Week 6

JSP: Saving and Retrieving Data from Database

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
13/03/2019	21/02/2019	Addition of Revision History, Table of Contents, Formatting Cover Page	Fakhrul Adli Mohd Zaki

Table of Contents

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: Using JSP Page to Access a Simple MySQL Database

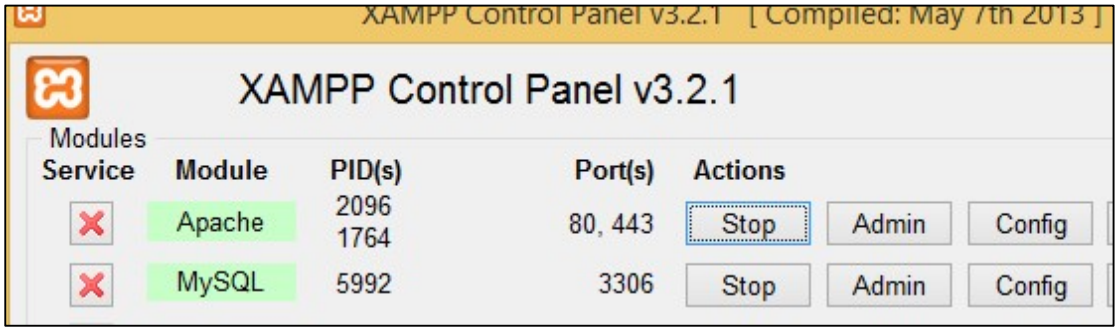
Objective:	Write a JSP that can insert data to MYSQL database as "Welcome to access MySQL database with JSP. ….!" and also display steps of how to connect with MYSQL database.
Problem Description:	1. Create a table known as FirstTable using database schema CF3107, create the first column as a character length 45. 2. Create SampleInsertionRecord.jsp page to process and acknowledge the user upon inserting record in the database.
Estimated time:	20 minutes

Step 1 - Create a table namely *FirstTable* using phpMyAdmin

1. Start XAMPP control panel.

2. Start the Apache web server.

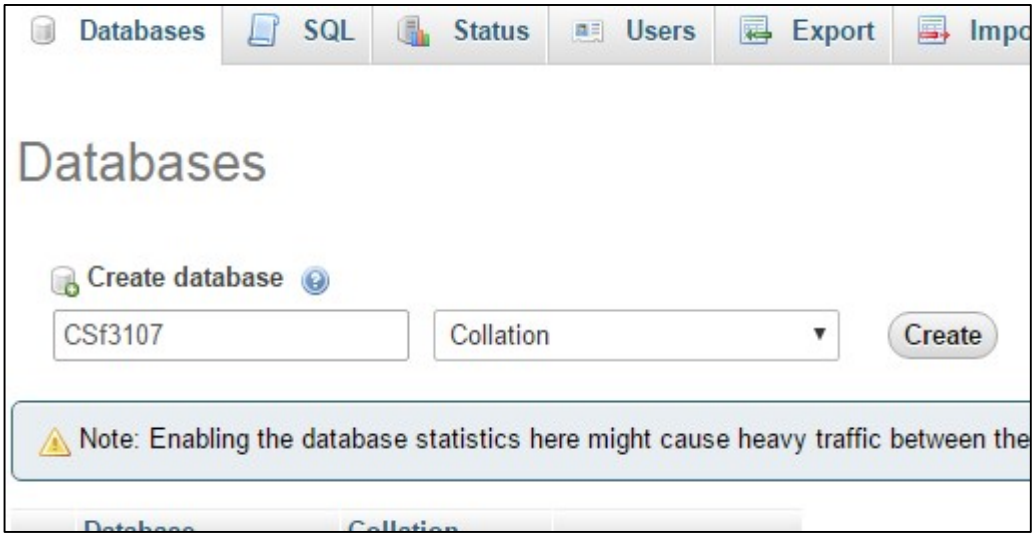
3. Start the MySQL database.



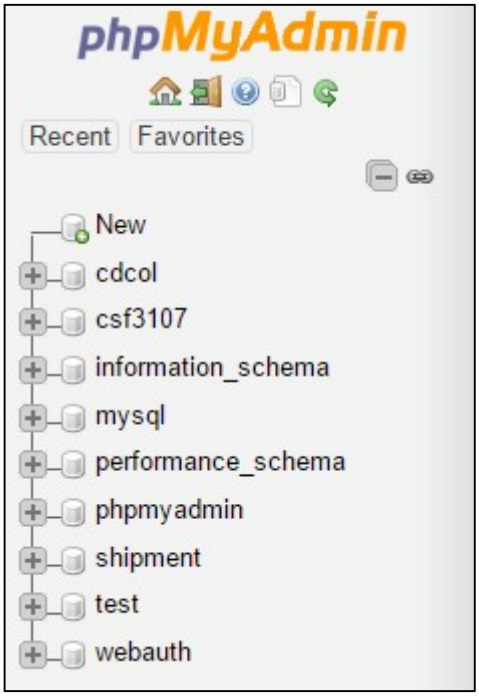
4. Click the *Admin* button for MySQL.

5. Go to *Database*'s tab.

6. Key-in as *CSF3107* and click *Create* button.



7. Database schema successfully created.

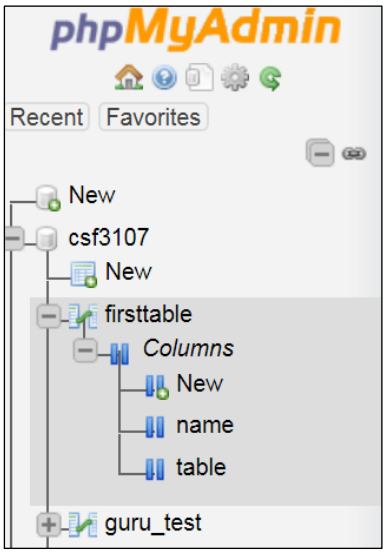


8. Use any tool to manipulate the SQL statement. Create table ***FirstTable*** in *csf3107* database schema.

9. Create ***FirstTable***'s table.

10. Execute the SQL statement.

11. Table successfully created.



Step 2 - Create *SampleInsertionRecord.jsp* to insert data in *FirstTable* table.

1. Go to C:\CSF3107 Lab's directory and create sub-directory as Lab 6

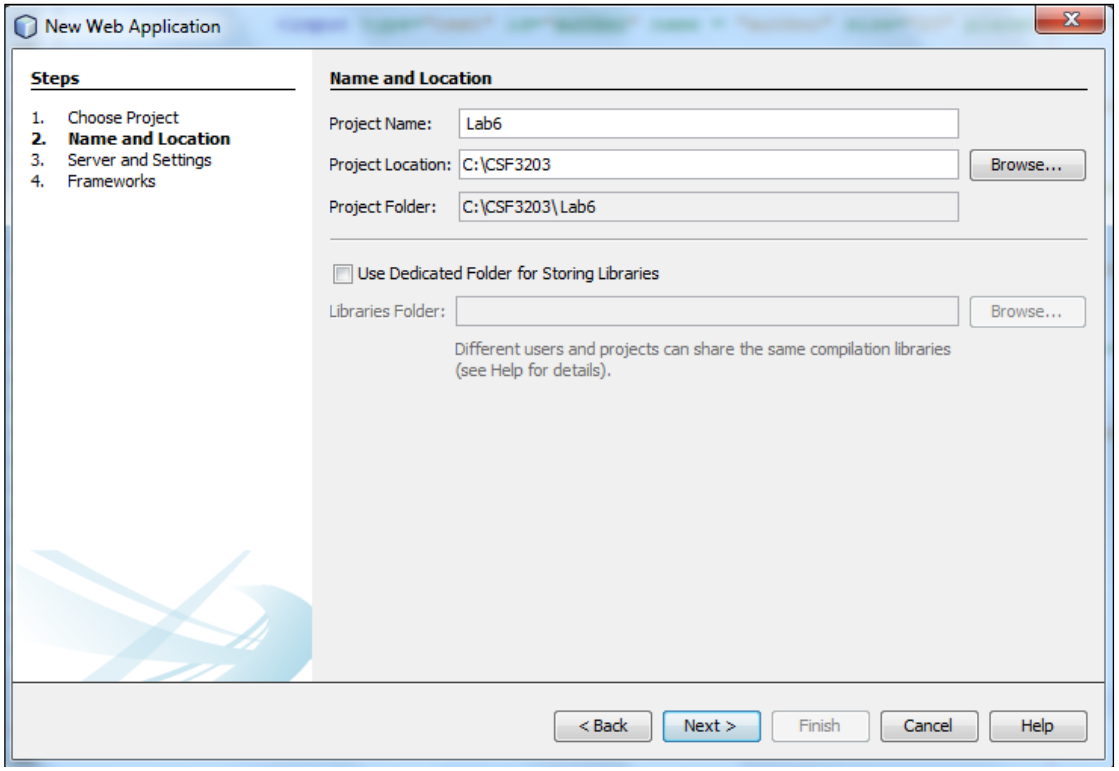
2. Go to NetBeans.

3. Go to File -> New Project.

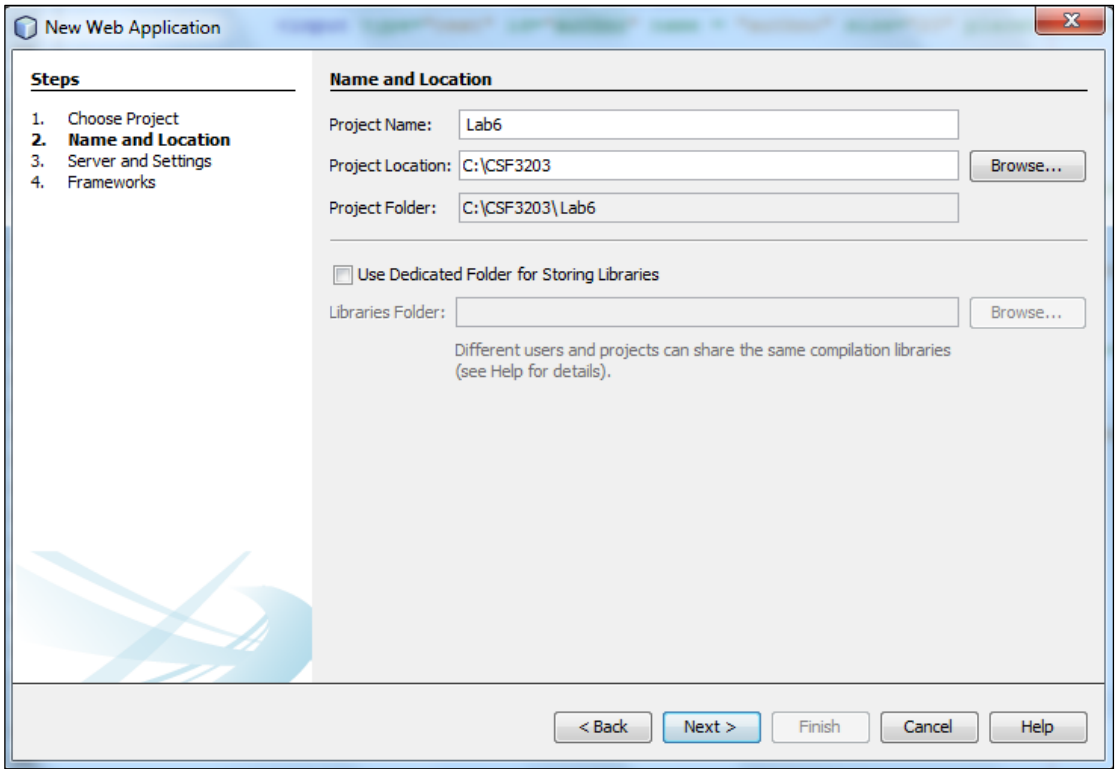
4. Select Java Web -> Web Application.

5. Click the *Next* button.

6. Type Project Name: *Lab6*.



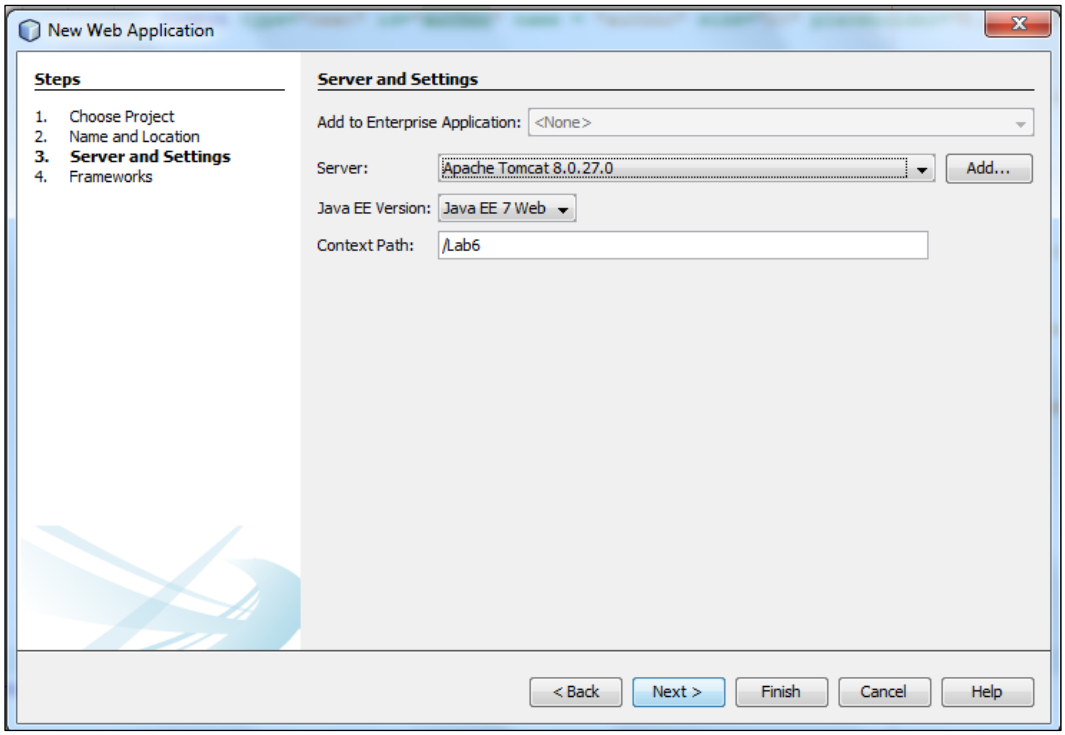
7. Choose Project Location: *C:\CSF3107\Lab6*



8. Click the *Next* button.

9. Select Server: *Apache Tomcat*.

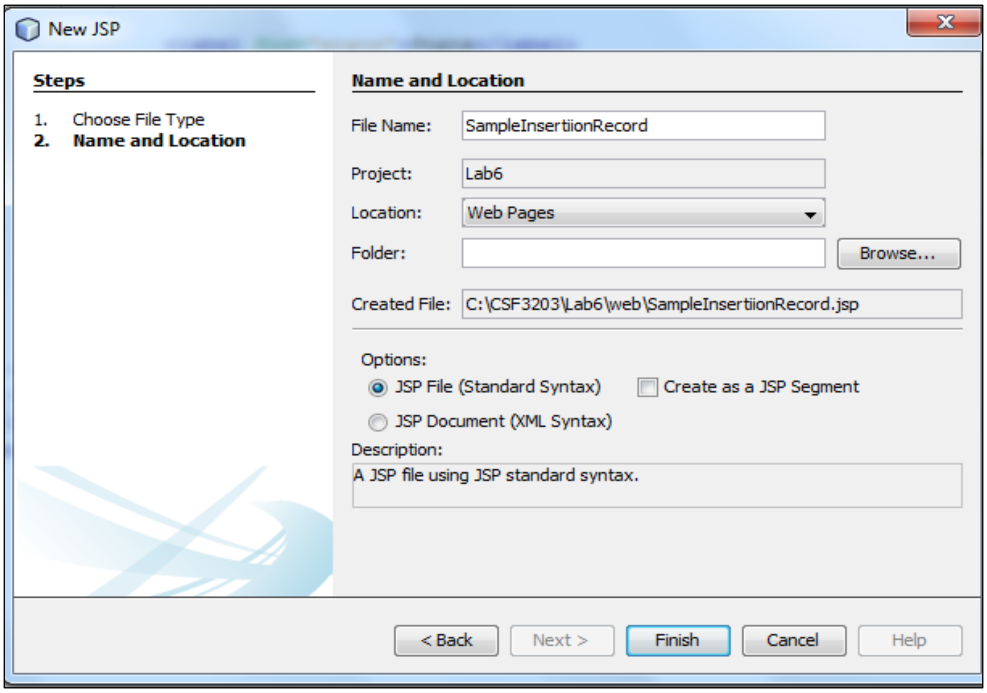
10. Select Java EE Version: Java EE 7 Web.



11. Click the *Next* button.

12. Click the *Finish* button.

13. Create a new JSP's page for and rename ***SampleInsertionRecord.***



14. Type header1 as ***Lab 6 - Task 1 - Sample Insertion records into MySQL through JSP's page.***

```
<h1>Lab 6 - Task 1 - Sample Insertion records into MySQL through JSP's page</h1>
<%
%>
```

15. To support the database driver, we need to use JSP Page Directive to provide directions and instructions to the container.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page language="java"%>
<%@page import="java.sql.*"%>
```

16. Write the following code:

```
<%

    int result;

    //Step 1: Load JDBC driver...
    Class.forName("com.mysql.jdbc.Driver");
    out.println("Step 1: MySQL driver loaded...!");
%>
<br>

<%
    //Step 2: Establish the connection...
    String myURL = "jdbc:mysql://localhost/csf3107";
    Connection myConnection = DriverManager.getConnection(myURL, "root", "admin");
    out.println("Step 2: Database is connected...!");
%>
<br>

<%
    //Step 3: Create a PreparedStatement object...
    out.println("Step 3: Prepared Statements created...!");

    //Prepared SQL Query as a String...
    String sInsertQry = "INSERT INTO FirstTable VALUE(?)";

    //Call method preparedStatement
    PreparedStatement myPS = myConnection.prepareStatement(sInsertQry);
%>
<br>
```

```
<%
    //Assign each value to respective columns for Book's table... (C-Create)
    out.println("Step 4: Perform insertion of record...!");
    String name = "Welcome to access MySQL database with JSP. ….!";
    myPS.setString(1, name);

    result = myPS.executeUpdate();

    if (result > 0) {
%>
<br>

<%
        out.println("Step 5: Close database connection...!");

        out.println(" ");
        out.println("Database connection is closed...!");

