

LAB 6: Database Programming in Servlets

Lab Activities

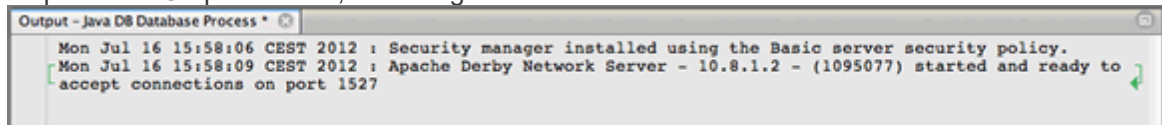
1. Create a database and a table in Java Derby

Task 1: Starting the Server and Creating a Database

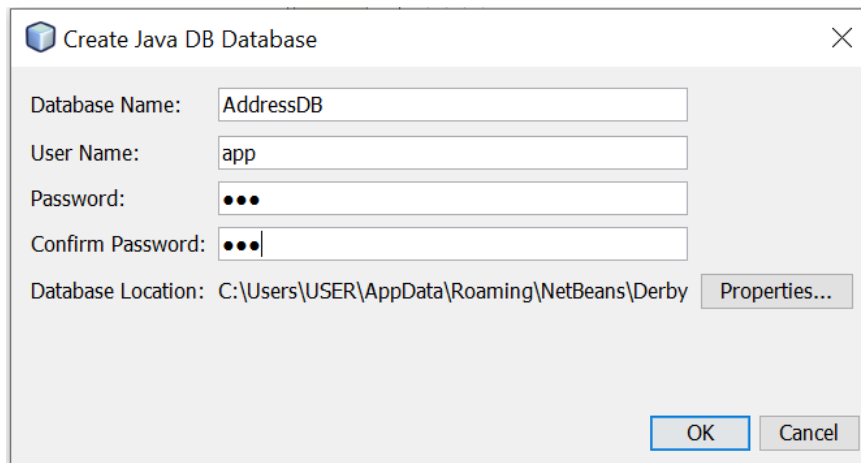
The Java DB Database menu options are displayed when you right-click the Java DB node in the Services window. This contextual menu items allow you to start and stop the database server, create a new database instance, as well as register database servers in the IDE (as demonstrated in the previous step).

To start the database server:

- In the **Services** window, right-click the **Java DB** node and choose **Start Server**. Note the following output in the Output window, indicating that the server has started:




- Right-click the **Java DB** node and choose **Create Database** to open the Create Java DB Database dialog.
- Type **AddressDB** for the Database Name.
- Type **app** for the User Name and Password. