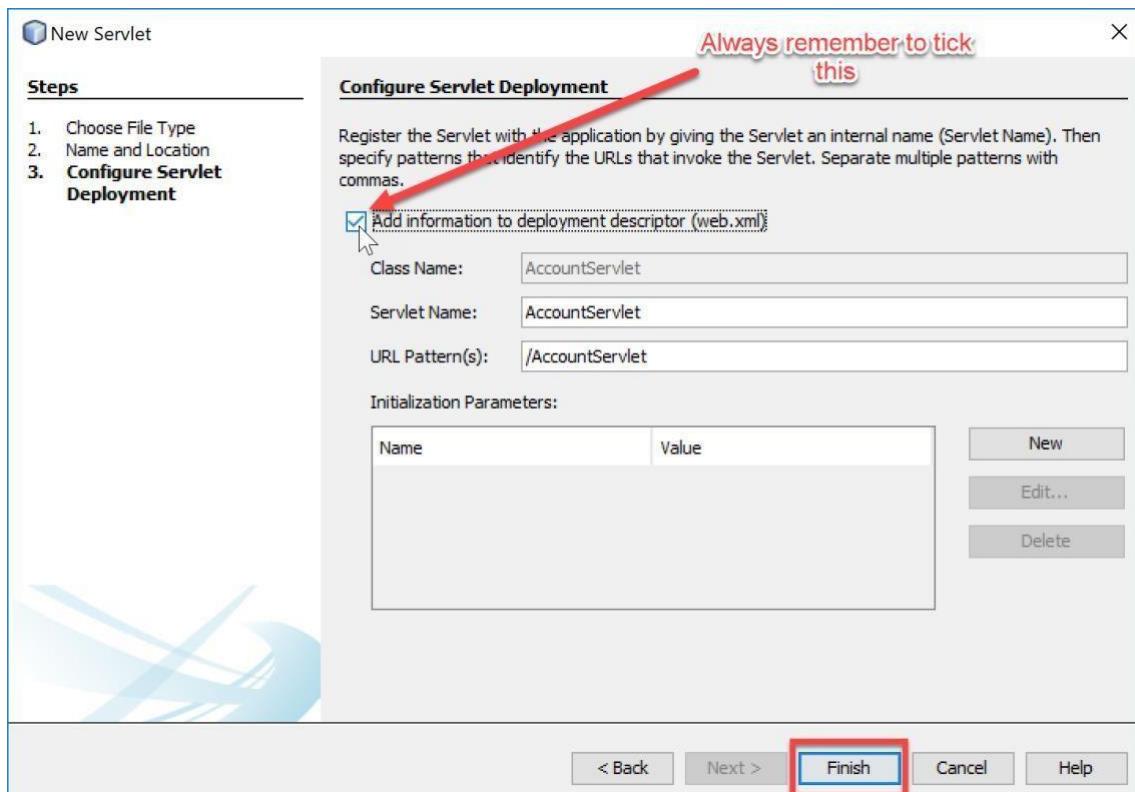
```

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

```

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.





The screenshot shows the 'AccountServlet.java - Editor' window. The code editor displays the following Java code:

```
1  ... 5 lines
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;
```

A red callout bubble points to the 'import java.util.HashMap;' line, which is highlighted with a red box. The text inside the bubble reads: 'Once again, we need to import HashMap class. This time we use this to store the information about the user accounts.'

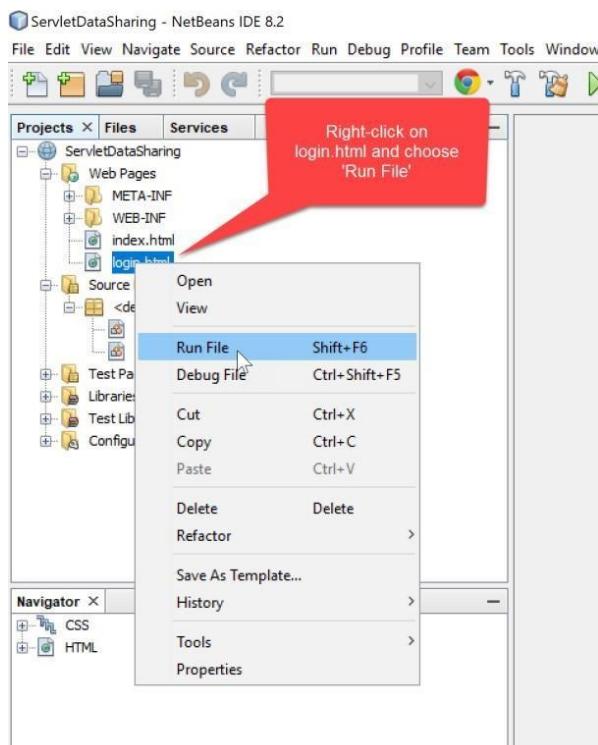
AccountServlet.java - Editor

AccountServlet.java

Source History

```
15  /*...4 lines */
16  public class AccountServlet extends HttpServlet {
17      HashMap<String, String[]> account = new HashMap();
18
19      @Override
20      public void init() throws ServletException{
21          super.init();
22          account.put("Ali", new String[]{"31/01/2019: 2000.00", "28/02/2019: 3000.00"});
23          account.put("Ahmad", new String[]{"31/01/2019: 100.00", "28/02/2019 5000.00"});
24          account.put("Muthu", new String[]{"31/01/2019: 1000", "28/02/2019 2000"});
25      }
26
27  }
28
29  /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
30  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31      throws ServletException, IOException {
32      response.setContentType("text/html;charset=UTF-8");
33
34      String userid_login = (String)request.getAttribute("userid");
35
36      try (PrintWriter out = response.getWriter()) {
37
38          out.println("<!DOCTYPE html>");
39          out.println("<html>");
40          out.println("<head>");
41          out.println("<title>Servlet AccountServlet</title>");
42          out.println("</head>");
43          out.println("<body>");
44
45          if(account.get(userid_login)==null){
46              out.println("<h1>Sorry, no information found!</h1>");
47          }
48          else{
49              out.println("<h1>Account status for: " + userid_login + "</h1>");
50              for(String tempAcc: account.get(userid_login)){
51                  out.println("<h2>" +tempAcc+"</h2>");
52              }
53          }
54
55          out.println("</body>");
56          out.println("</html>");
57      }
58  }
59
60  }
61
62  }
63
64  }
65
66 }
```

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 login form with a black background. At the top, it says "Welcome to CSF3107". Below that, it says "Please insert your username and password". There are two input fields: "Username: Ahmad" and "Password:". 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.

The screenshot shows account status information for a user named "Ahmad". It displays two dates with their corresponding account balances: "31/01/2019: 100.00" and "28/02/2019 5000.00".

Reflections:

1. What have you learnt from this exercise?
 - Servlets have a initialization (`init()`), handling requests (`doGet()`, `doPost()`), and finally being destroyed (`destroy()`). This knowledge helps in managing resources and performing necessary actions at each stage of the servlet's life.
2. What are the common methods used in Java Servlet?
 - `void init():` This method is called by the servlet container to initialize the servlet.
 - `void doGet():` This method is invoked by the servlet container to handle HTTP GET

requests.

- `void doPost():` This method is invoked by the servlet container to handle HTTP POST requests.
- `void destroy():` This method is called by the servlet container to indicate that the servlet is being taken out of service.

OUTPUT

Login Page

localhost:8080/Servlet... A ⌂ ⌂

Welcome to CSF3107

Please insert your username and password

Username:

Password:

Login Reset

Servlet AccountServlet

localhost:8080/Servlet... A ⌂ ⌂ ⌂

Account status for: Ali

31/01/2019: 2000.00

28/02/2019: 3000.00

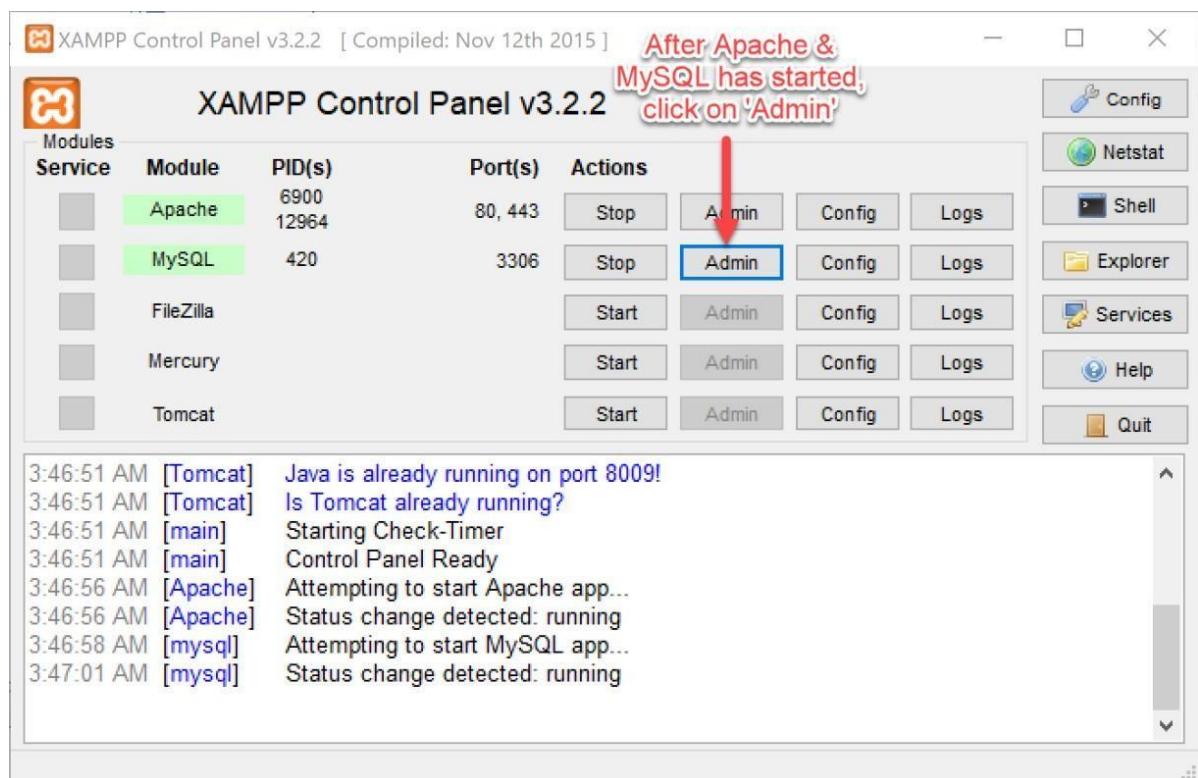
Task 2: Creating A Table in MySQL Database

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.

localhost/phpmyadmin/server_databases.php?server=1

phpMyAdmin

Server: 127.0.0.1

Databases SQL Status User accounts Export

New 1 Click on 'New' to create a new Database

amstmg
information_schema
mysql
performance_schema
phpmyadmin
test
yii2advanced

Create database 2 Name your database as 'CSF3107'

CSF3107 latin1_swedish_ci Create

Database	Collation	Action
amstmg	utf8_general_ci	<input type="button" value="Check privileges"/>
information_schema	utf8_general_ci	<input type="button" value="Check privileges"/>
mysql	latin1_swedish_ci	<input type="button" value="Check privileges"/>
performance_schema	utf8_general_ci	<input type="button" value="Check privileges"/>
phpmyadmin	utf8_bin	<input type="button" value="Check privileges"/>
test	latin1_swedish_ci	<input type="button" value="Check privileges"/>
yii2advanced	latin1_swedish_ci	<input type="button" value="Check privileges"/>
Total: 7		

Check all With selected: Drop

Note: Enabling the database statistics here might cause heavy traffic between the w

- Enable statistics

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

localhost/phpmyadmin/db_structure.php?server=1&db=csf3107

phpMyAdmin

Server: 127.0.0.1 » Database: csf3107

Structure SQL Search Query Export Import

New 1 Click on your database link, CSF3107

amstmg
information_schema
mysql
performance_schema
phpmyadmin
test
yii2advanced

No tables found in database.

Create table 2 Create a table, name it as 'users'

Name: users Number of columns: 4 3 Click 'Go'

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.

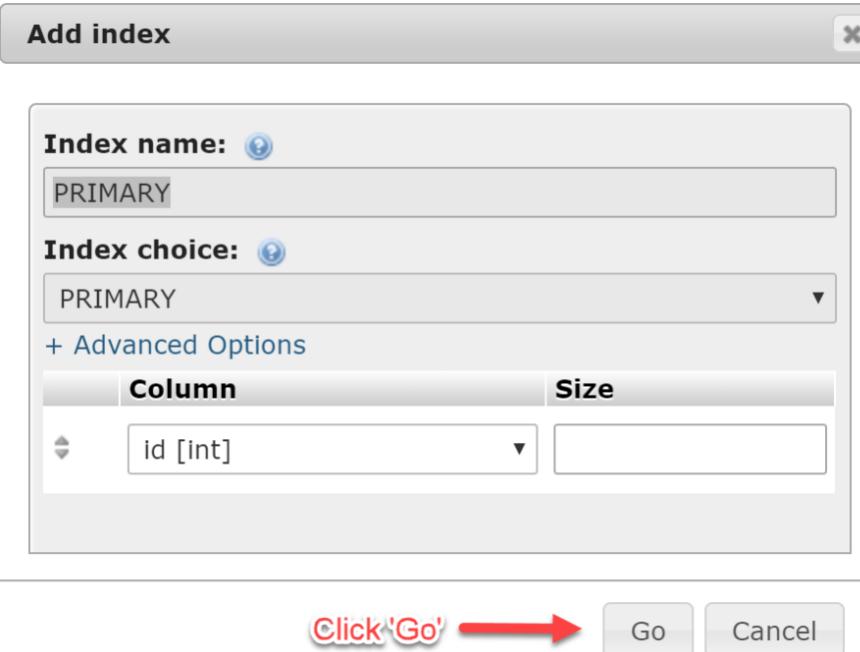
Click on A_L to enable autoincrement for field 'id'

Click on A_L to enable autoincrement for field 'id'

Click Save

Preview SQL Save

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



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

The screenshot shows the phpMyAdmin interface. On the left, the database tree shows 'csf3107' selected, with 'New', 'amstmg', and 'users' as children. The 'users' node has 'Columns' and 'Indexes' as sub-nodes. A red arrow labeled '1 Click on 'users' table' points to the 'users' node. On the right, the main panel shows the 'Structure' tab for the 'users' table. The table has four columns: id, username, password, and roles. A red box highlights the table structure. A red arrow labeled '2 You will see an empty table' points to the table area. Below the table, there is a 'Query results operations' section with a 'Create view' button. At the bottom, there is a 'Bookmark this SQL query' section with a 'Label:' input field and a checkbox for 'Let every user access this bookmark'.

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

The screenshot shows the phpMyAdmin interface for inserting data into the 'users' table. On the left, the database tree is the same as the previous screenshot. The main panel shows the 'Insert' tab for the 'users' table. There are two data entry forms. The top form has columns 'id', 'username' (value 'Ali'), 'password' (value '1234'), and 'roles' (value 'admin'). The bottom form has columns 'id', 'username' (value 'Ahmad'), 'password' (value '4567'), and 'roles' (value 'user'). Both forms have a 'Null' column and a 'Value' column. A red box highlights the second data entry form. A red arrow labeled '1 Click on 'Insert' link to insert a new data' points to the 'Insert' link in the top navigation bar. A red arrow labeled '2 Untick the 'Ignore' checkbox' points to the 'Ignore' checkbox in the first form. A red arrow labeled '3 Fill in the details as shown' points to the second form. A red arrow labeled '4 Click 'Go' to save the record' points to the 'Go' button at the bottom right of the second form. At the very bottom, there is a message 'Continue insertion with 2 18 rows'.

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`

Data in the 'users' table

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 Edit Copy Delete Export

OUTPUT

The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
1 • SELECT * FROM csm3023.users;
```

Results Grid:

	id	username	password	roles
▶	1	Ali	1234	Admin
	2	Ahmad	4567	User
*	NULL	NULL	NULL	NULL

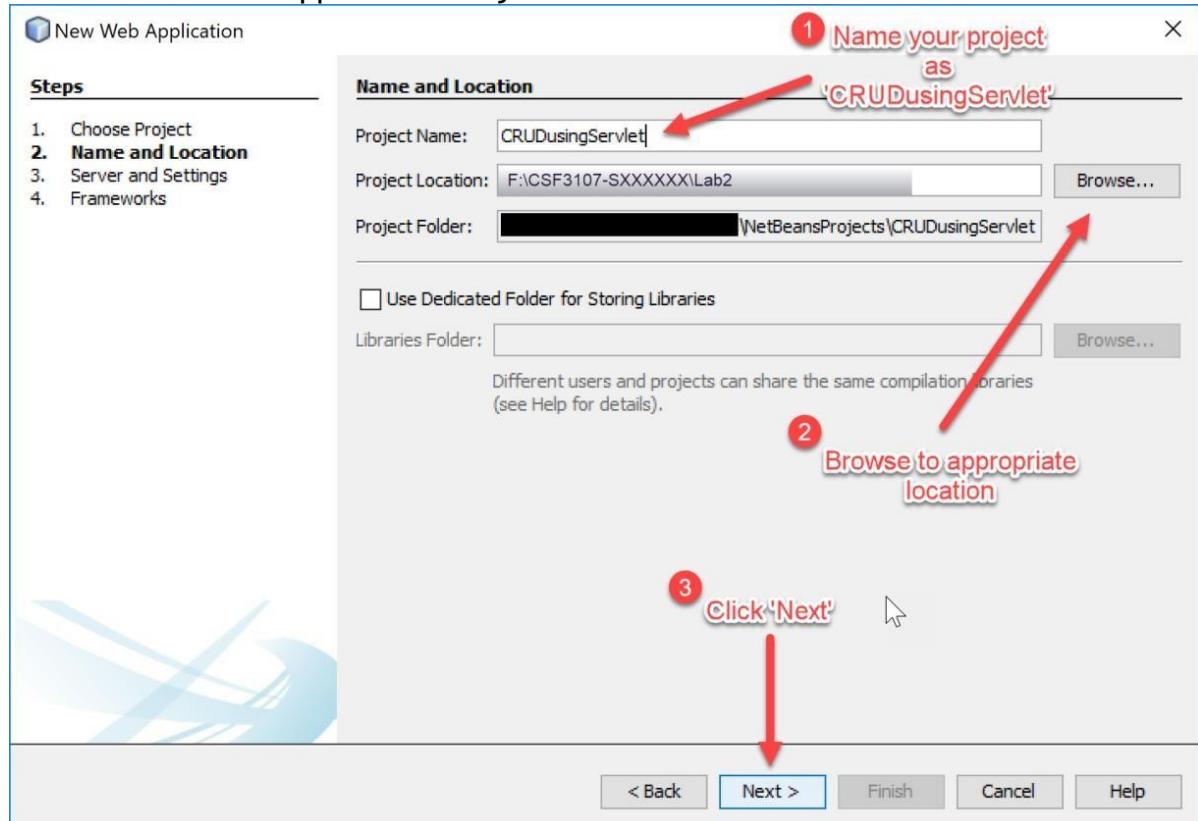
Task 3: Setting the Environment of Web Application for Database Connection

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

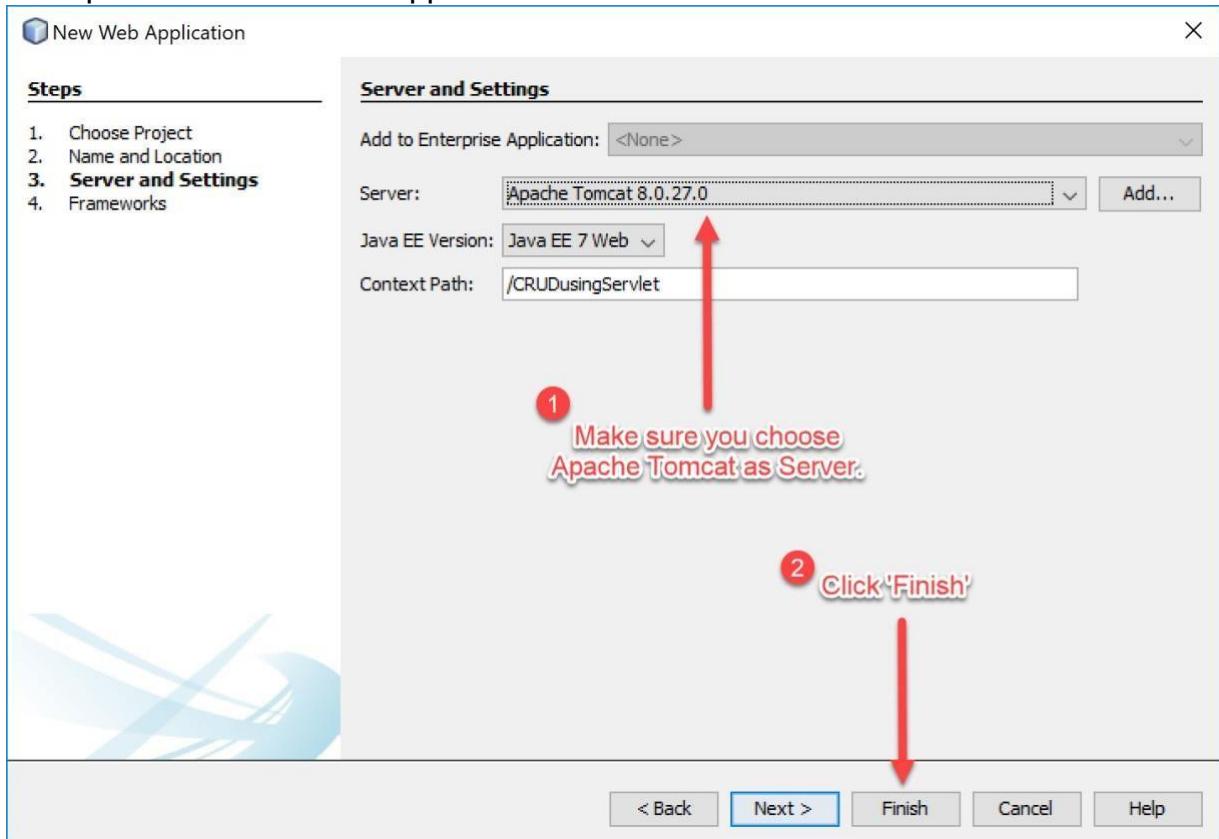
Problem Description: Import MySQL JDBC Library to an existing project

Estimated time: 5 minutes

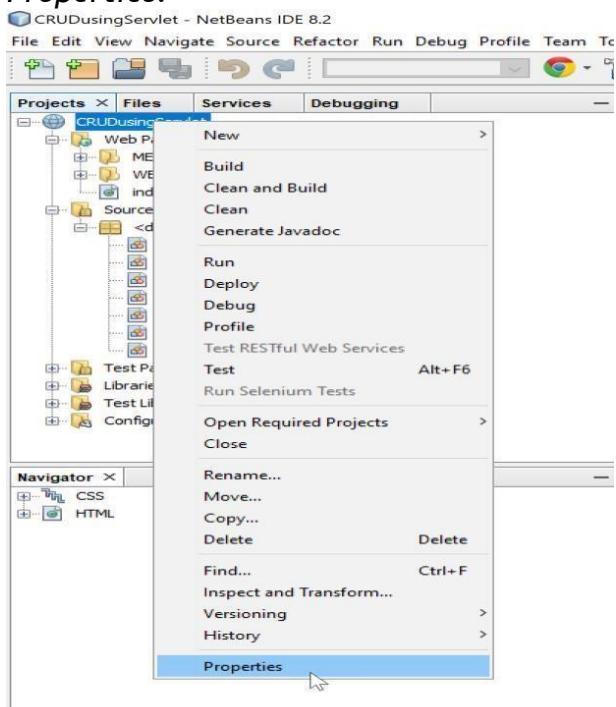
1. Create a new Web Application Project.



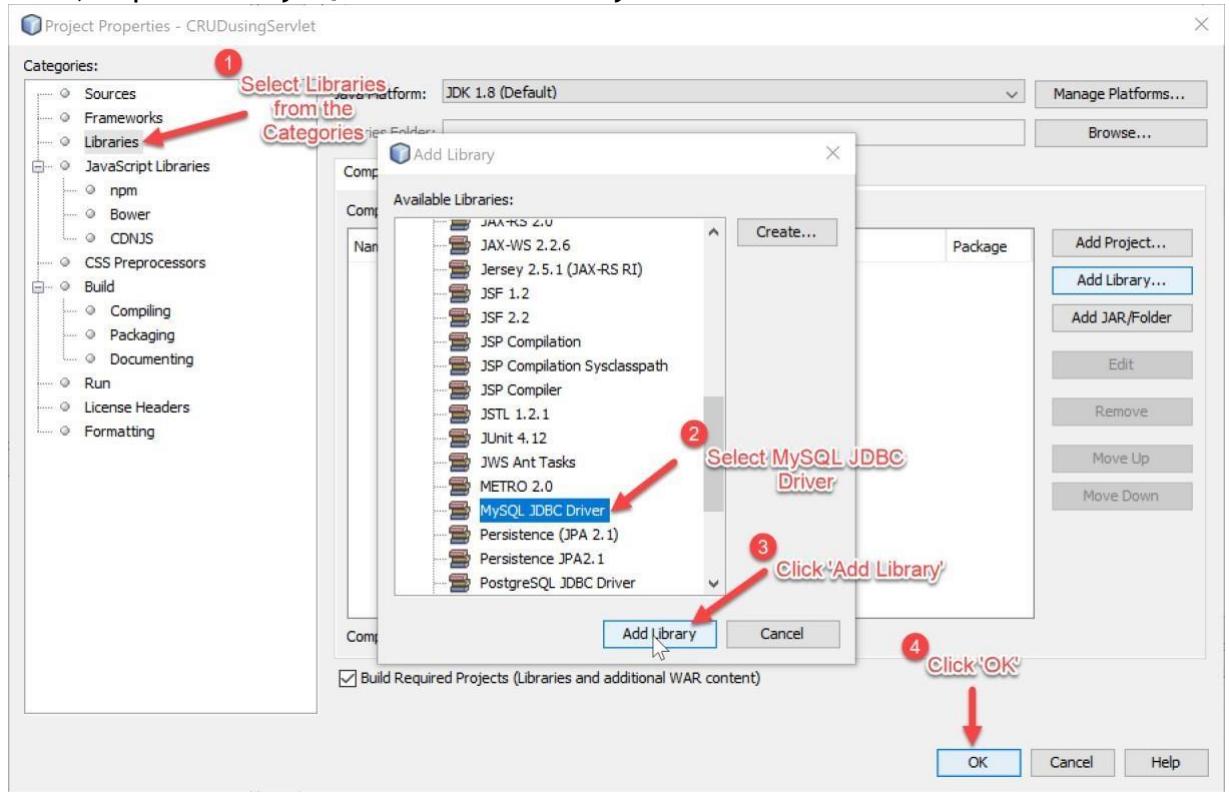
2. Use Apache Tomcat as our application server.



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

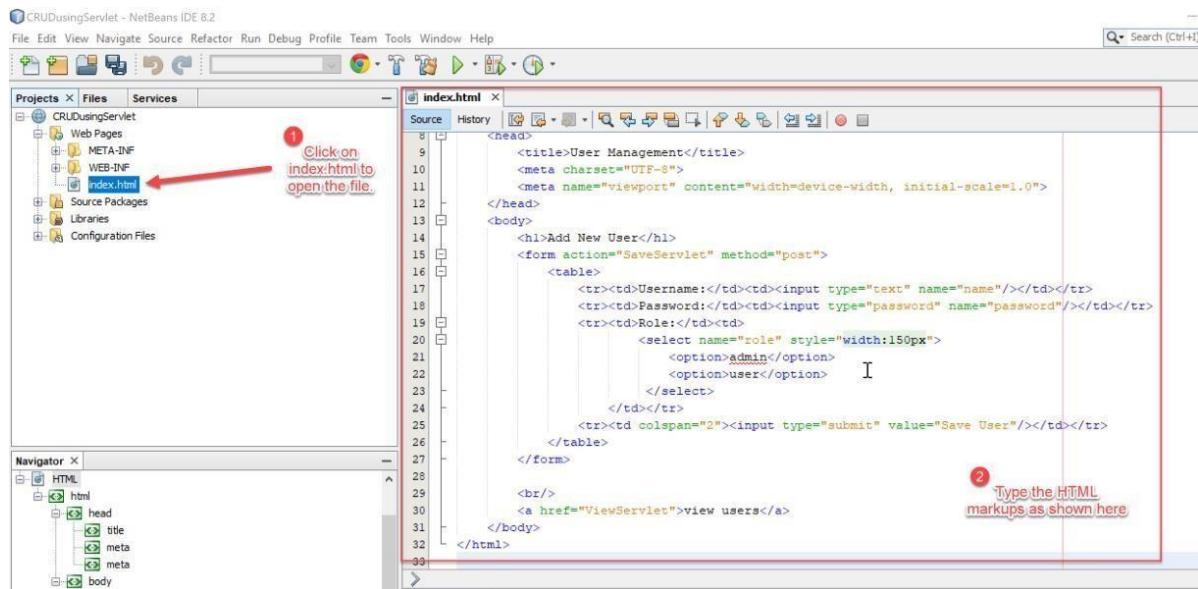
Problem Description: Program five different servlets to handle database operations such as insert, update and delete.

- 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

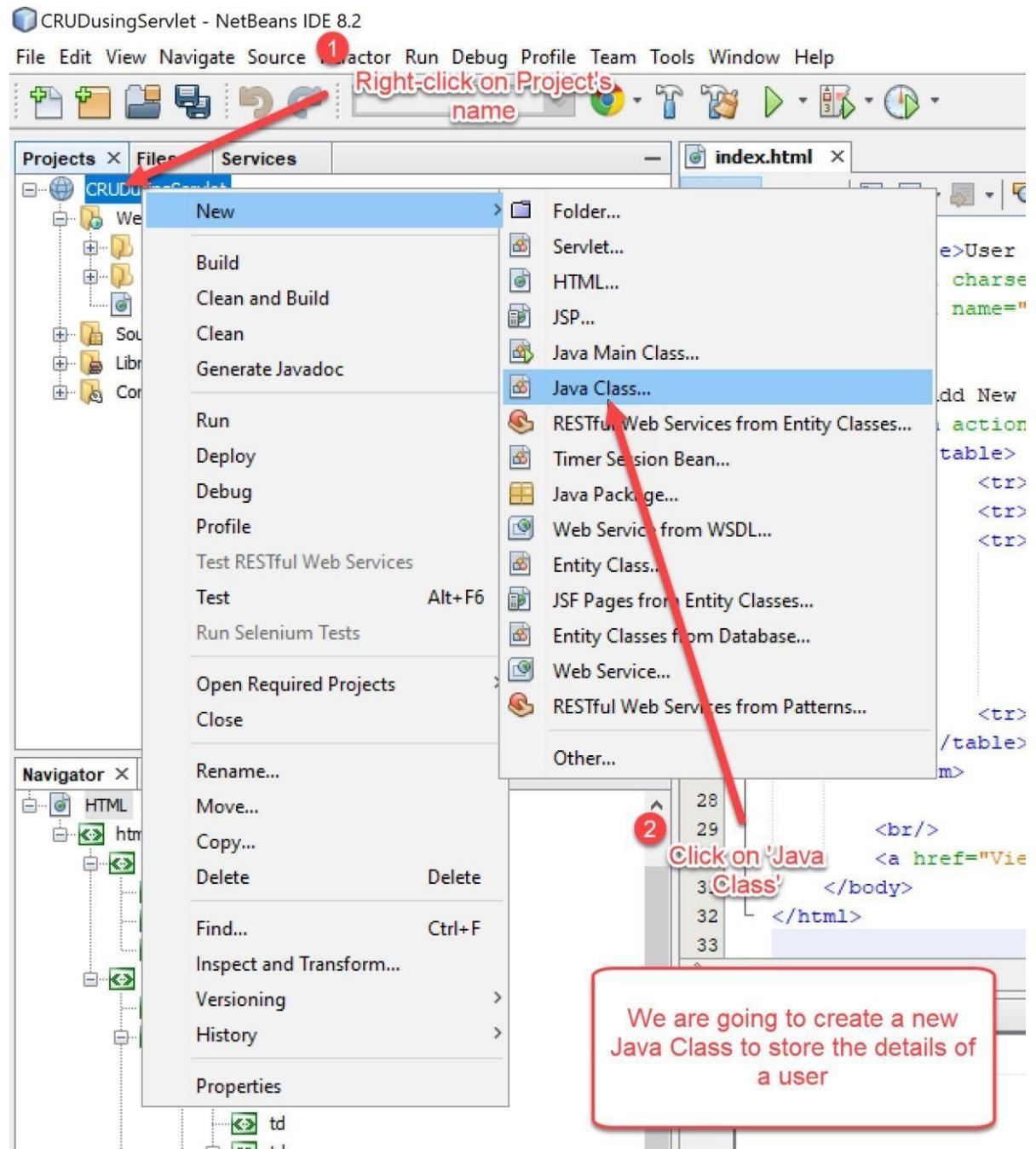
Apart from the servlets, we are going to develop two custom Java class known as JavaBeans and Data Access Object (DAO).

Estimated time: 90 minutes

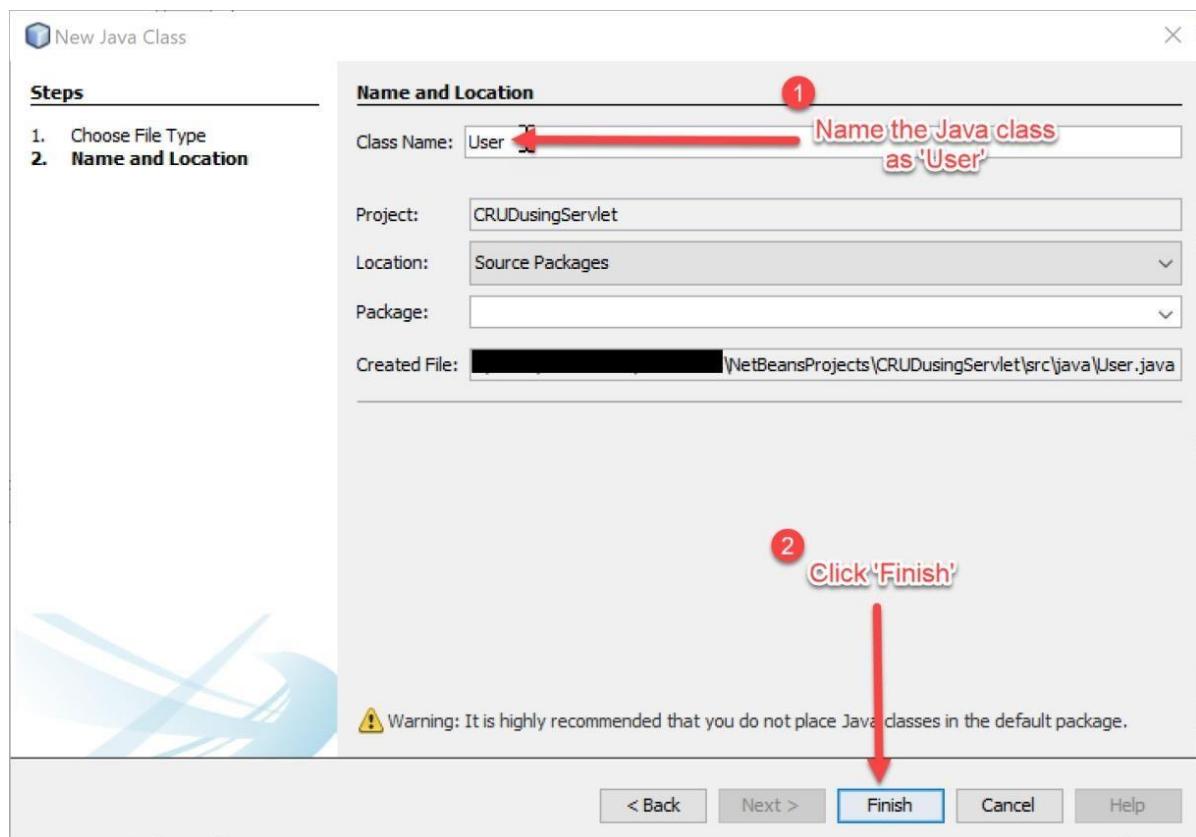
1. Open the Web Application Project (*CRUDusingServlet*) created in Task 3.
2. Open the index.html file and edit as follows.



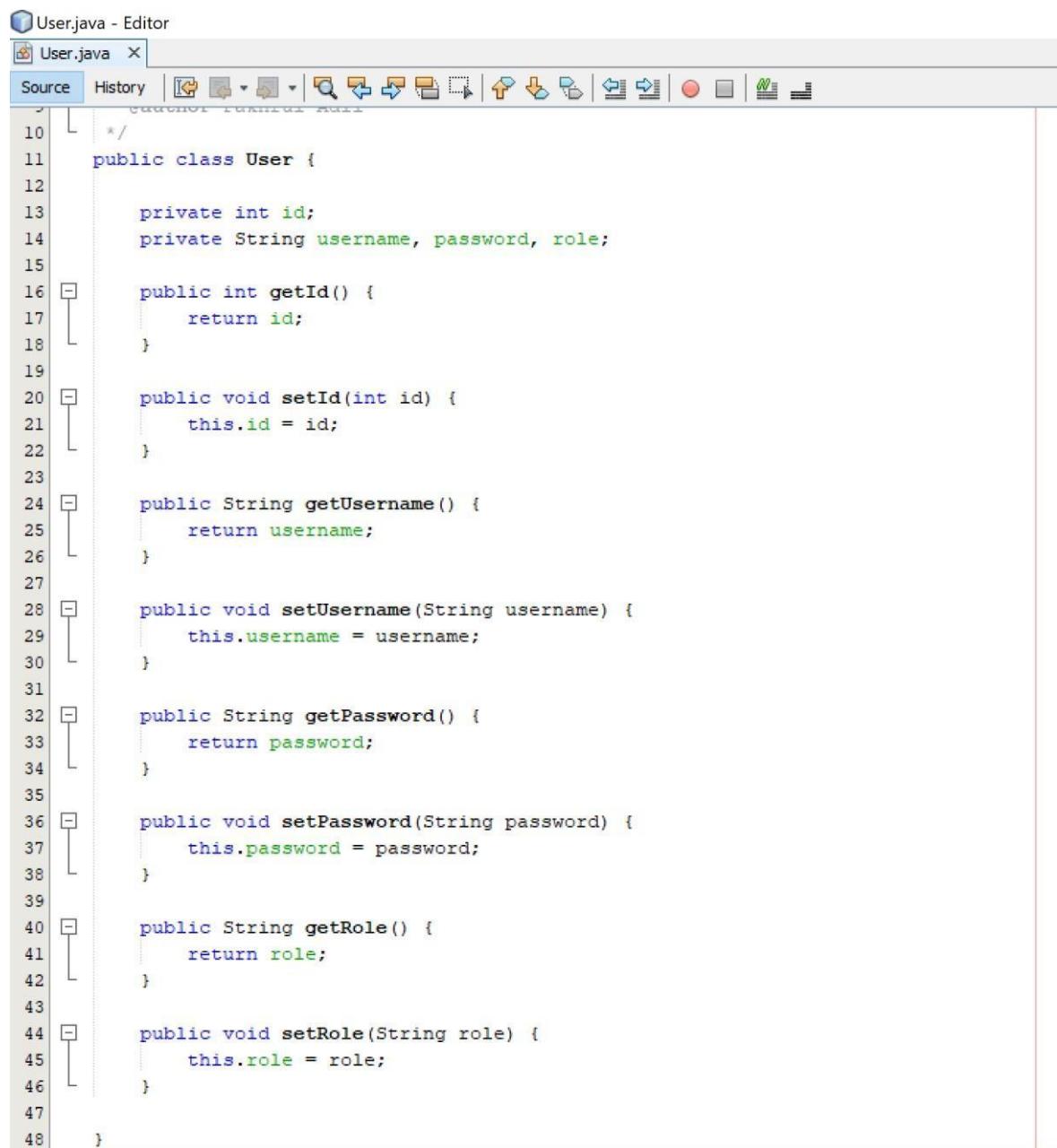
3. Now we are going to create a Java class which is specially used to store the user data.



4. Name the Java class as *User*.



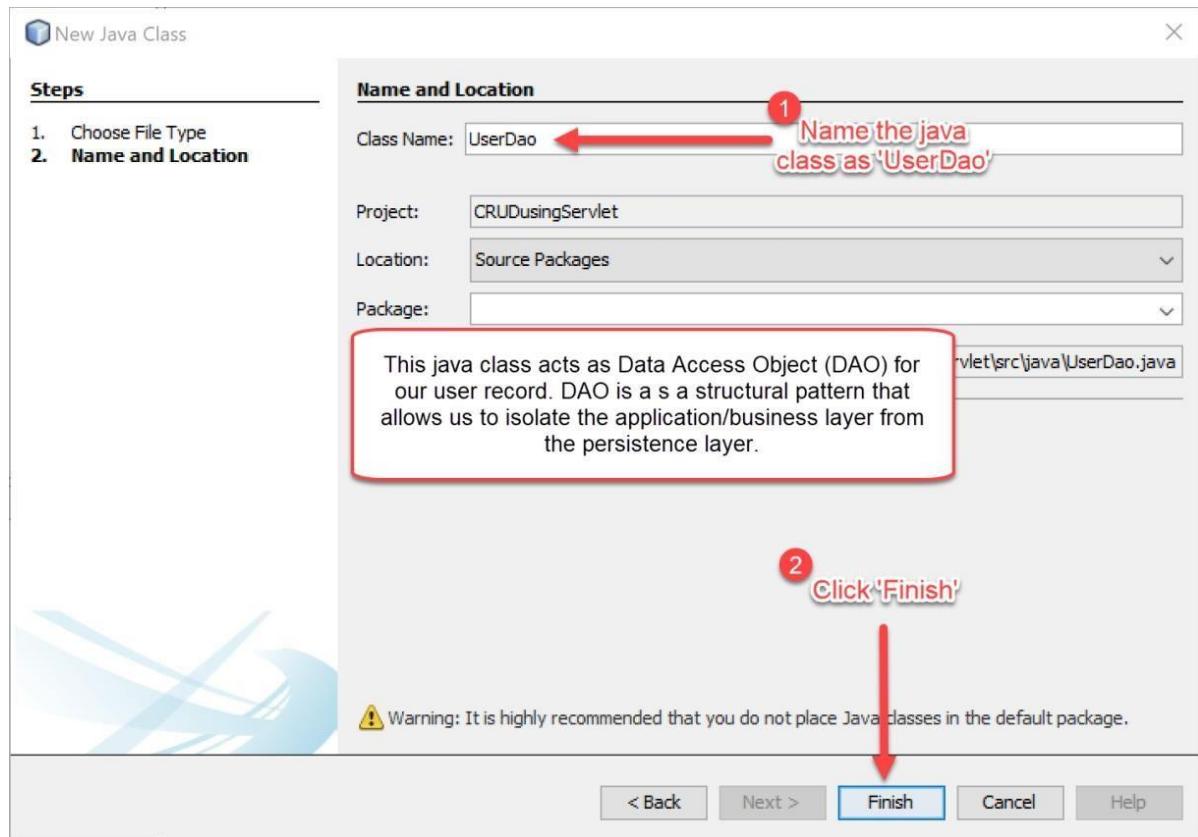
5. Type the following codes in User.java file.



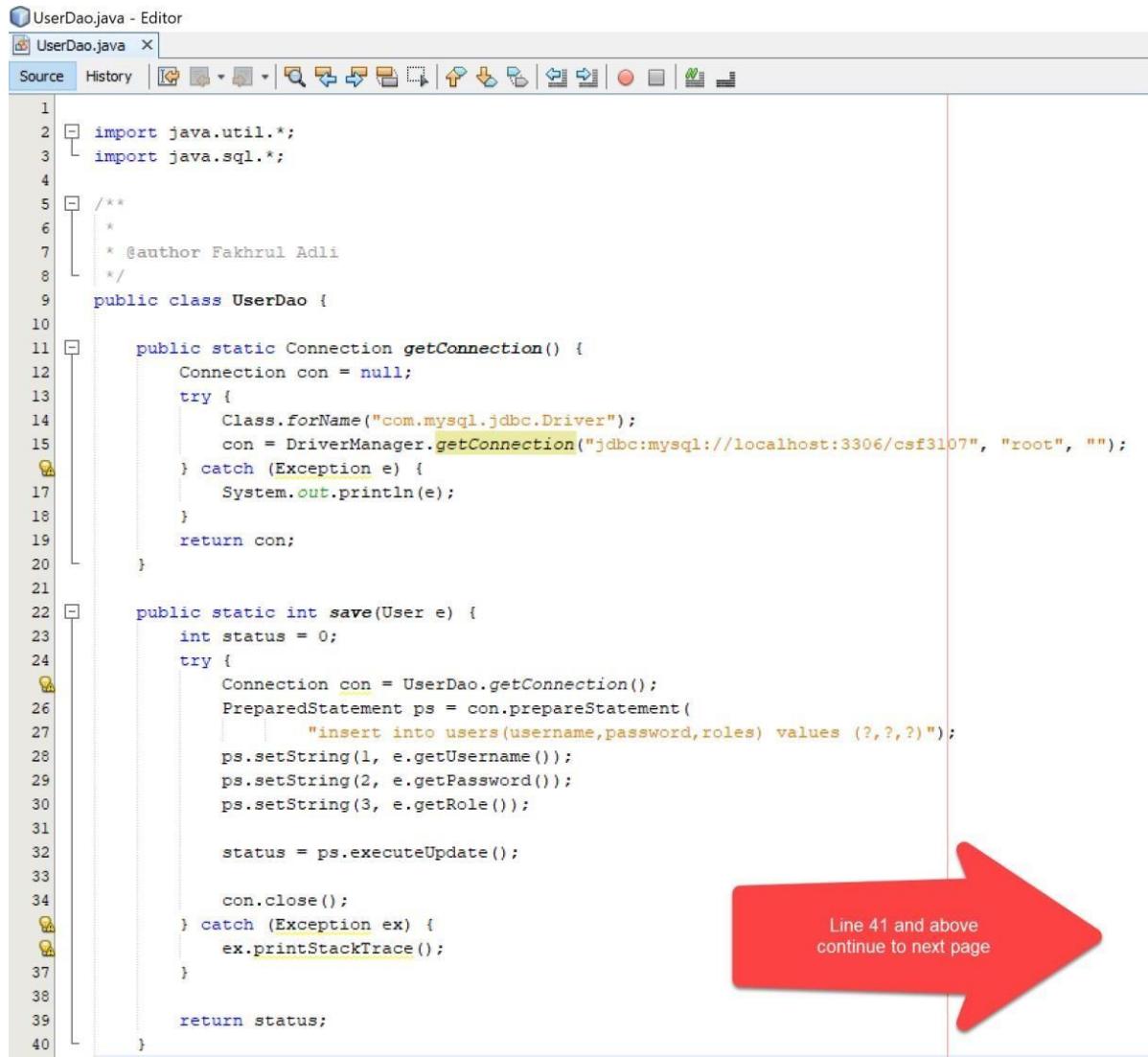
The screenshot shows a Java code editor window titled "User.java - Editor". The tab bar at the top has "User.java" selected. The toolbar below the tabs includes icons for cut, copy, paste, find, and save. The main pane displays the following Java code:

```
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      }
49  }
```

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.



UserDao.java - Editor

UserDao.java

Source History

```
1 import java.util.*;
2 import java.sql.*;
3
4 /**
5  * 
6  * @author Fakhru Adli
7  */
8
9 public class UserDao {
10
11     public static Connection getConnection() {
12         Connection con = null;
13         try {
14             Class.forName("com.mysql.jdbc.Driver");
15             con = DriverManager.getConnection("jdbc:mysql://localhost:3306/csf3107", "root", "");
16         } catch (Exception e) {
17             System.out.println(e);
18         }
19         return con;
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.

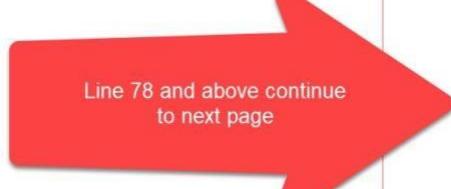
UserDao.java - Editor

UserDao.java

Source History

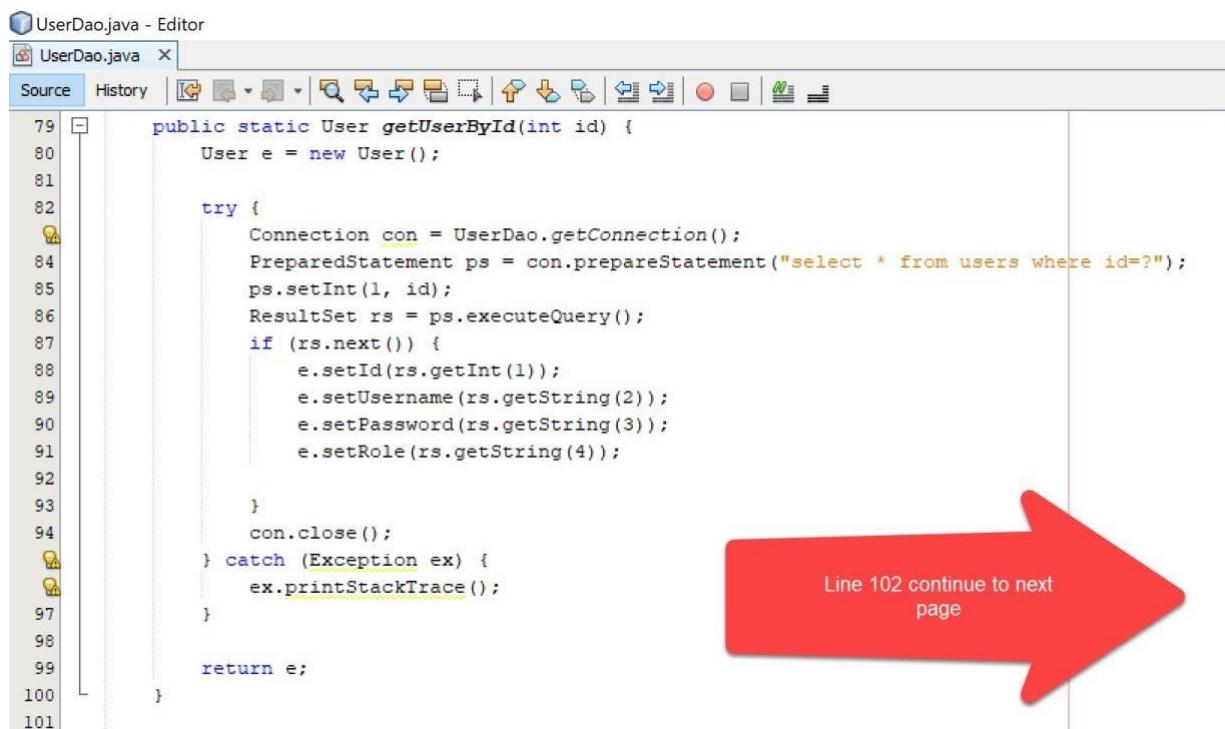
```
42     public static int update(User e) {
43         int status = 0;
44         try {
45             Connection con = UserDao.getConnection();
46             PreparedStatement ps = con.prepareStatement(
47                 "update users set username=?,password=?,roles=? where id=?");
48             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    }
```

Line 102 continue to next page

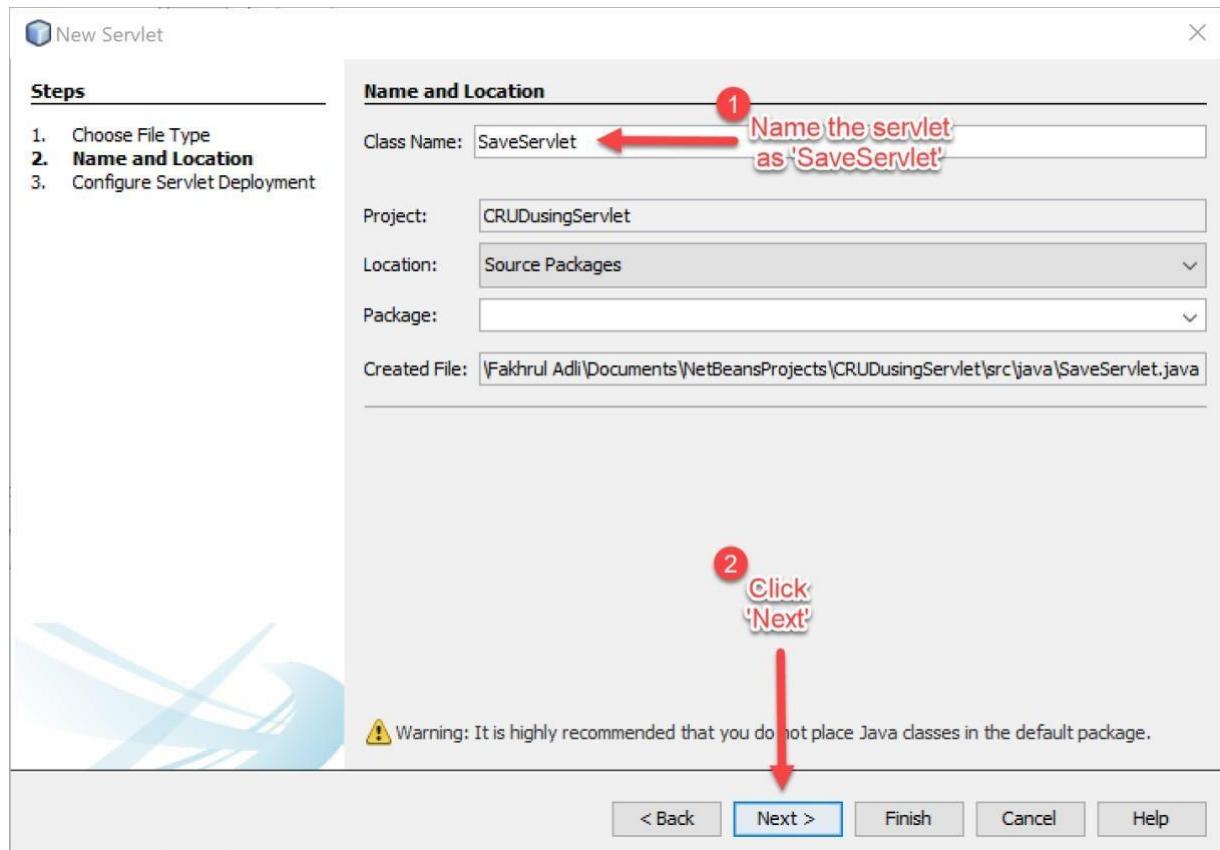
10. Save (CTRL+s) UserDao.java file.



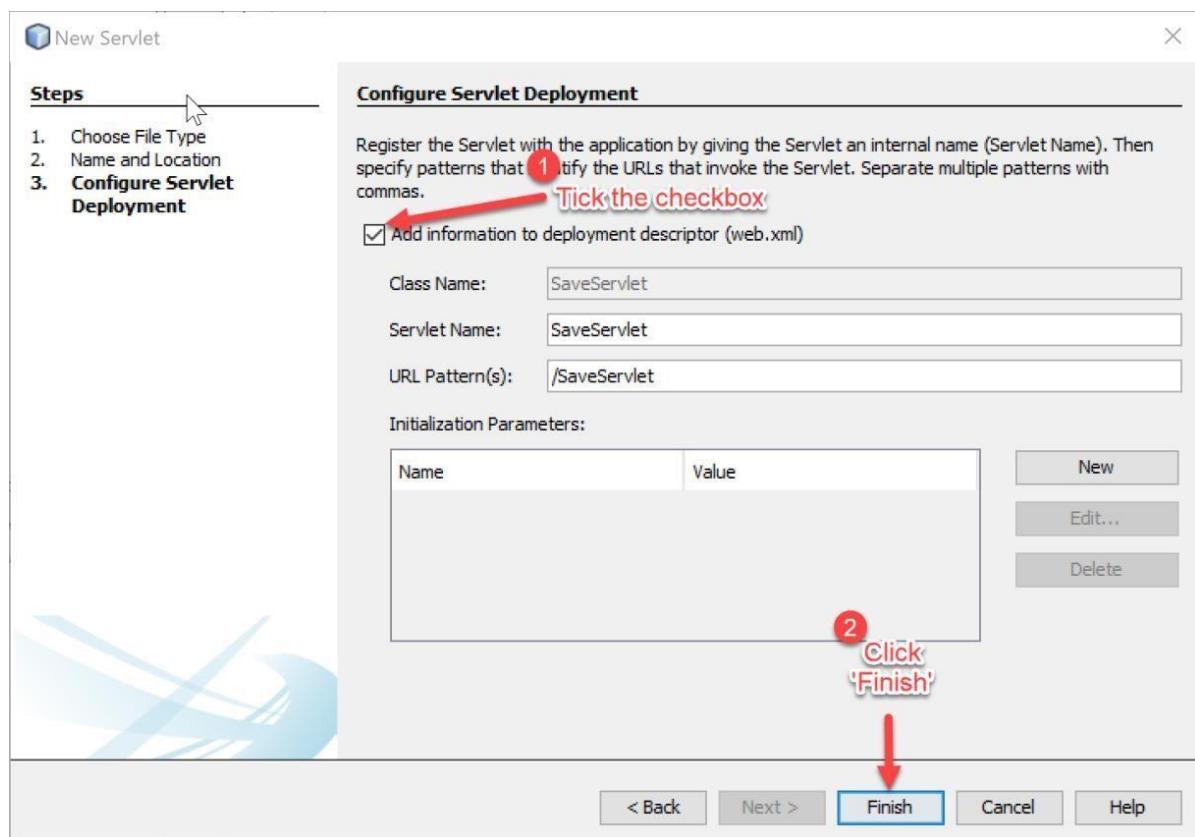
```
101
102     public static List<User> getAllUsers() {
103         List<User> list = new ArrayList<User>();
104
105         try {
106             Connection con = UserDao.getConnection();
107             PreparedStatement ps = con.prepareStatement("select * from users");
108             ResultSet rs = 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

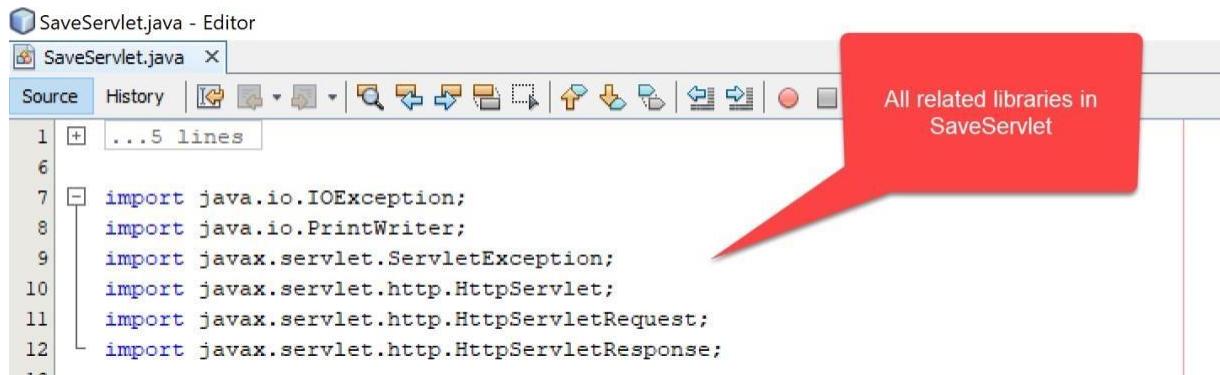
11. After finish coding the UserDao.java, now we are to code the servlet for saving data into the database.



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

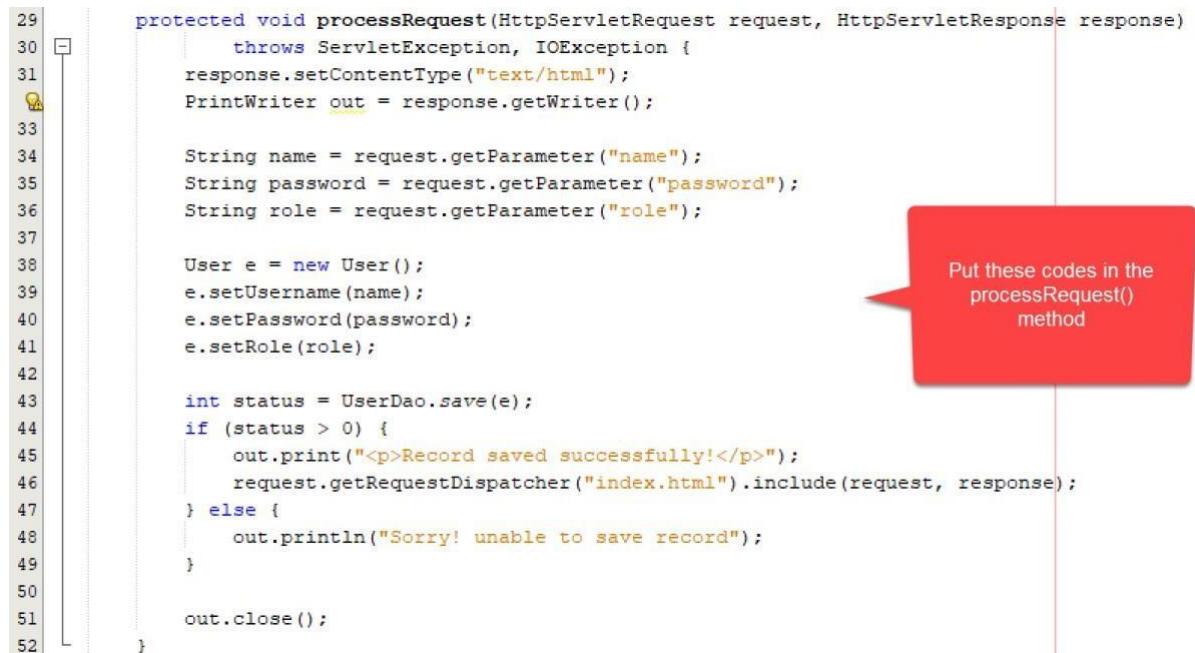


13. Make sure all related libraries are included in the SaveServlet.java file.



```
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;
```

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();
}
```

15. Save the file after finished.

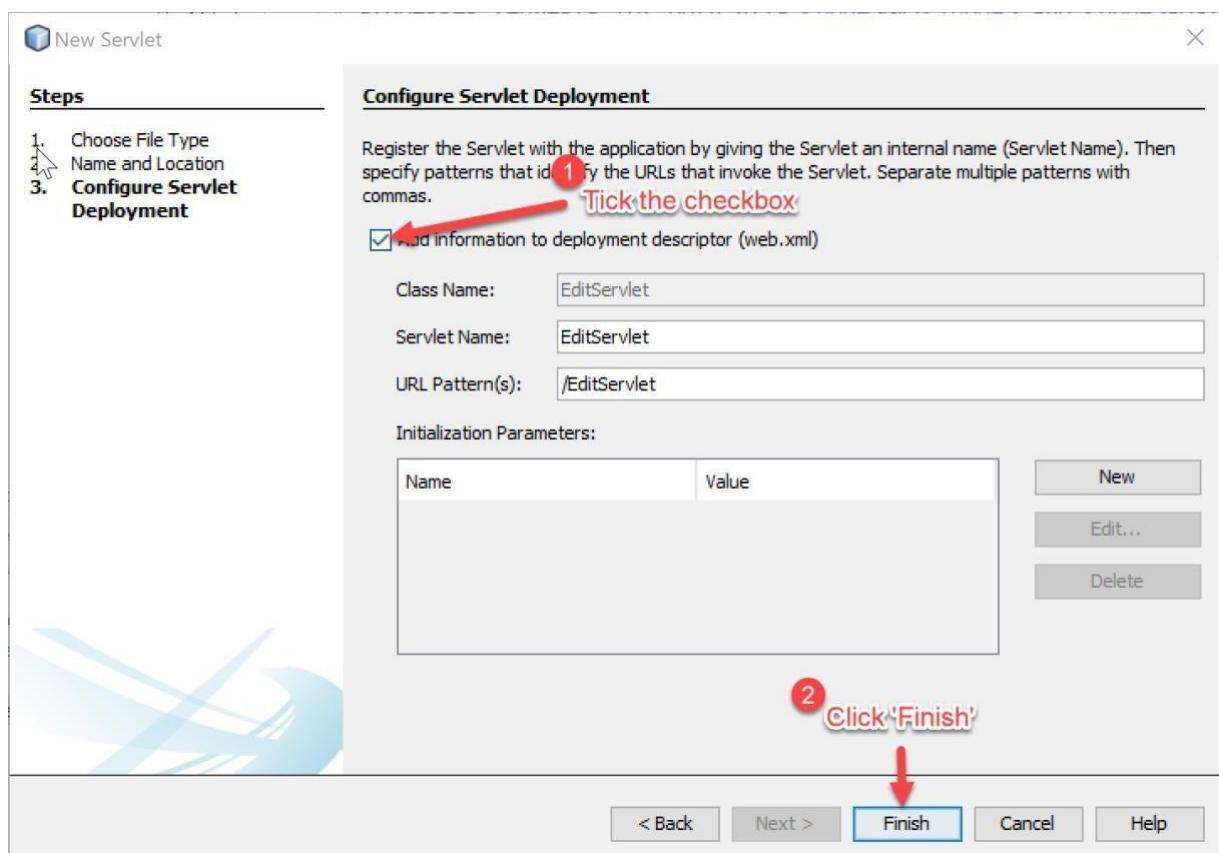
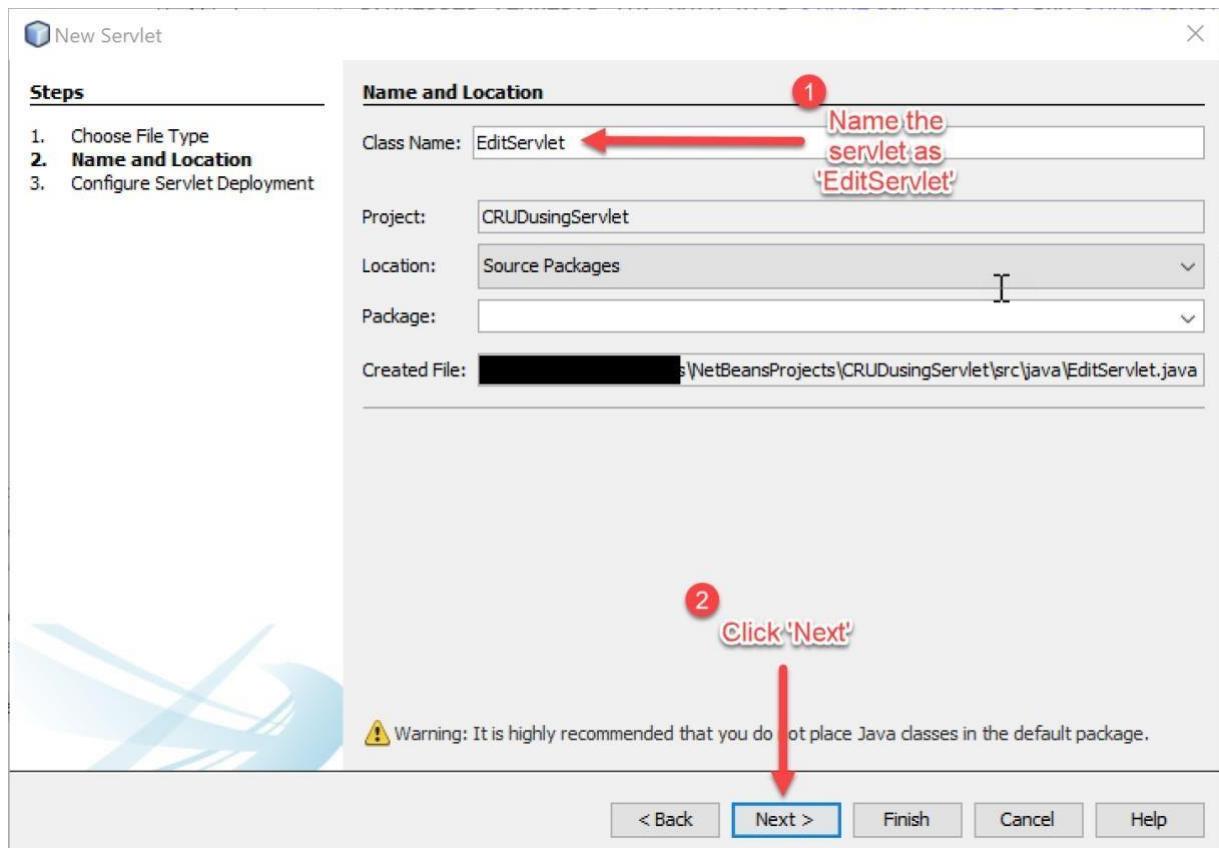
16. Next, we are going to create a servlet to view our data in database. Create a new servlet *ViewServlet*, tick ‘Add information to deployment descriptor (web.xml)’ and click finish. Insert the following codes to the servlet.

```

ViewServlet.java - Editor
ViewServlet.java X
Source History | Import all related libraries
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;
7
8 import java.util.List;
9
10 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
11     throws ServletException, IOException {
12     response.setContentType("text/html");
13     PrintWriter out=response.getWriter();
14     out.println("<a href='index.html'>Add New User</a>");
15     out.println("<h1>User List</h1>");
16
17     List<User> list=UserDao.getAllUsers();
18
19     out.print("<table border='1' width='100%'>");
20     out.print("<tr><th>Id</th><th>Name</th><th>Password</th><th>Role</th>" +
21             "<th>Edit</th><th>Delete</th></tr>");
22     for(User e:list){
23         out.print("<tr><td>" +e.getId() + "</td><td>" +e.getUsername() + "</td><td>" +
24                 +e.getPassword() + "</td><td>" +e.getRole() + "</td><td><a href='EditServlet?id=" +
25                 +e.getId() + "'>edit</a></td> <td><a href='DeleteServlet?id=" +
26                 +e.getId() + "'>delete</a></td></tr>");
27     }
28     out.print("</table>");
29     out.close();
30 }
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

```

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

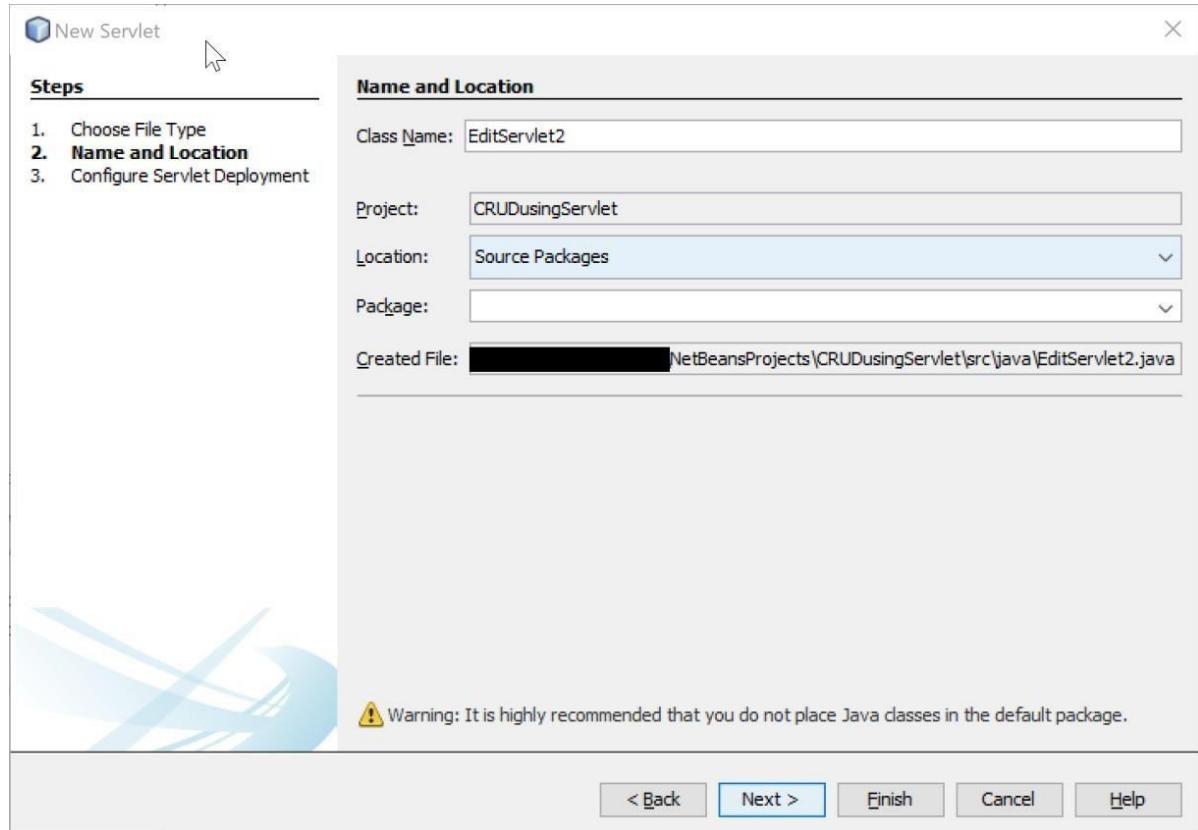
Update User

Name:

Password:

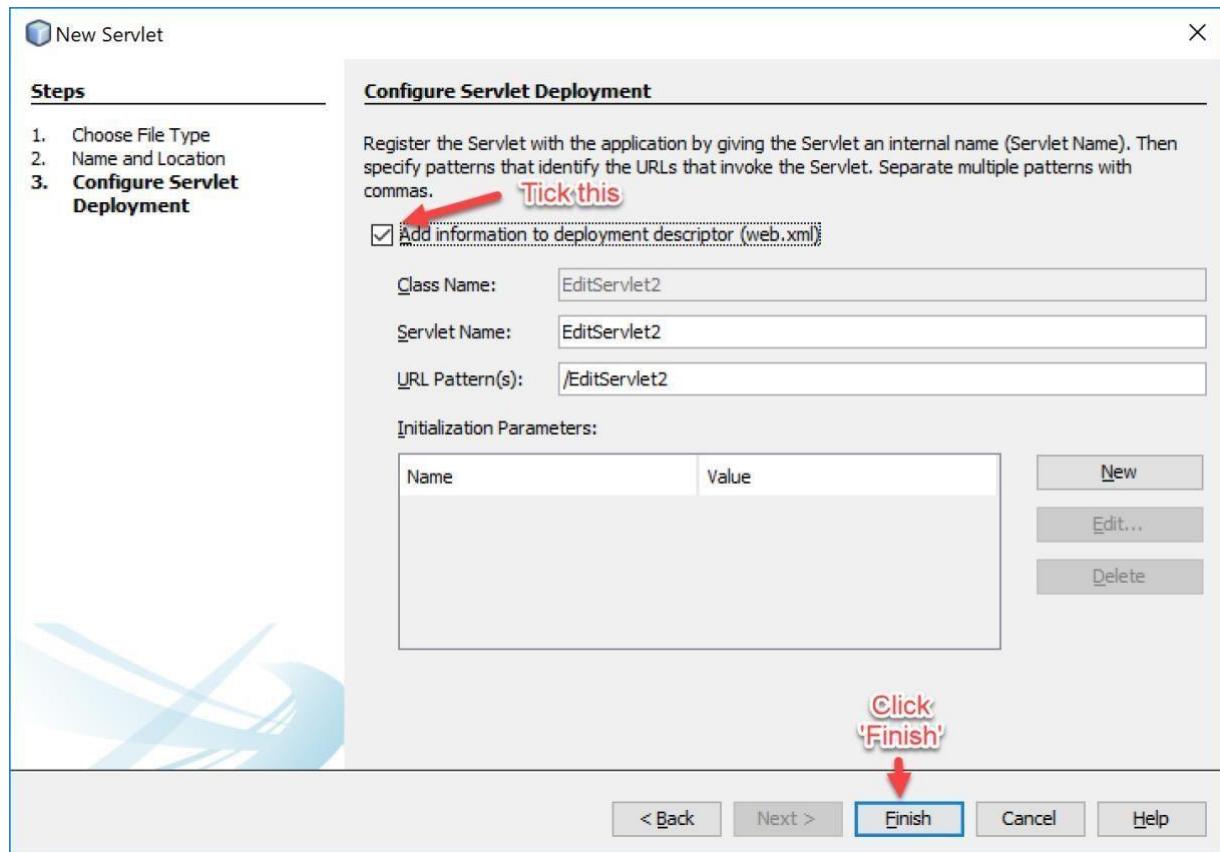
Role: ▾

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.



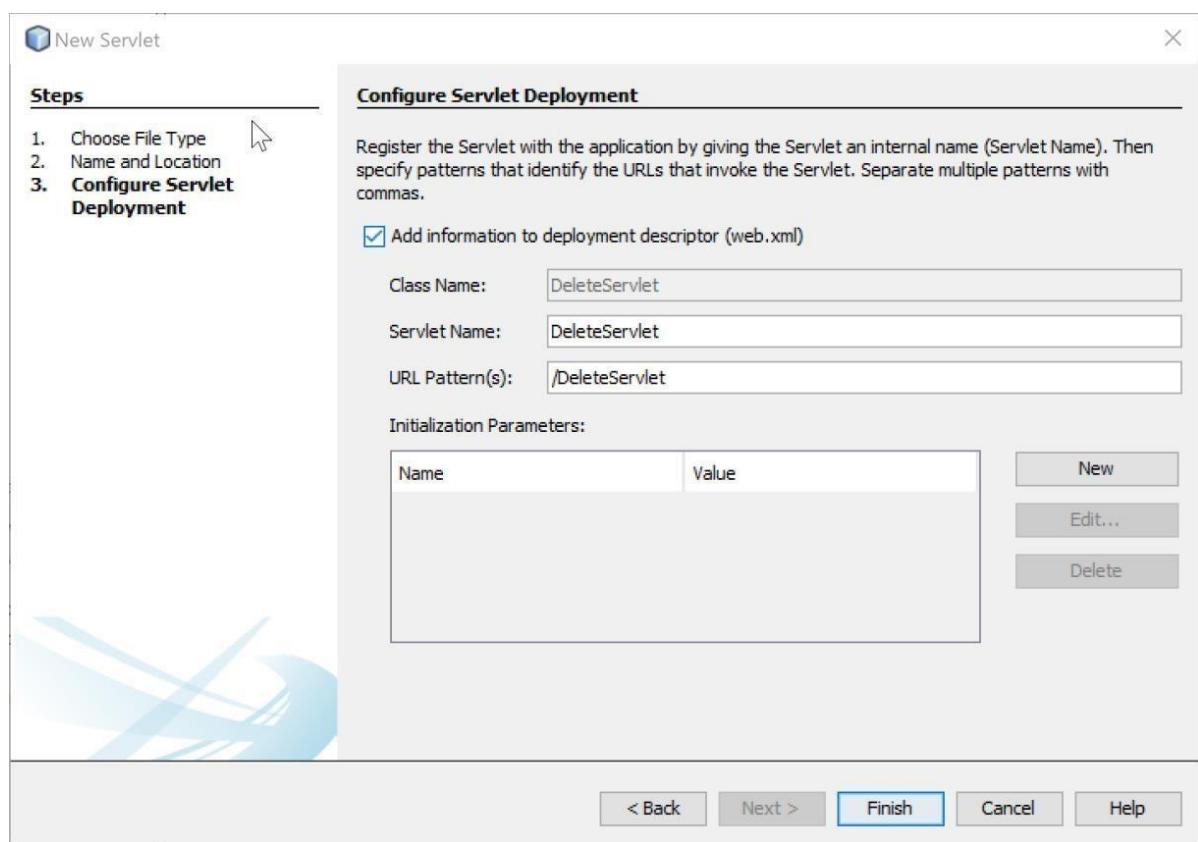
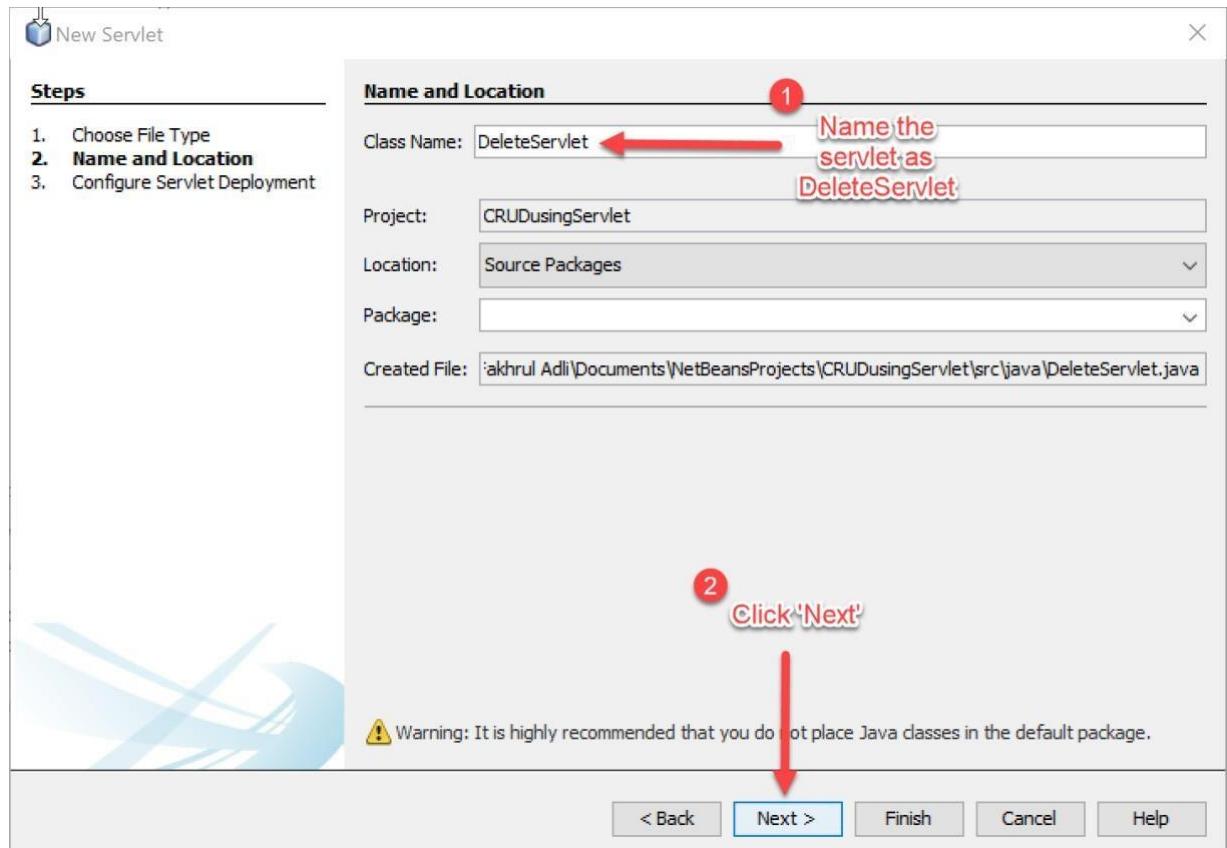
22. Type the following codes in the processRequest() method in

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 points to the code with the text: "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.

```
1  ...5 lines
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.servlet.ServletException;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10
11
12
13
14  /**...4 lines */
15  public class DeleteServlet extends HttpServlet {
16
17
18      /**
19       * Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
20      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
21          throws ServletException, IOException {
22          String sid = request.getParameter("id");
23          int id = Integer.parseInt(sid);
24          UserDao.delete(id);
25          response.sendRedirect("ViewServlet");
26      }
27
28
29
30
31
32
33
34
35 }
```

Make your DeleteServlet.java looks like this

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:

1. What is the name of the Java Library that you need to import before coding the web application with database operations?
 - JDBC (Java Database Connectivity)
 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.
 - in web inf folder
 - <servlet-name>: Specifies a name for the servlet.
 - <servlet-class>: Specifies the fully qualified class name of the servlet.
 - <servlet-mapping>: Maps a servlet to a URL pattern.
 3. Define the usage of Data Access Object (DAO) servlet. How it ease the business process in your servlet-based web application?

- DAO servlet is a design pattern used to separate the business logic of an application from the data persistence logic. DAO servlet acts as an intermediary between the servlets and the database. It encapsulates all the database-related operations the querying, updating, inserting, and deleting data.

OUTPUT

Add user :

User Management

localhost:8080/CRUDUsingServlet/index.html

Add New User

Username: akim

Password: 1122

Role: admin

Save User

[view users](#)

User Management

localhost:8080/CRUDUsingServlet/SaveServlet

Record saved successfully!

Add New User

Username:

Password:

Role: admin

Save User

[view users](#)

View user :

A screenshot of a web browser window titled "localhost:8080/CRUDUsingServlet". The URL in the address bar is "localhost:8080/CRUDUsingServlet/ViewServlet". The page displays a table titled "User List" with the following data:

ID	Name	Password	Role	Edit	Delete
1	Ali	1234	Admin	Edit	Delete
2	Ahmad	4567	User	Edit	Delete
5	akim	1122	admin	Edit	Delete

Edit user :

A screenshot of a web browser window titled "localhost:8080/CRUDUsingServlet". The URL in the address bar is "localhost:8080/CRUDUsingServlet/EditServlet?id=1". The page displays a form titled "Update User" with the following fields:

Name:

Password:

Role:

A screenshot of a web browser window titled "User Management". The URL in the address bar is "localhost:8080/CRUDUsingServlet/EditServlet2". The page displays a message: "Record saved successfully!"

Add New User

Username:

Password:

Role:

[view users](#)

Add New User

User List

Id	Name	Password	Role	Edit	Delete
1	Ali	1234	User	Edit	Delete
2	Ahmad	4567	User	Edit	Delete
5	akim	1122	admin	Edit	Delete

Delete user :

Add New User

User List

Id	Name	Password	Role	Edit	Delete
1	Ali	1234	User	Edit	Delete
2	Ahmad	4567	User	Edit	Delete
5	akim	1122	admin	Edit	Delete

Add New User

User List

Id	Name	Password	Role	Edit	Delete
1	Ali	1234	User	Edit	Delete
2	Ahmad	4567	User	Edit	Delete