



UNIVERSITI MALAYSIA TERENGGANU

CSM3023 WEB BASED APPLICATION DEVELOPMENT (K1)

BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONORS

**LAB 7 – JSP : PERFORM, CREATE, UPDATE, RETRIEVE AND DELETE (CRUD)
SEMESTER 4 2023/2024**

Prepared for:

DR. MOHAMAD NOR HASSAN

Prepared by:

NUR ARINA BINTI ABDUL MALEK (S65361)



Week 7

JSP: Perform Create, Update, Retrieve and Delete (CRUD)

Web Programming 2



Lecturers

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

(UMT)

Revision History

Revision Date	Previous Revision Date	Summary of Changes	Changes Marked
		First Issue	Mohamad Nor Hassan
		Second Issue	Dr Rabiei Mamat Dr Faizah Aplop Dr Fouad Ts Dr Rosmayati Mohemad Fakhrul Adli Mohd Zaki
			Fakhrul Adli Mohd Zaki

Table of Contents

Task 1: Perform Basic CRUD Process Using Java Servlet	5
---	---

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: Perform Basic CRUD Process Using Java Servlet

Objective:

Using Java Servlet to perform creating, retrieving, updating and deleting (CRUD) records from MySQL database.

Problem Description:

You are required to perform basic CRUD process using Java Servlet.

1. Create table users in CSF3203 database schema.
2. Create a three Java class that representing User (act as a JavaBeans to represent business object), DBConnection (to open and close database connection) and UserDao (act as a Data Access Object (DAO) to perform CRUD process).
3. Create UserController servlet to control the CRUD process.
4. Create index.jsp page as a main page.
5. Create listUser.jsp page to perform retrieving of a list of users.
6. Create user.jsp page to create a new record for user.
7. Create editUser.jsp page to update existing record for specific user.

Estimated time: 120 minutes

Step 1 - Create table users in CSF3107 database schema

1. Open XAMPP Control Panel.
2. Start Apache and MySQL module.
3. Open phpMyAdmin by clicking Admin button for MySQL module.
4. Select CSF3107 database schema.
5. Go to SQL tab.
6. Create table known as users.



7. Click Go button to execute the query.

Step 2 - Create three Java class that representing User (act as a JavaBeans to represent business object), DBConnection (to open and close database connection) and UserDao (act as a Data Access Object (DAO)) to perform CRUD process

1. Create new web application as CRUDServlet.
2. Create Java class User to represent the business object for user.
3. Name the package as com.model.
4. Write a coding for getter and setter for each instance variable.

```
1  package com.model;
2
3  /**
4   * @author mohamadnor
5   */
6  public class User {
7      private String userid;
8      private String firstName;
9      private String lastName;
10
11     public String getUserid() {
12         return userid;
13     }
14
15     public void setUserid(String userid) {
16         this.userid = userid;
17     }
18
19     public String getFirstName() {
20         return firstName;
21     }
```



```

22
23 - public void setFirstName(String firstName) {
24     this.firstName = firstName;
25 }
26
27 - public String getLastName() {
28     return lastName;
29 }
30
31 - public void setLastName(String lastName) {
32     this.lastName = lastName;
33 }
34 }

```

5. Create Java class *DBConnection* to open and close the database.
6. Name the package as *com.util*
7. Write a coding for connecting and closing to database.

```

5 package com.util;
6
7 /**
8  *
9  * @author mohamadnor
10  */
11
12 import java.io.IOException;
13 import java.sql.Connection;
14 import java.sql.DriverManager;
15 import java.sql.SQLException;
16
17 public class DBConnection {
18     private static Connection myConnection=null;
19     private static String myURL=myURL = "jdbc:mysql://localhost:3306/csf3203";
20
21     DBConnection() {
22
23

```

```

23
24     public static Connection getConnection() throws ClassNotFoundException
25     {
26         if (myConnection != null)
27         {
28             return myConnection;
29         }
30         else
31         try
32         {
33             //Establish and open MySQL database connection.....
34             Class.forName("com.mysql.jdbc.Driver");
35             myConnection = DriverManager.getConnection(myURL, "root", "admin");
36         }
37         catch (SQLException e)
38         {
39             e.printStackTrace();
40         }
41         return myConnection;
42     }

```

```

44     public void closeConnection() throws ClassNotFoundException
45     {
46         try
47         {
48             myConnection.close();
49         }
50         catch (SQLException e)
51         {
52             e.printStackTrace();
53         }
54     }
55 }

```

8. Create Java class *UserDao* to perform CRUD process.
9. Name the package as *com.dao*.
10. Write codes to perform CRUD process.

```

5     package com.dao;
6
7     /**
8      *
9      * @author mohamadnor
10    */
11
12    import java.sql.Connection;
13    import java.sql.PreparedStatement;
14    import java.sql.ResultSet;
15    import java.sql.SQLException;
16    import java.sql.Statement;
17
18    import java.util.ArrayList;
19    import java.util.List;
20    import com.model.User;
21    import com.util.DBConnection;
22

```

```

24    public class UserDao {
25
26        private Connection connection;
27
28        public UserDao() throws ClassNotFoundException {
29            connection = DBConnection.getConnection();
30        }
31
32        public void addUser(User user) {
33            try {
34                PreparedStatement preparedStatement = connection
35                    .prepareStatement("insert into users(userid, firstname, lastname) values (?, ?, ?)");
36                // Parameters start with 1
37                preparedStatement.setString(1, user.getUserid());
38                preparedStatement.setString(2, user.getFirstName());
39                preparedStatement.setString(3, user.getLastName());
40                preparedStatement.executeUpdate();
41            } catch (SQLException e) {
42                e.printStackTrace();
43            }
44        }
45    }

```

```

46
47 public void deleteUser(String userId) {
48     try {
49         PreparedStatement preparedStatement = connection
50             .prepareStatement("delete from users where userid=?");
51         // Parameters start with 1
52         preparedStatement.setString(1, userId);
53         preparedStatement.executeUpdate();
54
55     } catch (SQLException e) {
56         e.printStackTrace();
57     }
58 }

```

```

59
60 public void updateUser(User user) {
61     try {
62         PreparedStatement preparedStatement = connection
63             .prepareStatement("update users set firstname=?, lastname=? " +
64                             "where userid=?");
65         // Parameters start with 1
66         preparedStatement.setString(1, user.getFirstName());
67         preparedStatement.setString(2, user.getLastName());
68         preparedStatement.setString(3, user.getUserid());
69         preparedStatement.executeUpdate();
70
71     } catch (SQLException e) {
72         e.printStackTrace();
73     }
74 }

```

```

75
76 public List<User> getAllUsers() {
77     List<User> users = new ArrayList<User>();
78     try {
79         Statement statement = connection.createStatement();
80         ResultSet rs = statement.executeQuery("select * from users");
81         while (rs.next()) {
82             User user = new User();
83             user.setUserid(rs.getString("userid"));
84             user.setFirstName(rs.getString("firstname"));
85             user.setLastName(rs.getString("lastname"));
86             users.add(user);
87         }
88     } catch (SQLException e) {
89         e.printStackTrace();
90     }
91
92     return users;
93 }

```

```

95
96     public User getUserById(String userId) {
97         User user = new User();
98         try {
99             PreparedStatement preparedStatement = connection.prepareStatement(
100                 "select * from users where userid=?");
101             preparedStatement.setString(1,userId);
102             ResultSet rs = preparedStatement.executeQuery();
103
104             while (rs.next()) {
105                 user.setUserid(rs.getString("userid"));
106                 user.setFirstName(rs.getString("firstname"));
107                 user.setLastName(rs.getString("lastname"));
108             }
109         } catch (SQLException e) {
110             e.printStackTrace();
111         }
112
113         return user;
114     }
115 }

```

Step 3 - Create UserController servlet in order to control and redirect the request to the respective CRUD process and page