        out.print("<p>" + "The record : (" + name
                + ") is successfully created..!" + "</p>");
    }
    //Step 5: Close database connnection...!
    myConnection.close();
%>
```

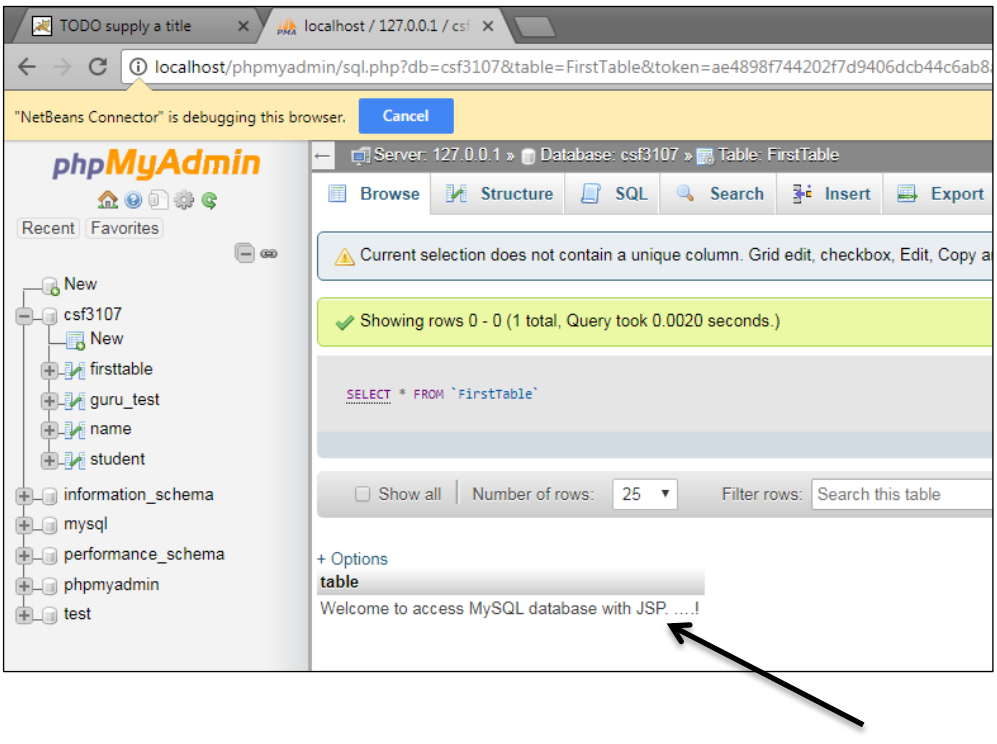
17. Save and compile ***SampleInsertionRecord.jsp*** file.

18. Run the ***SampleInsertionRecord.jsp*** file and sample of output is shown below:

Step 3 - go to the database

1. Go to Database schema (csf3107)

2. Click on-> *csf3107* -> *FirstTable* -> then *Browser* (see the data is already there!!)

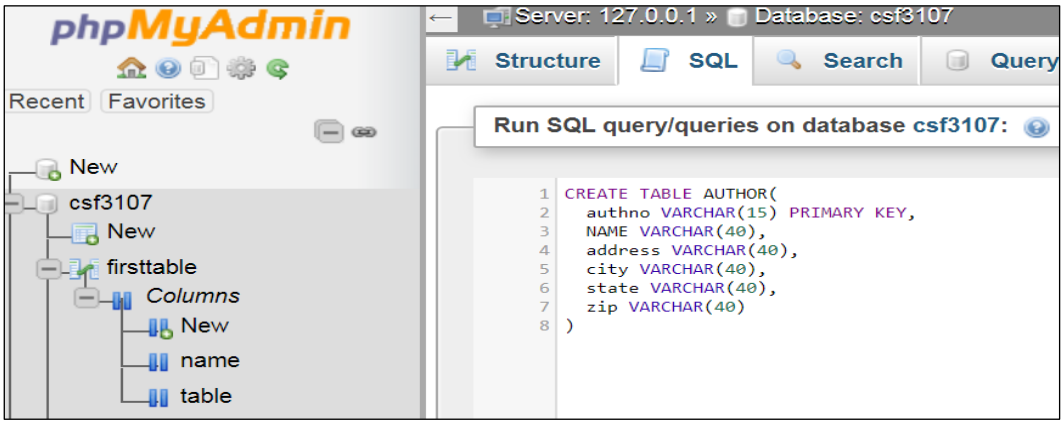


Task 2: Create Records via JSP Page

Objective:	Using JSP to insert records retrieve from MySQL database.
Problem Description:	1. Create a table known as **Author** using database schema CF3107 using these attributes: • **authno** as a character length 15 and must be primary key name. • **address** as a character length 40 • **city** as a character length 40 • **state** as a character length 40 • **zip** as a character length 40 2. Create **insertAuthor.jsp** as a main interface to register a new author. 3. Create **processAuthor.jsp** page to process and acknowledge the user upon inserting record in the database.
Estimated time:	40 minutes

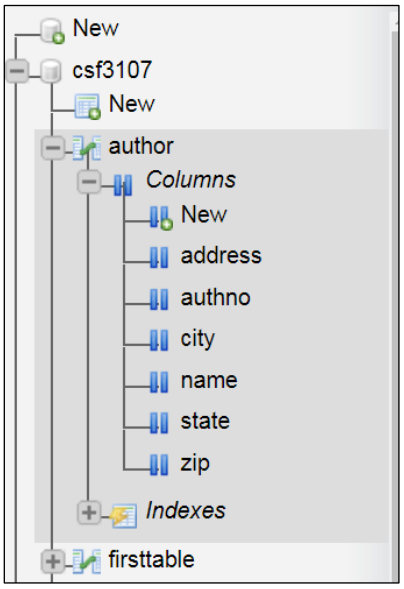
1. Use any tool to manipulate the SQL statement. Create table ***author*** in *csf3107* database schema.

2. Create ***author***'s table.

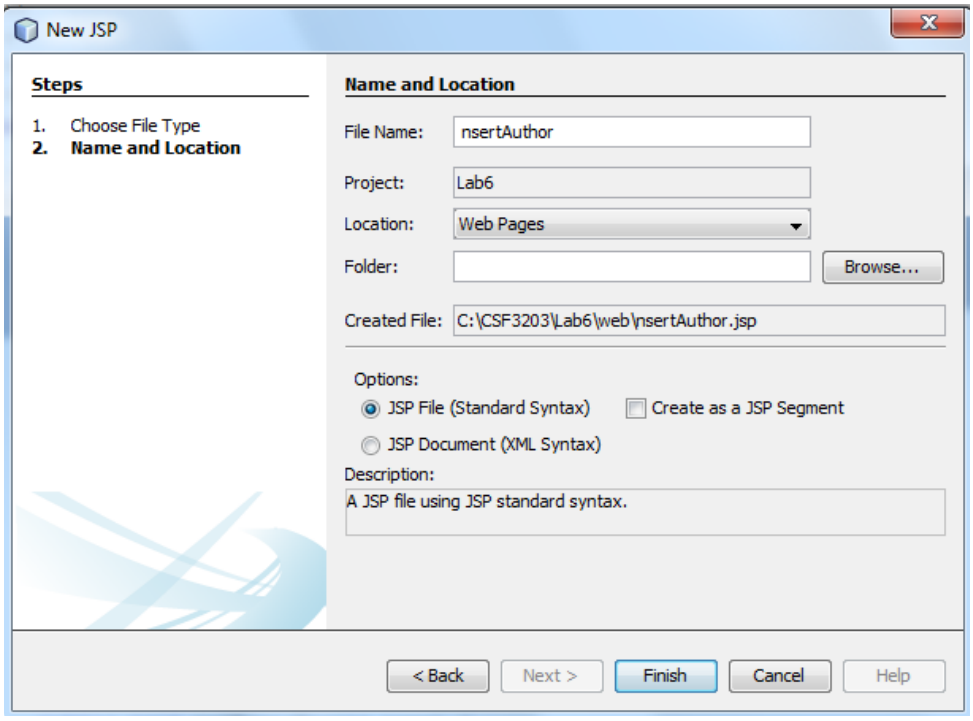


3. Execute the SQL statement.

4. Table successfully created.



5. Create a new JSP page and rename as *insertAuthor*.



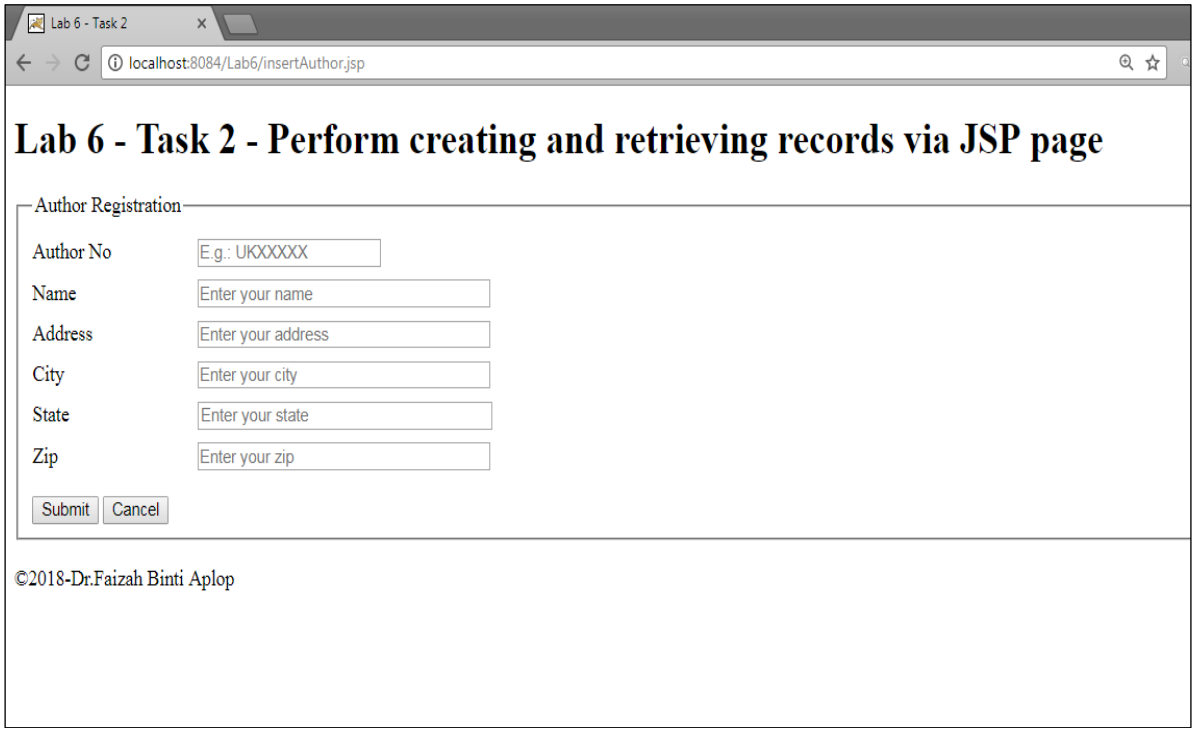
6. Write an HTML code to

   a. Display six (6) labels and textfields representing *Author No, Name, Address, City, State and Zip (in the combo box).*

   b. Create a *Submit* button and *Cancel* button.

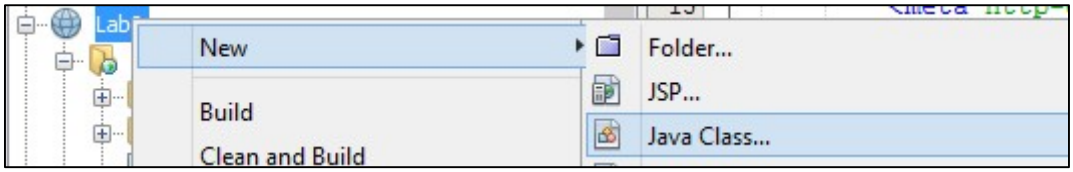
   c. Upon submission, redirect the page to *processAuthor.jsp* page.

7. Produce the following output;



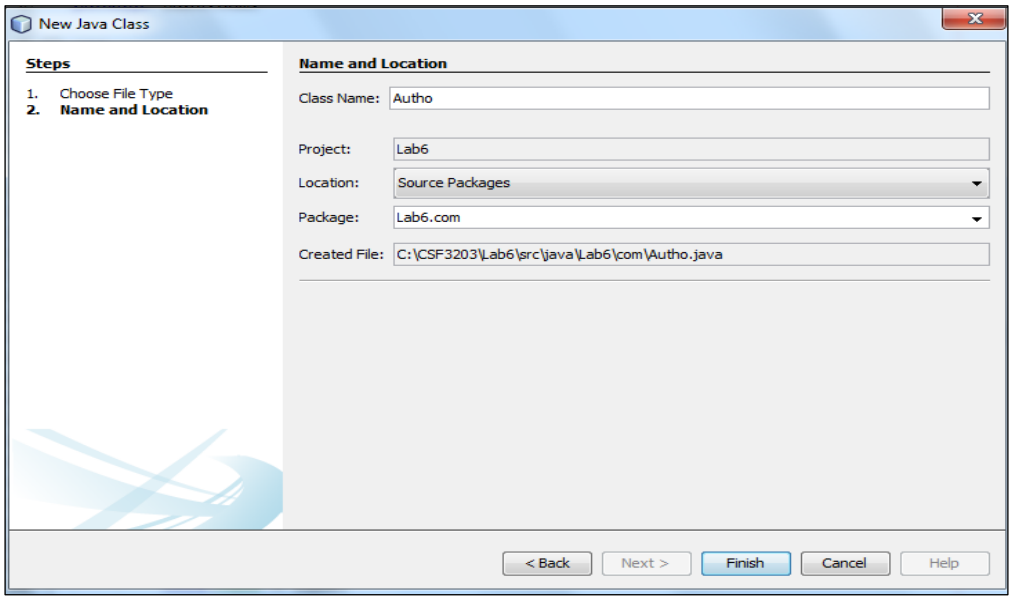
8. Go to *Lab6* project folder.

9. Right click -> New -> Java Class



10. Click Java Class

11. Rename Class Name as *Author* and package *as lab6.com*.



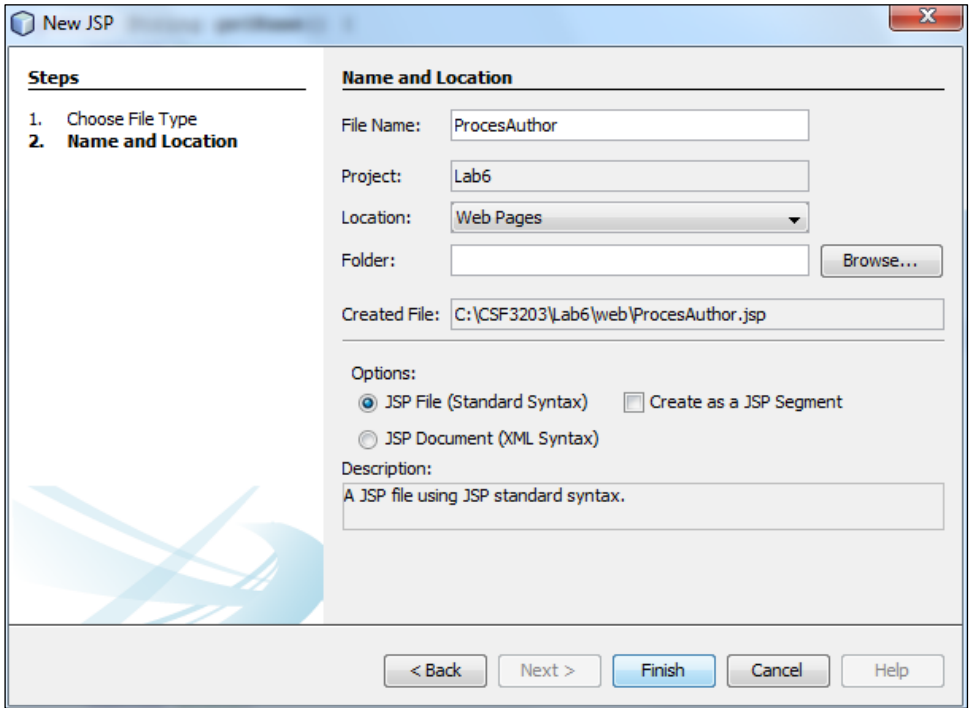
12. Define **Six (6)** instance variables for *Author* class.