1. Create a Java servlet known as *UserController*.
2. Name the package as *com.controller*.
3. Import the related API and package.

```

5     package com.controller;
6
7     import java.io.IOException;
8     import java.io.PrintWriter;
9     import java.text.ParseException;
10
11     import javax.servlet.RequestDispatcher;
12     import javax.servlet.ServletException;
13     import javax.servlet.http.HttpServlet;
14     import javax.servlet.http.HttpServletRequest;
15     import javax.servlet.http.HttpServletResponse;
16
17     import com.dao.UserDao;
18     import com.model.User;

```

4. Remove *processRequest()* method.
5. Define the static instance variables and the constructor.


```

24 public class UserController extends HttpServlet {
25
26     private static String INSERT = "/user.jsp";
27     private static String EDIT = "/editUser.jsp";
28     private static String LIST_USER = "/listUser.jsp";
29     private UserDao dao;
30
31     public UserController() throws ClassNotFoundException {
32         super();
33         dao = new UserDao();
34     }

```

6. Write a code for *doGet()* method in order to determine the respective CRUD process and redirect to related page request.

```

37 @Override
38 protected void doGet(HttpServletRequest request, HttpServletResponse response)
39     throws ServletException, IOException {
40     String forward="";
41     String action = request.getParameter("action");
42
43     if (action.equalsIgnoreCase("delete")) {
44         String userId = request.getParameter("userId");
45         dao.deleteUser(userId);
46         forward = LIST_USER;
47         request.setAttribute("users", dao.getAllUsers());
48     }
49     else if (action.equalsIgnoreCase("edit")) {
50         forward = EDIT;
51         String userId = request.getParameter("userId");
52         User user = dao.getUserById(userId);
53         request.setAttribute("user", user);
54     }
55     else if (action.equalsIgnoreCase("listUser")){
56         forward = LIST_USER;
57         request.setAttribute("users", dao.getAllUsers());
58     }
59     else if (action.equalsIgnoreCase("insert")) {
60         forward = INSERT;
61     }
62
63     RequestDispatcher view = request.getRequestDispatcher(forward);
64     view.forward(request, response);
65 }

```

7. Write a code for doPost() method in order to perform creating or updating the record and finally, redirect to related page request.

```
67  @Override
68  protected void doPost(HttpServletRequest request, HttpServletResponse response)
69      throws ServletException, IOException {
70      String action = request.getParameter("action");
71
72      User user = new User();
73      user.setUserid(request.getParameter("userid"));
74      user.setFirstName(request.getParameter("firstName"));
75      user.setLastName(request.getParameter("lastName"));
76
77      if( action.equalsIgnoreCase("edit") )
78      {
79          dao.updateUser(user);
80      }
81      else
82      {
83          dao.addUser(user);
84      }
85
86      RequestDispatcher view = request.getRequestDispatcher(LIST_USER);
87      request.setAttribute("users", dao.getAllUsers());
88      view.forward(request, response);
89  }
90 }
```

Step 4 - Create an index.jsp page that act as a main page

1. Create jsp page and key-in filename as *index.jsp*.
2. Write an HTML markup and JSP action tag to forward the page to UserController servlet with URL parameter as *action=listUser*.

```
7  <@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Sample Perform CRUD Using Java Servlet</title>
13     </head>
14     <body>
15         <h1>Sample Perform CRUD Using Java Servlet</h1>
16         <jsp:forward page="/UserController?action=listUser" />
17     </body>
18 </html>
19
```

3. Compile the file

Step 5 - Create listUser.jsp page to perform retrieving of a list of users.

1. Create jsp page and key-in filename as listUser.jsp.
2. Add standard.jar and jstl.jar in library project folder.
3. Add the taglib directive to listUser.jsp.

```
4      Author      : mohamadnor
5      -->
6
7      <@page contentType="text/html" pageEncoding="UTF-8"%>
8      <@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9      <@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
```

4. Write and HTML markup and JSTL syntax to display the records.

```
11 <!DOCTYPE html>
12 <html>
13 <head>
14 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15 <title>User Lists</title>
16 </head>
17 <body>
18 <p>List of Users..!</p>
19 <table border=1>
20 <thead>
21 <tr>
22 <th>User Id</th>
23 <th>First Name</th>
24 <th>Last Name</th>
25 <th colspan=2>Action</th>
26 </tr>
27 </thead>
28 <tbody>
```

```
30 <c:forEach items="${users}" var="user">
31 <tr>
32 <td><c:out value="${user.userid}" /></td>
33 <td><c:out value="${user.firstName}" /></td>
34 <td><c:out value="${user.lastName}" /></td>
35 <td><a href="UserController?action=edit&userId=<c:out value="${user.userid}" />">Update</a></td>
36 <td><a href="UserController?action=delete&userId=<c:out value="${user.userid}" />">Delete</a></td>
37 </tr>
38 </c:forEach>
39 </tbody>
40 </table>
41 <p><a href="UserController?action=insert">Add User</a></p>
42 </body>
43 </html>
```

5. Compile the page

Step 6 - Create user.jsp page as a page for creating new record for user.

1. Create jsp page and key-in file name as user.jsp.
2. Create HTML markup and call UserController servlet from this page.

```
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>New record</title>
13 </head>
14 <body>
15     <br/>
16     <p><b>New Record</b></p>
17     <form name="frmAddUser" action="UserController" method="POST">
18         <table border="0">
19             <tbody>
20                 <tr>
21                     <td>User ID :</td>
22                     <td><input type="text" name="userid" value="" size="25" required /></td>
23                 </tr>
24                 <tr>
25                     <td>First Name :</td>
26                     <td><input type="text" name="firstName" value="" size="40" /></td>
27                 </tr>
28                 <tr>
29                     <td>Last Name :</td>
30                     <td><input type="text" name="lastName" value="" size="40" /></td>
31                 </tr>
32                 <tr>
33                     <td><input type="hidden" name="action" value="insert" /></td>
34                 </tr>
35                 <tr>
36                     <td>
37                         <input type="submit" value="Submit" name="submit" />
38                         <input type="reset" value="Cancel" name="cancel" />
39                     </td>
40                 </tr>
41             </tbody>
42         </table>
43     </form>
44 </body>
45 </html>
```

3. Compile the file.

Step 7 - Create editUser.jsp page as a page for updating existing record for specific user.

1. Create jsp page and key-in file name as editUser.jsp.
2. Add the taglib directive to editUser.jsp.

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9 <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
10
```

3. Write and HTML markup and JSTL syntax to display the records.

```
11 <!DOCTYPE html>
12 <html>
13 <head>
14 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15 <title>Edit User</title>
16 </head>
17 <body>
18 <p>Updating User Profile</p>
19 <form name="frmEditUser" action="UserController" method="POST">
20 <table border="0">
21 <tbody>
22 <tr>
23 <td>User ID :</td>
24 <td><input type="text" name="userid" readonly="readonly" value="<c:out value='${user.userid}' />" size="25" /></td>
25 </tr>
26 <tr>
27 <td>First Name :</td>
28 <td><input type="text" name="firstName" value="<c:out value='${user.firstName}' />" size="40" /></td>
29 </tr>
30 <tr>
31 <td>Last Name :</td>
32 <td><input type="text" name="lastName" value="<c:out value='${user.lastName}' />" size="40" /></td>
33 </tr>
34 <tr>
35 <td><input type="hidden" name="action" value="edit" /></td>
36 </tr>
37 <tr>
38 <td>
39 <input type="submit" value="Submit" name="submit" />
40 </td>
41 </tr>
42 </tbody>
43 </table>
44 </form>
45 </body>
46 </html>
```

4. Compile the file.

Running the program and perform CRUD process

1. Run index.jsp page.
2. Click Add User button to create new record.
3. Click hyperlink Update in order to update an existing record.
4. Click hyperlink Delete in order to delete an existing record.

Reflection:

1. Why we use servlet for Java Web Application?

MY WORK