```
 */
package Lab6.com;

/**
 *
 * @author fd
 */
public class author {

    private String authno;
    private String name;
    private String address;
    private String city;
    private String state;
    private String zip;
```

13. Define the *getter* and *setter* method for corresponding attributes.

```
public String getAuthno() {
    return authno;
}
public void setAuthno(String authno) {
    this.authno = authno;
}

public String getName() {
    return name;
}

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

public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}

public String getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

public String getState() {
    return state;
}

public void setState(String state) {
    this.state = state;
}

public String getZip() {
    return zip;
}

public void setZip(String zip) {
    this.zip = zip;
}
```

14. Create a new JSP page as *processAuthor*.



15. Add the page directive to *processAuthor.jsp* page.

```
6
7    <%@page contentType="text/html" pageEncoding="UTF-8"%>
8    <%@page language="java"%>
9    <%@page import="java.sql.*"%>
```

16. Create an *author*'s object using JSP Standard Action tag.

```
<jsp:useBean id="myAthour" class="Lab6.com.author" scope="request"/>
```

17. Assign data entry from page *insertAuthor.jsp* page into author's bean.

18. Load the database driver and create a connection to the database.

```
<h1>Lab 6 - Task 1 - Perform creating and retrieving records via JSP page</h1>

<jsp:setProperty name="myAuthor" property="*"/>

<%
    int result;

    Class.forName("com.mysql.jdbc.Driver");

    String myURL = "jdbc:mysql://localhost/csf3107";
    Connection myConnection = DriverManager.getConnection(myURL, "root", "admin");
```

19. Create a *PreparedStatement*'s object.

```
String sInsertQry = "INSERT INTO Author(authno, name, address, city, state, zip) VALUES(?, ?, ?, ?, ?, ?)";

PreparedStatement myPS = myConnection.prepareStatement(sInsertQry);

myPS.setString(1, myAuthor.getAuthno());
myPS.setString(2, myAuthor.getName());
myPS.setString(3, myAuthor.getAddress());
myPS.setString(4, myAuthor.getCity());
myPS.setString(5, myAuthor.getState());
myPS.setString(6, myAuthor.getZip());
```

20. Execute the query and display the result.

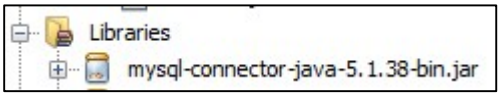
```
result = myPS.executeUpdate();
if (result > 0) {
    out.println("\tRecord successfully added into Author table...!");
    out.print("<p>" + "Record with author no " + myAuthor.getAuthno()
            + " successfully created..!" + "</p>");
    out.print("<p>" + "Details of record are; " + "</p>");
    out.print("<p>Name : " + myAuthor.getName() + "</p>");
    out.print("Address : " + myAuthor.getAddress() + "</p>");
    out.print("<p>City : " + myAuthor.getCity() + "</p>");
    out.print("<p>State : " + myAuthor.getState() + "</p>");
    out.print("<p>Zip : " + myAuthor.getZip() + "</p>");
```

21. Close database connection.