```
1 • CREATE TABLE users(  
2     userid VARCHAR(15) NOT NULL,  
3     firstname VARCHAR(35),  
4     lastname VARCHAR(15),  
5     CONSTRAINT user_userid_pk PRIMARY KEY(userid)  
6 );  
7
```

User.java

```
package com.model;  
  
/**  
 *  
 * @author ryna  
 */  
public class User {  
    private String userid;  
    private String firstName;  
    private String lastName;  
  
    public String getUserid() {  
        return userid;  
    }  
  
    public void setUserid(String userid) {  
        this.userid = userid;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
}
```

```
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

}
```

DBConnection.java

```
5 package com.util;
6
7 /**
8  *
9  * @author rynaas
10  */
11 import java.io.IOException;
12 import java.sql.Connection;
13 import java.sql.DriverManager;
14 import java.sql.SQLException;
15
16 public class DBConnection {
17     private static Connection myConnection=null;
18     private static String myURL=myURL = "jdbc:mysql://localhost:3306/csm3023_lab7";
19
20     DBConnection() {
21     }
22
23
24     public static Connection getConnection() throws ClassNotFoundException
25     {
26         if (myConnection != null)
27         {
28             return myConnection;
29         }
30         else
31         try
32         {
33             //Establish and open MYSQL database connection....
34             Class.forName("com.mysql.jdbc.Driver");
35             myConnection = DriverManager.getConnection(myURL, "root", "admit");
36
37         }
38         catch (SQLException e)
39         {
40             e.printStackTrace();
41         }
42         return myConnection;
43     }
44
45     public void closeConnection() throws ClassNotFoundException
46     {
47         try
48         {
49             myConnection.close();
50         }
51         catch (SQLException e)
52         {
53             e.printStackTrace();
54         }
55     }
56 }
```

UserDAO.java

```
4  /**
5  package com.dao;
6
7  /**
8  *
9  * @author ryna
10 */
11 import java.sql.Connection;
12 import java.sql.PreparedStatement;
13 import java.sql.ResultSet;
14 import java.sql.SQLException;
15 import java.sql.Statement;
16
17 import java.util.ArrayList;
18 import java.util.List;
19 import com.model.User;
20 import com.util.DBConnection;
21
22 public class UserDao {
23
24     private Connection connection;
25
26     public UserDao() throws ClassNotFoundException{
27         connection = DBConnection.getConnection();
28     }
29
30     public void addUser(User user) {
31         try{
32             PreparedStatement preparedStatement = connection
33                 .prepareStatement("insert into users(userid, firstname, lastname) values (?, ?, ?)");
34             //Parameters start with 1
35             preparedStatement.setString(1, user.getUserid());
36             preparedStatement.setString(2, user.getFirstName());
37             preparedStatement.setString(3, user.getLastName());
38             preparedStatement.executeUpdate();
39
40         } catch (SQLException e){
41             e.printStackTrace();
42         }
43     }
44
45     public void deleteUser (String userId){
46         try{
47             PreparedStatement preparedStatement = connection
48                 .prepareStatement("delete from users where userid=?");
49             //parameters start with 1
50             preparedStatement.setString(1, userId);
51             preparedStatement.executeUpdate();
52         } catch (SQLException e){
53             e.printStackTrace();
54         }
55     }
56
57     public void updateUser(User user){
58         try{
59             PreparedStatement preparedStatement = connection
60                 .prepareStatement("update users set firstname=?, lastname=?" +
61                     "where userid=?");
62             //parameters start with 1
63             preparedStatement.setString(1, user.getFirstName());
64             preparedStatement.setString(2, user.getLastName());
65             preparedStatement.setString(3, user.getUserid());
66             preparedStatement.executeUpdate();
67         } catch (SQLException e){
68             e.printStackTrace();
69         }
70     }
71 }
```

```
1 public List<User> getAllUsers(){
2     List<User> users = new ArrayList<User>();
3
4     try{
5         Statement statement = connection.createStatement();
6         ResultSet rs = statement.executeQuery("select * from users");
7         while (rs.next()){
8             User user = new User();
9             user.setUserid(rs.getString("userid"));
10            user.setFirstName(rs.getString("firstName"));
11            user.setLastName(rs.getString("lastName"));
12            users.add(user);
13        }
14    } catch (SQLException e){
15        e.printStackTrace();
16    }
17
18    return users;
19 }
20
21 public User getUserById(String userId){
22     User user = new User();
23     try{
24         PreparedStatement preparedStatement = connection.prepareStatement(
25             "select * from users where userid=?");
26         preparedStatement.setString(1,userId);
27         ResultSet rs = preparedStatement.executeQuery();
28
29         while (rs.next()){
30             user.setUserid(rs.getString("userid"));
31             user.setFirstName(rs.getString("firstName"));
32             user.setLastName(rs.getString("lastName"));
33         }
34     }
35 }
36
37 com.dao.UserDao > updateUser > try > catch SQLException e >
38
39 103     }
40 104 } catch (SQLException e){
41     e.printStackTrace();
42 106 }
43 107 return user;
44 108 }
45 109 }
46 110
47 111
```

UserController.java

```
3  * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
4  */
5  package com.controller;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import java.text.ParseException;
10 import jakarta.servlet.RequestDispatcher;
11 import jakarta.servlet.ServletException;
12 import jakarta.servlet.annotation.WebServlet;
13 import jakarta.servlet.http.HttpServlet;
14 import jakarta.servlet.http.HttpServletRequest;
15 import jakarta.servlet.http.HttpServletResponse;
16
17 import com.dao.UserDao;
18 import com.model.User;
19
20 /**
21  *
22  * @author ryna
23  */
24 @WebServlet(name = "UserController", urlPatterns = {"/UserController"})
25 public class UserController extends HttpServlet {
26
27     /**
28      * Processes requests for both HTTP GET and POST
29      * methods.
30      *
31      * @param request servlet request
32      * @param response servlet response
33      * @throws ServletException if a servlet-specific error occurs
34      * @throws IOException if an I/O error occurs
35      */
36
37     private static String INSERT = "/user.jsp";
38     private static String EDIT = "/editUser.jsp";
39     private static String LIST_USER = "/listUser.jsp";
40     private UserDao dao;
41
42     public UserController() throws ClassNotFoundException {
43         super();
44         dao = new UserDao();
45     }
46
47
48     // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
49     /**
50      * Handles the HTTP GET method.
51      *
52      * @param request servlet request
53      * @param response servlet response
54      * @throws ServletException if a servlet-specific error occurs
55      * @throws IOException if an I/O error occurs
56      */
57     @Override
58     protected void doGet(HttpServletRequest request, HttpServletResponse response)
59         throws ServletException, IOException {
60         String forward="";
61         String action = request.getParameter("action");
62
63         if (action.equalsIgnoreCase("delete")){
64             String userId = request.getParameter("userId");
65             dao.deleteUser(userId);
66             forward = LIST_USER;
67             request.setAttribute("users", dao.getAllUsers());
68         }
69     }
70
71 }
```