```
    //Step 5: Close database connnection...!
      System.out.println("Step 5: Close database connection...!");
      myConnection.close();
      System.out.println(" ");
      System.out.println("Database connection is closed...!");
%>
```

22. Save and compile *prosessAuthor.jsp* file.

IMPORTANT: Please add **MySQL Java connector** to your project before running the program.



23. Run *insertAuthor.jsp* page.

24. Key-in the record.

25. Click *Submit* button.

4. The record will save in the database, and user get a notification.

Lab 6 - Task 1 - Perform creating and retrieving records via JSP page

Record successfully added into Author table...!

Record with author no gsk23322 successfully created..!

Details of record are;

Name : Fouad

Address : Malaysia

City : KT

State : UMT

Zip : 23

Reflection

1. What have you learnt from this exercise?

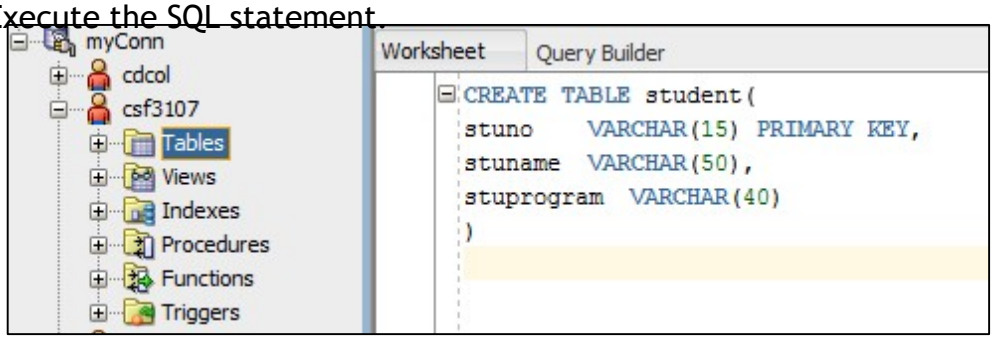
2. Define step by step before you successfully perform the transaction in a database.

Task 3: Create Records Constrained by Regular Expression In JSP

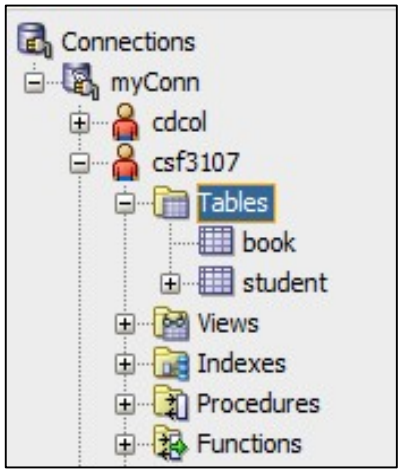
Objective:	Using JSP Standard Action, scriptlets and regular expression to insert records retrieve from MySQL database.
Problem Description:	1. Create a table known as **student** using database schema CF3107 using these attributes: • **stuid** as a character length 15 and must be the primary key • **stuname** as a character length 50 • **stuprogram** as a character length 40 • **address** as a character length 40 2. Create **insertStudent.jsp** as a main interface to register new book. 3. Create **processStudent.jsp** page to process and acknowledge the user upon inserting record in the database. 4. Create **displayStudent.jsp** page to populate records. 5. Create **errorStudent.jsp** to handle an error.
Estimated time:	50 minutes

Step 1 - Create a table book using phpMyAdmin

1. Create a table as a student in the *csf3107* database schema.
2. Execute the SQL statement.

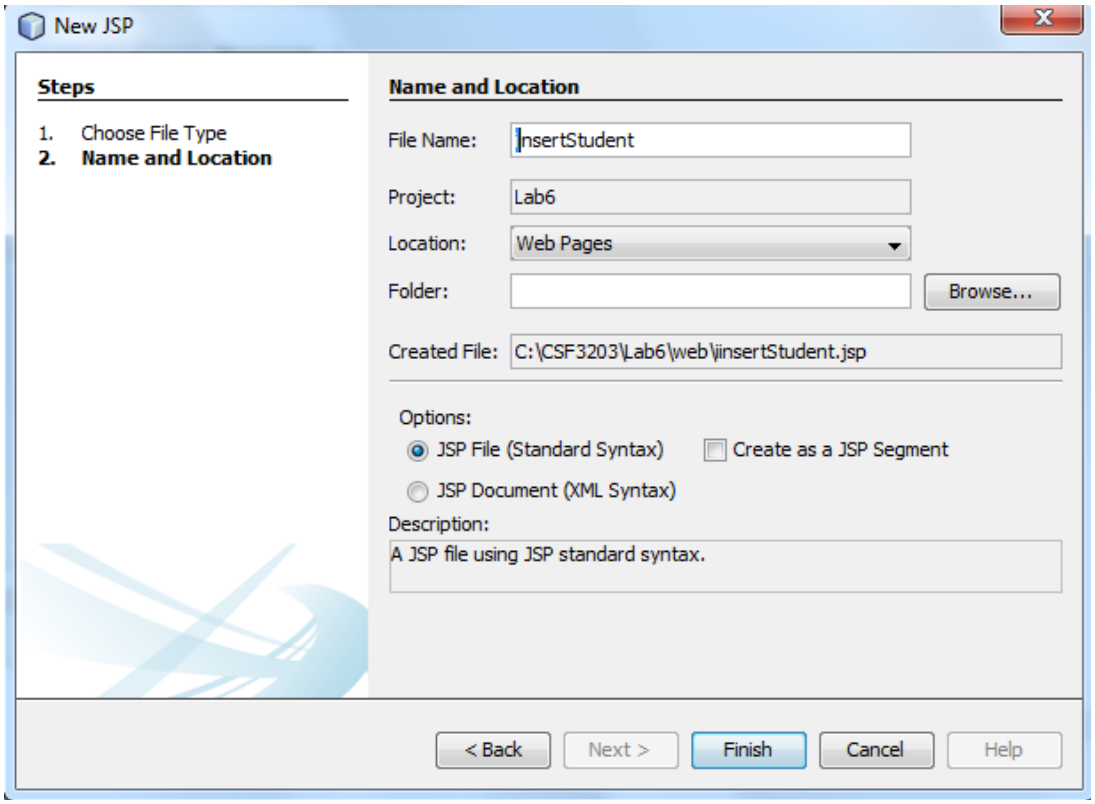


3. Table successfully created.



Step 2 - Create *insertStudent.jsp* as a main interface to register a new student

1. Create a new JSP's page and rename as *insertStudent*.

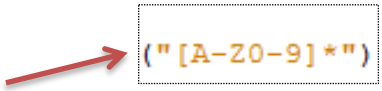


2. Write an HTML code to

   a. Display three (3) labels and textfields representing *Student ID, Name* and *Program (in the combo box)*.

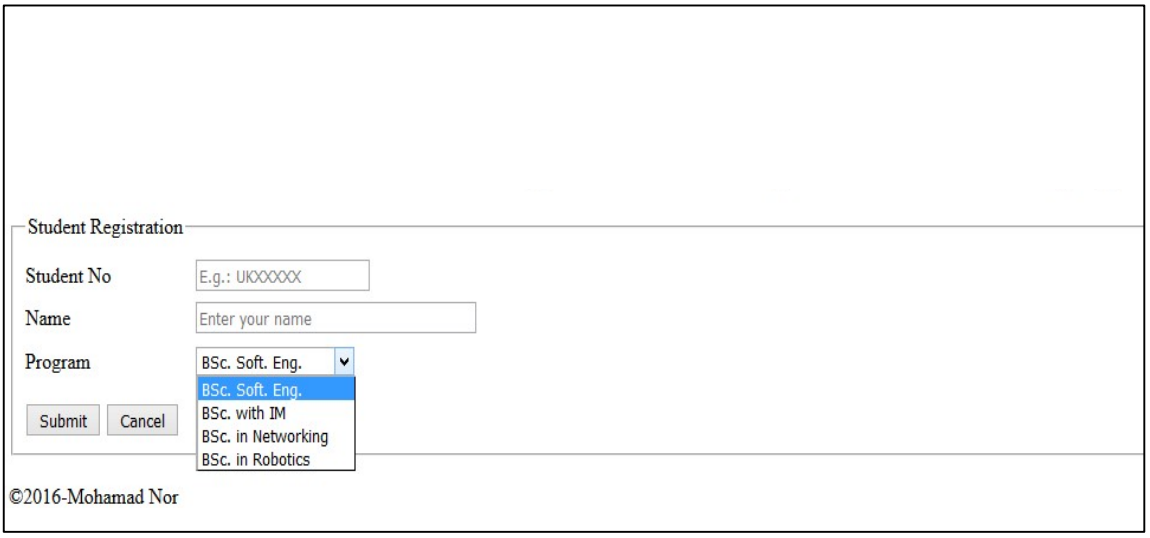
   b. The first field must be started with captain letters then numbers input.

   (Use the following regular expression in *Book JavaBeans* file)



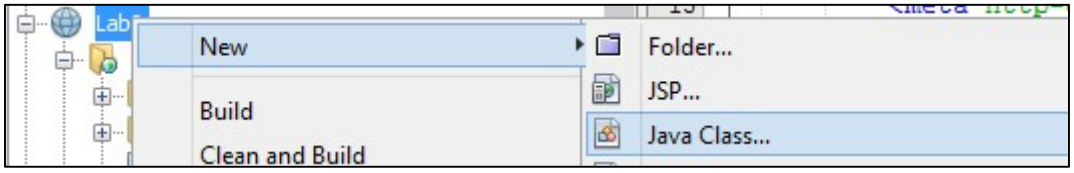
   c. Create a *Submit* button and *Cancel* button.

3. Produce the following output;



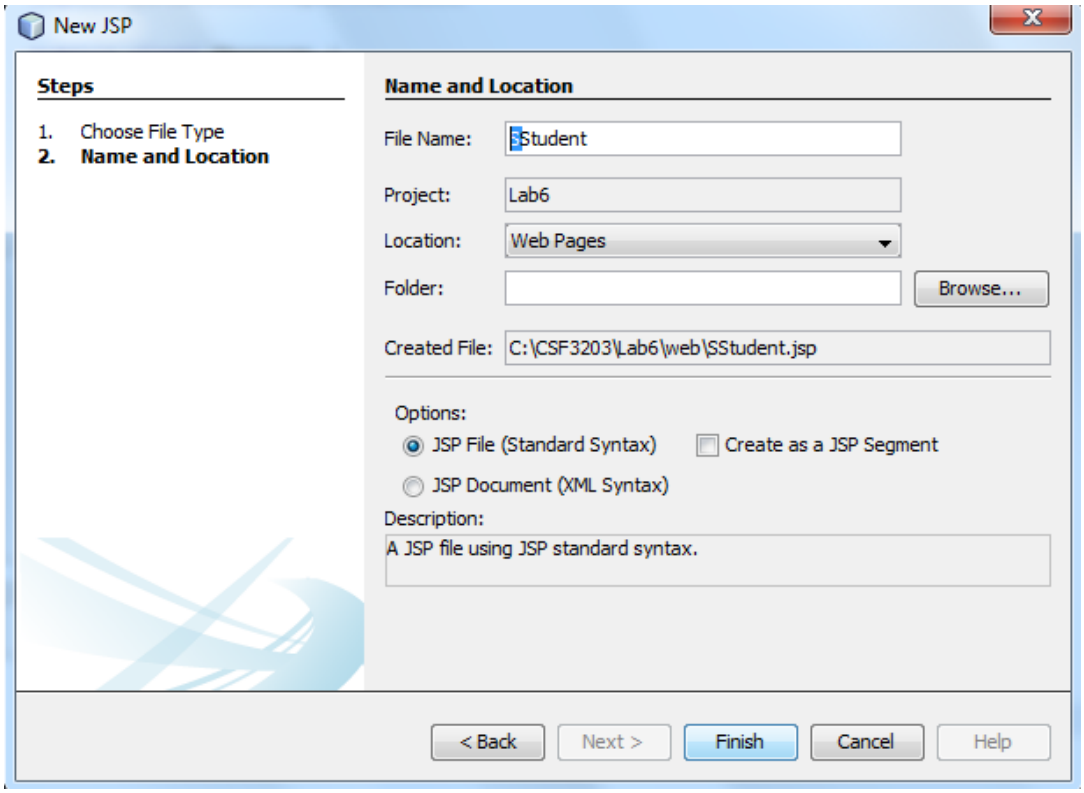
Step 3 - Create *Book JavaBeans*

1. Go to *Lab6* project folder.
2. Right click -> New -> Java Class



3. Click Java Class

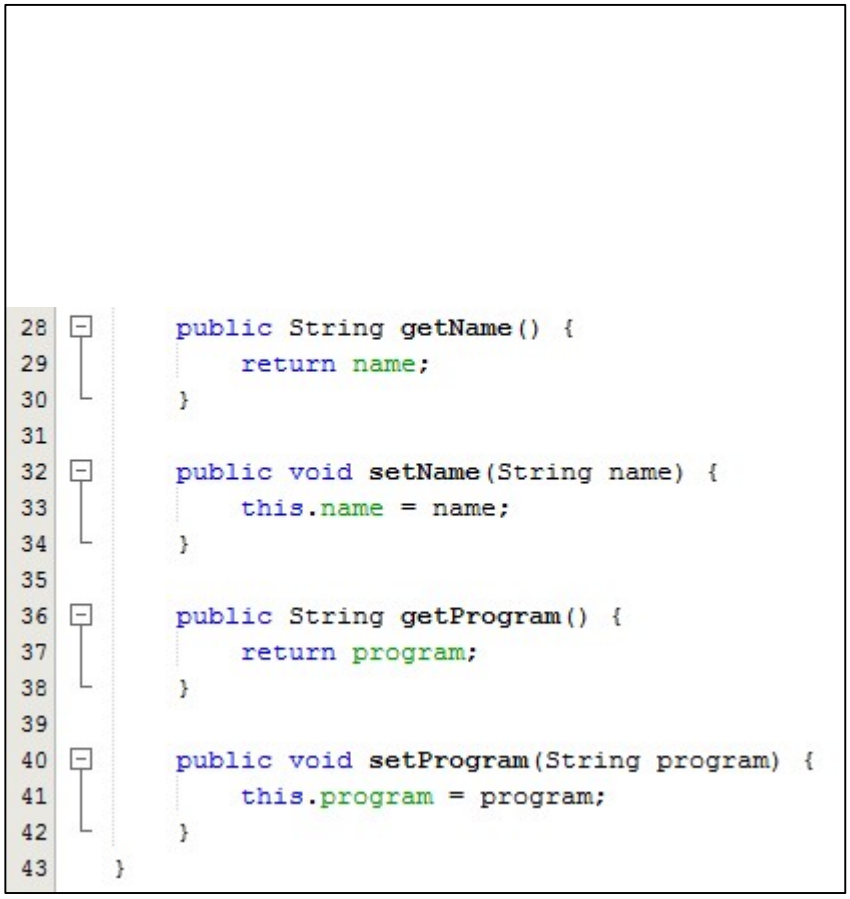
4. Rename Class Name as *Book* and package *as lab6.com*.



5. Define **THREE (3)** instance variables for *Book* class.