```

71         else if (action.equalsIgnoreCase("edit")){
72             forward = EDIT;
73             String userId = request.getParameter("userId");
74             User user = dao.getUserById(userId);
75             request.setAttribute("users", dao.getAllUsers());
76         }
77
78         else if (action.equalsIgnoreCase("listUser")){
79             forward = LIST_USER;
80             request.setAttribute("users", dao.getAllUsers());
81         }
82
83         else if (action.equalsIgnoreCase("insert")){
84             forward = INSERT;
85         }
86
87         RequestDispatcher view = request.getRequestDispatcher(forward);
88         view.forward(request, response);
89     }
90
91     /**
92     * Handles the HTTP <code>POST</code> method.
93     *
94     * @param request servlet request
95     * @param response servlet response
96     * @throws ServletException if a servlet-specific error occurs
97     * @throws IOException if an I/O error occurs
98     */
99     @Override
100     protected void doPost(HttpServletRequest request, HttpServletResponse response)
101         throws ServletException, IOException {
102
103         String action = request.getParameter("action");
104         //String action = request.getParameter("action");
105
106         User user = new User();
107         user.setUserId(request.getParameter("userId"));
108         user.setFirstName(request.getParameter("firstName"));
109         user.setLastName(request.getParameter("lastName"));
110
111         if(action.equalsIgnoreCase("edit"))
112         {
113             dao.updateUser(user);
114         }
115         else
116         {
117             dao.addUser(user);
118         }
119
120         RequestDispatcher view = request.getRequestDispatcher(LIST_USER);
121         request.setAttribute("users", dao.getAllUsers());
122         view.forward(request, response);
123     }
124
125     /**
126     * Returns a short description of the servlet.
127     *
128     * @return a String containing servlet description
129     */
130     @Override
131     public String getServletInfo() {
132         return "Short description";
133     } // </editor-fold>
134
135 }
136
137

```


Exercise

Q1) Implement profile registration using servlet

1. Create a table known as *userprofile* using database schema *CF3107* using these attributes.

- username as a character length 15 and must be primary key
- icno as a character length 15
- firstname as varchar(50)

2. Create an entry form.

3. Create a servlet known as *profileServlet*.

4. Use *profileServlet* to acknowledge user about the profile registration.

Q2) Applying session in student registration.

1. Create main interface for student registration; studentid, name.
(*studentRegister.jsp*)
2. When student click Submit button, it will redirect to confirmation page
(*confirmRegister.jsp*)
3. When user click Proceed button, current page will forward notification to end user via Notification page (*notificationRegister.jsp*)

MY Work

```
8 CREATE TABLE userprofile(  
9     username VARCHAR(15) PRIMARY KEY,  
10    icno VARCHAR(15),  
11    firstname VARCHAR(50)  
12 );
```

profileServlet.java

```
1  /*  
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template  
4   */  
5  package com.controller;  
6  
7  import java.io.IOException;  
8  import java.io.PrintWriter;  
9  import java.sql.Connection;  
10 import java.sql.DriverManager;  
11 import java.sql.PreparedStatement;  
12 import java.sql.SQLException;  
13 import jakarta.servlet.ServletException;  
14 import jakarta.servlet.annotation.WebServlet;  
15 import jakarta.servlet.http.HttpServlet;  
16 import jakarta.servlet.http.HttpServletRequest;  
17 import jakarta.servlet.http.HttpServletResponse;  
18  
19  /*  
20   * @author ryna  
21   */  
22  @WebServlet(name = "profileServlet", urlPatterns = {"/profileServlet"})  
23  public class profileServlet extends HttpServlet {  
24  
25      /**  
26       * Processes requests for both HTTP GET and POST  
27       * methods.  
28       *  
29       * @param request servlet request  
30       * @param response servlet response  
31       * @throws ServletException if a servlet-specific error occurs  
32       * @throws IOException if an I/O error occurs  
33       */  
34      protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
35          throws ServletException, IOException {
```