```
6   package lab9.com;
7
8   /**
9    *
10   * @author mnor
11   */
12  public class Student
13  {
14      //Create attributes...
15      private String stuno;
16      private String name;
17      private String program;
18  }
```

6. Define the *getter* and *setter* method for corresponding attributes.



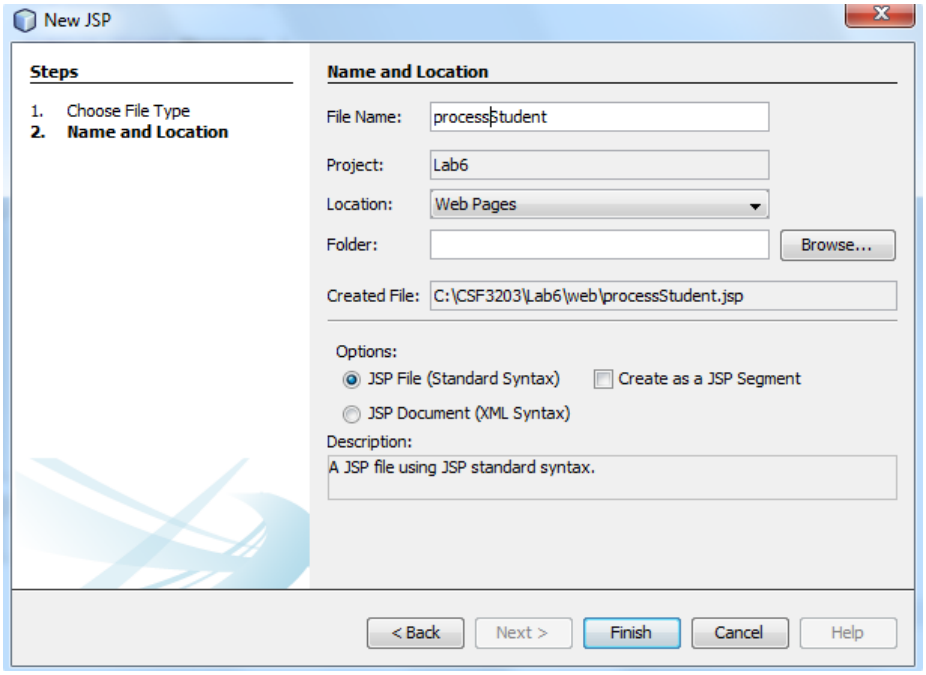
```
28      public String getName() {
29          return name;
30      }
31
32      public void setName(String name) {
33          this.name = name;
34      }
35
36      public String getProgram() {
37          return program;
38      }
39
40      public void setProgram(String program) {
41          this.program = program;
42      }
43  }
```

7. Define *getter* and *setter* method plus regular expression for *stuno* attribute.

```
public String getStuno() {
    Pattern pt = Pattern.compile("[A-Z0-9]*");
    Matcher mt = pt.matcher(stuno);
    boolean bl = mt.matches();
    if (bl == true) {
        valid = stuno;
    } else {
        valid = invalid;
    }
    return valid;
}
public void setStuno(String stuno) {
    this.stuno = stuno;
}
```

Step 4 - Create *processBook.jsp* to insert a record into the database

1. Create a new JSP's page for and rename as *processStudent*.



2. Add the page directive to *processStudent.jsp* page.

```
1  <%--
2      Document   : processStudent
3      Created on : 27-Apr-2016, 15:38:30
4      Author     : Mohamad Nor Hassan
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <%@page language="java"%>
9  <%@page import="java.sql.*"%>
10 <%@page errorPage="errorStudent.jsp" %>
```

3. Create a *Student*'s object using JSP Standard Action tag.

```
17      <!-- Create an object for Student-->
18      <jsp:useBean id="myStudent" class="Lab6.com.Student" scope="request"/>
19
```

4. Assign data entry from page *insertStudent.jsp* page into Student's bean.

```
23      <!-- Assign  data entry from page insertStudent.jsp page into Student's bean-->
24      <jsp:setProperty name="myStudent" property="*"/>
```

5. Load the database driver and create a connection to the database.

```
<%
    int result;

   //Step 1: Load JDBC driver...
     Class.forName("com.mysql.jdbc.Driver");
     System.out.println("Step 1: MySQL driver loaded...!");

   //Step 2: Establish the connection...
     String myURL = "jdbc:mysql://localhost/csf3107";
     Connection myConnection = DriverManager.getConnection(myURL, "root", "admin");
     System.out.println("Step 2: Database is connected...!");
```

6. Create a *PreparedStatement*'s object.

```
  //Step 3: Create a PreparedStatement object...
    System.out.println("Step 3: Prepared Statements created...!");

  //Prepared SQL Query as a String...
    String sInsertQry = "INSERT INTO Student(stuno, stuname, stuprogram) VALUES(?, ?, ?)" ;
    System.out.println("\tSQL Query: " + sInsertQry);

  //Call method preparedStatement
    PreparedStatement myPS = myConnection.prepareStatement(sInsertQry);

  //Assign each value to respective columns for Book's table... (C-Create)
    System.out.println("Step 4: Perform insertion of record...!");
    myPS.setString(1, myStudent.getStuno());
    myPS.setString(2, myStudent.getName());
    myPS.setString(3, myStudent.getProgram());
```

7. Execute the query and display the result.

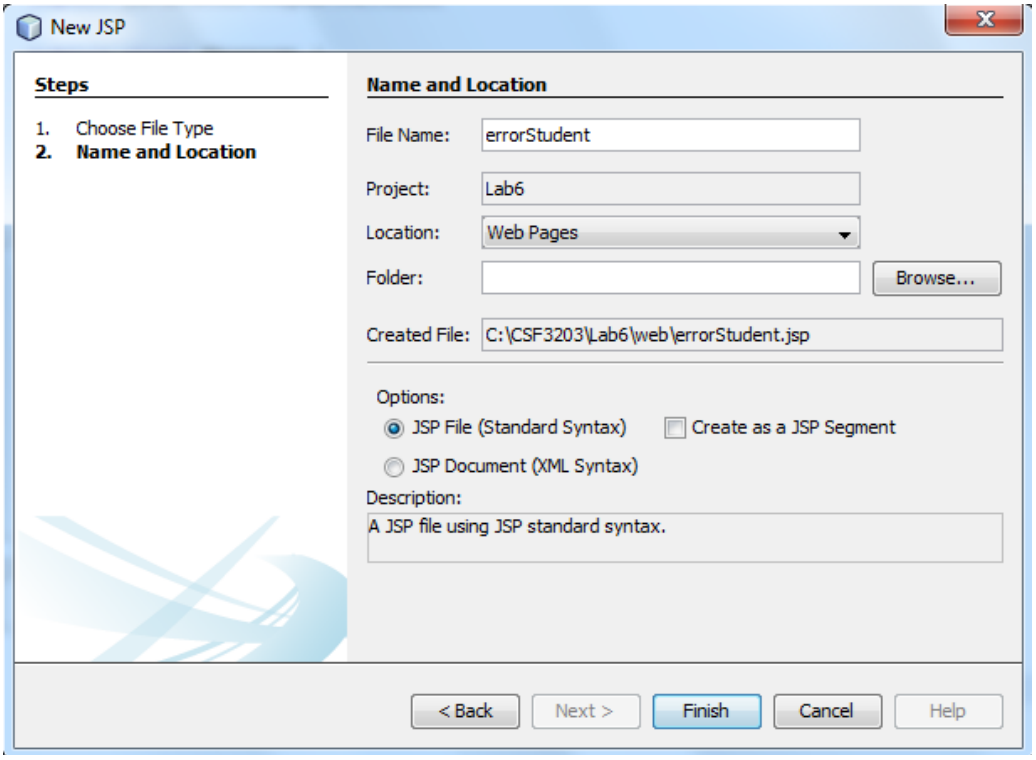
```
  //Step 4: Execute the query...
    result = myPS.executeUpdate();
    if ( result > 0 )
    {
     System.out.println("\tRecord successfully added into Book's table...!");
     out.print("<p>" + "Record with student no " + myStudent.getStuno() +
               " successfully created..!" + "</p>");
     out.print("<p>" + "Details of record are; " + "</p>");
     out.print("<p>Student ID : " + myStudent.getStuno() + "</p>");
     out.print("<p>Name : " + myStudent.getName() + "</p>");
     out.print("<p>Program  : " + myStudent.getProgram() + "</p>");
    }
```

8. Close database connection.

```
    //Step 5: Close database connnection...!
      System.out.println("Step 5: Close database connection...!");
      myConnection.close();
      System.out.println(" ");
      System.out.println("Database connection is closed...!");
%>
```

Step 5 - Create *errorBook.jsp* to display any error message

1. Create new JSP's file and rename as *errorStudent.jsp*.



2. Define the page directive to declare that this is an error page.

3. Complete remaining of code.

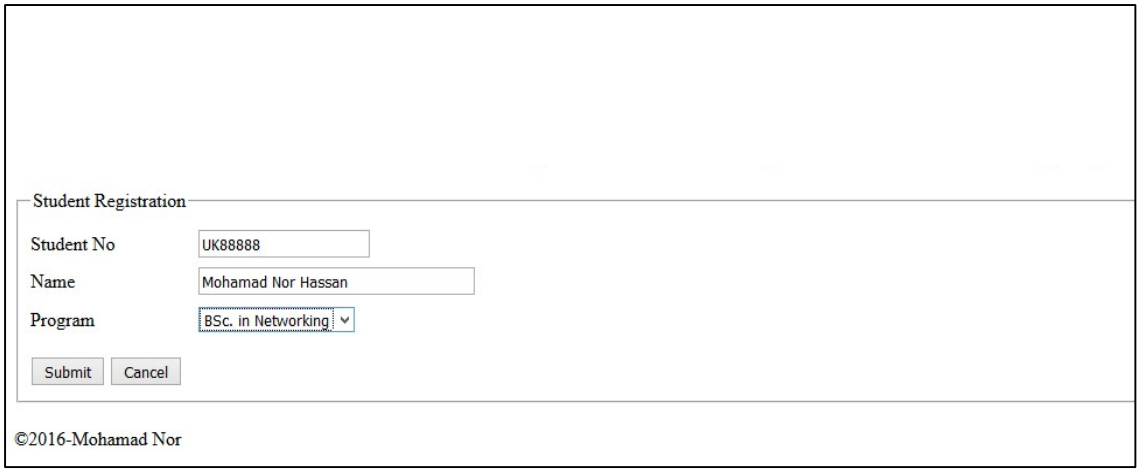
```
15      </head>
16      <body>
17          <form id="errorFrm" action="insertStudent.jsp" method="post">
18              <h1>Lab 9 - Task 1 - Perform creating and retrieving records via JSP page</h1>
19              <                 : when inserting record...!</p>
20              <p><jsp:expression> exception.getMessage() </jsp:expression></p>
21              <br>
22          </form>
23      </body>
24  </html>
```

4. Save and compile *errorStudent.jsp*'s file

Step 6 - Running the program and create a new database

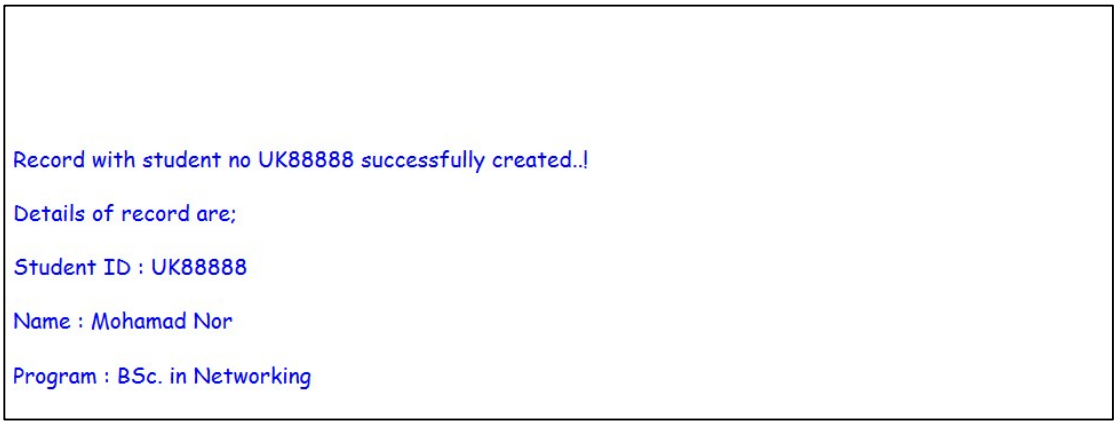
1. Run *insertStudent.jsp* page.

2. Key-in the record.



3. Click *Submit* button.

4. The record will save in the database, and user get a notification.



Reflection

1. What have you learnt from this exercise?

2. Define step by step before you successfully perform the transaction in a database.

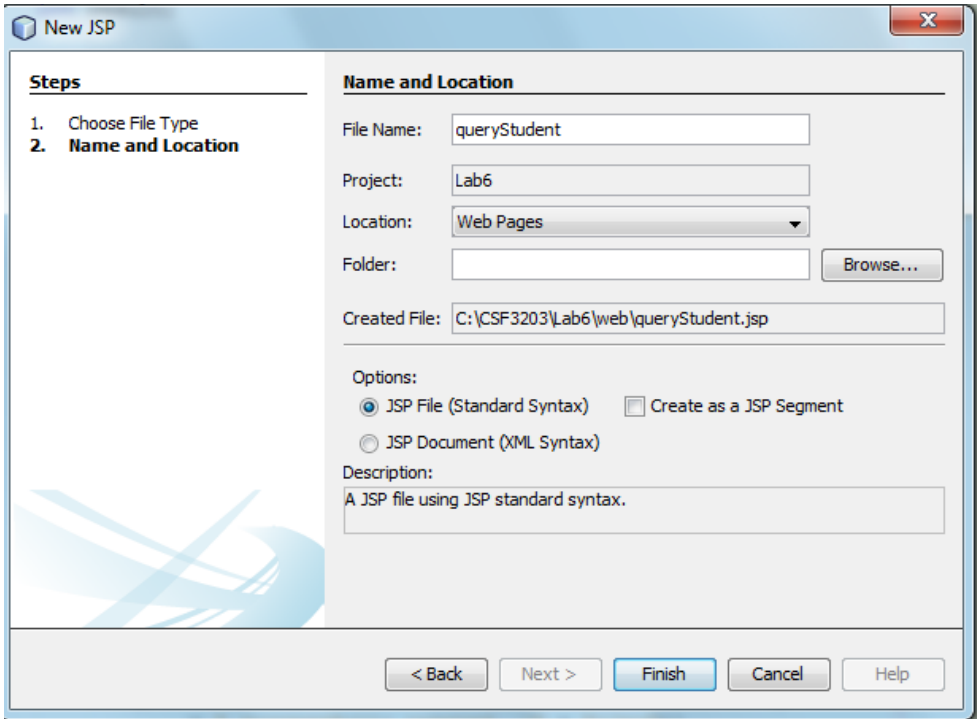
Task 4: Perform Retrieving Records Via JSP Page

Objective:	Use Java Scriptlet to query a list of records.
Problem Description:	Retrieve student records and populate in the table.
Estimated time:	30 minutes

1. Run program *insertStudent.jsp* from Task 3.
2. Insert the following records;

UK12489	Ahmad Salam	BSc with IM
UK56789	Rosnah Azman	BSc Soft. Eng.
UK67342	Liew Cheng Huat	Bsc in Robotics

3. Go to *Lab6*'s project.
4. Create a new JSP's file.
5. Ke-in file name as *queryStudent*.



6. Rename title as Lab 6 - *Task 3*.

7. Rename <*h1*> as Lab 6 - *Task 4 : Retrieving record vis JSP page*.

8. Use JSP page directive to include the information such as content type, and use Java SQL API.

```
<%--
    Document   : queryStudent
    Created on : 27-Apr-2016, 18:01:17
    Author     : Mohamad Nor Hassan
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
```

9. Use a Java scriptlet to create a simple structure of HTML table.

```
        <%
            out.print("<table>");
                out.print("<thead>");
                    out.print("<tr>");
                        out.print("<th>" + "ISBNNo" + "</th>");
                        out.print("<th>" + "Author" + "</th>");
                        out.print("<th>" + "Title" + "</th>");
                    out.print("</tr>");
                out.print("</thead>");
                out.print("<tbody>");
        %>
```

10. Then, load the database driver and connect into the database.

```
<%
    //Step 1: Load JDBC driver...
      Class.forName("com.mysql.jdbc.Driver");
      System.out.println("Step 1: MySQL driver loaded...!");

    //Step 2: Establish the connection...
      String myURL = "jdbc:mysql://localhost/csf3107";
      Connection myConnection = DriverManager.getConnection(myURL, "root", "admin");
      System.out.println("Step 2: Database is connected...!");
```

11. Create Statement for the query.

```
//Step 3: Create a statement object...
  Statement myStatement = myConnection.createStatement();
```

12. Perform query to retrieve records from the Student's table.

```
//Step 4: Perform retrieve record from Student's table... (R-Retrieve)
String myQuery = "SELECT * FROM student";
ResultSet myResulSet = myStatement.executeQuery(myQuery);
```

13. Fetch the record into HTML's table.

```
while ( myResulSet.next() )
{
   out.print("<tr>");
     out.print("<td width=\"20%\">" + myResulSet.getString(1) + "</td>");
     out.print("<td width=\"40%\">" + myResulSet.getString(2) + "</td>");
     out.print("<td width=\"40%\">" + myResulSet.getString(3) + "</td>");
   out.print("</tr>");
}
```

14. Close the database connection.