```

35 response.setContentType("text/html;charset=UTF-8");
36
37 PrintWriter out = response.getWriter();
38 Connection conn = null;
39 PreparedStatement stmt = null;
40
41 try {
42     // Get form data
43     String username = request.getParameter("username");
44     String icno = request.getParameter("icno");
45     String firstname = request.getParameter("firstname");
46
47     // Load the JDBC driver
48     Class.forName("com.mysql.cj.jdbc.Driver");
49
50     // Establish a connection
51     conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/csm3023", "root", "admin");
52
53     // Prepare the SQL statement
54     String sql = "INSERT INTO userprofile (username, icno, firstname) VALUES (?, ?, ?)";
55     stmt = conn.prepareStatement(sql);
56     stmt.setString(1, username);
57     stmt.setString(2, icno);
58     stmt.setString(3, firstname);
59
60     // Execute the statement
61     int rows = stmt.executeUpdate();
62
63     if (rows > 0) {
64         out.println("<h1>Profile registered successfully!</h1>");
65     } else {
66         out.println("<h1>Profile registration failed.</h1>");
67     }
68 } catch (SQLException | ClassNotFoundException e) {
69     out.println("<h1>Error: " + e.getMessage() + "</h1>");
70     e.printStackTrace();
71 } finally {
72     try {
73         if (stmt != null) stmt.close();
74         if (conn != null) conn.close();
75     } catch (SQLException e) {
76         e.printStackTrace();
77     }
78 }
79
80 out.close();
81
82
83 // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
84 /**
85  * Handles the HTTP <code>GET</code> method.
86  *
87  * @param request servlet request
88  * @param response servlet response
89  * @throws ServletException if a servlet-specific error occurs
90  * @throws IOException if an I/O error occurs
91  */
92 @Override
93 protected void doGet(HttpServletRequest request, HttpServletResponse response)
94     throws ServletException, IOException {
95     response.setContentType("text/html");
96     PrintWriter out = response.getWriter();
97     out.println("<html>");
98     out.println("<head><title>Profile Registration</title></head>");
99     out.println("<body>");
100     out.println("<h2>Profile Registration Form</h2>");
101     out.println("<form action='profileServlet' method='post'>");
102     out.println("<form action='profileServlet' method='post'>");
103     out.println("Username: <input type='text' name='username' maxlength='15' required><br><br>");
104     out.println("IC Number: <input type='text' name='icno' maxlength='15' required><br><br>");
105     out.println("First Name: <input type='text' name='firstname' maxlength='50' required><br><br>");
106     out.println("<input type='submit' value='Register'>");
107     out.println("</form>");
108     out.println("</body>");
109     out.println("</html>");
110
111 }
112
113 @Override
114 protected void doPost(HttpServletRequest request, HttpServletResponse response)
115     throws ServletException, IOException {
116     processRequest(request, response);
117 }
118
119 @Override
120 public String getServletInfo() {
121     return "Short description";
122 } // </editor-fold>

```

studentRegister.jsp

```
<%--
  Document   : studentRegister
  Created on : 1 Jun 2024, 2:25:23 am
  Author      : ryna
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page session="true" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Student Registration</title>
</head>
<body>
  <h1>Student Registration</h1>
  <form action="confirmRegister.jsp" method="post">
    <label for="studentid">Student ID:</label>
    <input type="text" id="studentid" name="studentid" required><br><br>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required><br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

confirmRegister.jsp

```
<%--
  Document   : confirmRegister
  Created on : 1 Jun 2024, 3:16:02 am
  Author      : ryna
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page session="true" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Confirm Registration</title>
</head>
<body>
  <h1>Confirm Registration</h1>
  <%
    String studentId = request.getParameter("studentid");
    String name = request.getParameter("name");

    // Store the student details in session
    session.setAttribute("studentid", studentId);
    session.setAttribute("name", name);
  %>
  <p>Student ID: <%= studentId %></p>
  <p>Name: <%= name %></p>
  <form action="notificationRegister.jsp" method="post">
    <input type="submit" value="Proceed">
  </form>
</body>
</html>
```

notifocationRegister.jsp

```
<%--
Document    : notificationRegister
Created on  : 2 Jun 2024, 2:26:19 pm
Author      : ryna
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page session="true" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Registration Notification</title>
</head>
<body>
  <h1>Registration Notification</h1>
  <%
    String studentId = (String) session.getAttribute("studentid");
    String name = (String) session.getAttribute("name");

    if (studentId != null && name != null) {
      <p>Registration successful for Student ID: <%= studentId %> and Name: <%= name %></p>
      session.invalidate(); // End the session
    } else {
      <p>No registration information found.</p>
    }
  %>
</body>
</html>
```