```
      //Step 5: Close database connnection...!
        System.out.println("Step 5: Close database connection...!");
        myConnection.close();
        System.out.println(" ");
        System.out.println("Database connection is closed...!");

         out.print("</tbody>");
    out.print("</table>");
%>
```

15. Enhance the CSS for the table.

```
<style>
    table  {
              border-collapse: collapse;
           }

    td, th {
            border: 1px solid #999;
            padding: 0.5rem;
            text-align: left;
            }

          th{
              background:   gold;
             }
</style>
```

16. Save *queryStudent.jsp*

17. Compile and run *queryStudent.jsp*.

18. You should get the following output.

ISBNNo	Author	Title
UK12489	Ahmad Salam	BSc. with IM
UK56789	Rosnah Azman	BSc. Soft. Eng.
UK67342	Liew Cheng Huat	BSc. in Robotics
UK88888	Mohamad Nor	BSc. in Networking

Reflection

1. What have you learnt from this exercise?

2. Explain the differences when using *Statement()* and *PreparedStatement()*.

Task 5: Create A Record Using JSP Model 1

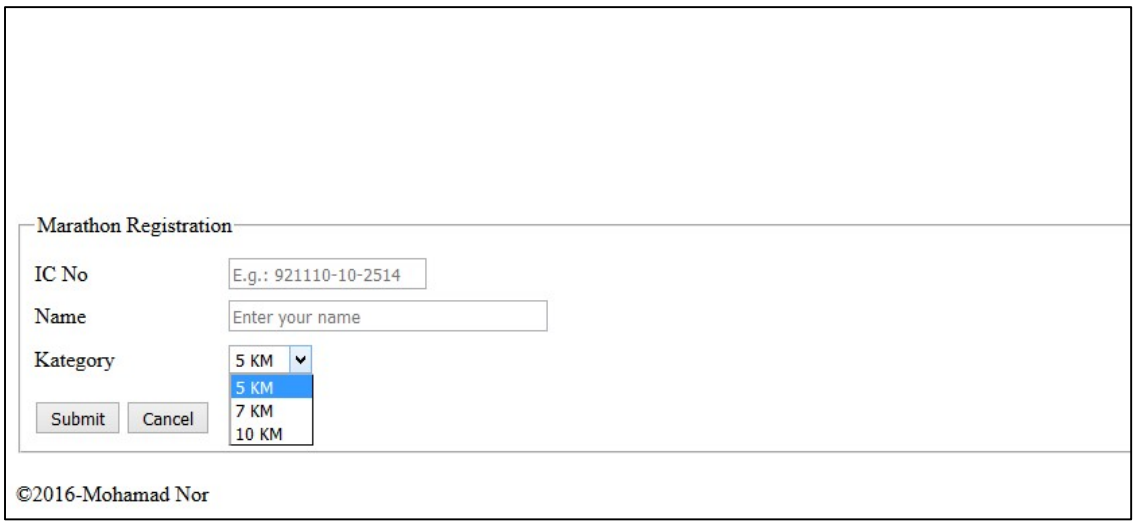
Objective:	Use JavaBeans to perform SQL transaction.
Problem Description:	Create a sample web form to register the Marathon event.
Estimated time:	40 minutes

1. Choose Project *Lab6*.

2. Create a new JSP's file.

3. Type file name as *registerMarathon*.

4. Prepare the following Graphical User Interface (GUI).



5. Create a JavaBeans *Marathon*.

```
/**
 * Bean   : Marathon.java
 * Author : Mohamad Nor Hassan
 * Date   : 27 April 2016
 */
public class Marathon {
    private String icno;
    private String name;
    private String category;

    public String getIcno() {
        return icno;
    }

    public void setIcno(String icno) {
        this.icno = icno;
    }

    public String getName() {
        return name;
    }

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

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }
}
```

6. Create a *Database* class that has two methods; *getConnection()*, and *closeConnection()*

```
package lab9.com;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import lab9.com.Marathon;

/**
 * Bean   : Database.java
 * Author : Mohamad Nor Hassan
 * Date   : 27 April 2016
 */
public class Database {
    private static Connection myConnection = null;
    private static String myURL = myURL = "jdbc:mysql://localhost:3306/csf3107";
    private int result = 0;

    public static Connection getConnection() throws ClassNotFoundException {

        if (myConnection != null) {
            return myConnection;
        }
        else try {

            Class.forName("com.mysql.jdbc.Driver");
            myConnection = DriverManager.getConnection(myURL, "root", "admin");
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
        return myConnection;
    }

    public void closeConnection() throws ClassNotFoundException
    {
        try {
            myConnection.close();
        }
        catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

7. Create a *MarathonDAO* class to perform SQL transaction for business object *Marathon* and store it into *package lab6.com*.

```
2
3   import java.sql.Connection;
4   import java.sql.PreparedStatement;
5   import java.sql.SQLException;
    import lab9.com.Database;
7
8   /**
9    * Bean   : MarathonDAO.java
10   * Author : Mohamad Nor Hassan
11   * Date   : 27 April 2016
12   */
13  public class MarathonDAO
14  {
        private Connection connection;
16      private int result = 0;
17      public MarathonDAO() throws ClassNotFoundException
18      {
19          connection = Database.getConnection();
20      }
```

```
    public int addDetails (Marathon marathon)
    {
       try {
            String mySQL = "INSERT INTO marathon(icno, name, category) values (?, ?, ?)";
            PreparedStatement preparedStatement = connection.prepareStatement(mySQL);

            System.out.println("IC No    = " + marathon.getIcno());
            System.out.println("Name     = " + marathon.getName());
            System.out.println("Category = " + marathon.getCategory());

            //Parameters
            preparedStatement.setString(1, marathon.getIcno());
            preparedStatement.setString(2, marathon.getName());
            preparedStatement.setString(3, marathon.getCategory());
            result = preparedStatement.executeUpdate();

       } catch (SQLException e) {
           e.printStackTrace();
       }
        return result;
    }
}
```

8. Create a new file name known as *processMarathon.jsp*.

9. Import related classes in *package lab6.com*.

```
    Document   : processMarathon
    Created on : 27-Apr-2016, 19:15:15
    Author     : Mohamad Nor Hassan
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="lab9.com.Database"%>
<%@page import="lab9.com.Marathon"%>
<%@page import="lab9.com.MarathonDAO"%>
```

10. Instantiate an object *Marathon*.

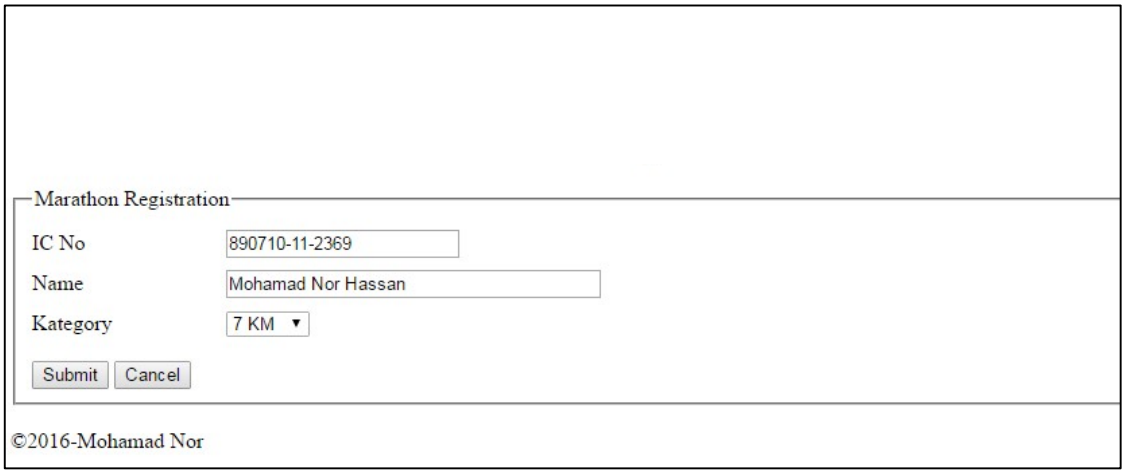
```
<!-- Create an object for Marathon-->
<jsp:useBean id="myMarathon" class="lab6.Marathon" scope="request"/>
```

11. Create a Java Scriptlet to invoke respective object for inserting record in *marathon*'s table.

```
<%
   int result;

   //Step 1: Create Database object...
     Database myDB = new Database();

     MarathonDAO object1 = new MarathonDAO();

     //Step 2: Add the records...
     result = object1.addDetails(myMarathon);

     //Step 3: Determine whether the transactino is sucess...
     if ( result > 0 )
     {
      System.out.println("\tRecord successfully added into Book's table...!");
      out.print("<p>" + "Record with IC No " + myMarathon.getIcno() +
                " successfully created..!" + "</p>");
      out.print("<p>" + "Details of record are; " + "</p>");
      out.print("<p>Ic No    : " + myMarathon.getIcno() + "</p>");
      out.print("<p>Name     : " + myMarathon.getName() + "</p>");
      out.print("<p>Category : " + myMarathon.getCategory() + "</p>");
     }

  //Step 4: Close database connnection...!
     System.out.println("Step 5: Close database connection...!");
     myDB.closeConnection();
     System.out.println(" ");
     System.out.println("Database connection is closed...!");
%>
```

12. Compile and save *processMarathon.jsp*.

13. Run *registerMarathon.jsp* and key-in related record.



14. The output will appear in a web browser.

Reflection

1. What have you learnt from this exercise?

2. Describe the benefits of using JavaBeans.

Exercise

Implement user login

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
   - **password** as a character length 10
   - **firstname** as varchar(50)
   - **lastname** as varchar(50)

2. Create **insertUser.html** as the main interface to register a new user.

3. Create **processUser.jsp** page to process the record.

4. Create **login.jsp page** to login to the system.

5. Create **doLogin.jsp** to validate username and password. If validation is successful, redirect the page to **main.jsp** page that displays the username, firstname and lastname.

6. If validation is unsuccessful, redirect the page to login.jsp with message 'Invalid username or password..!'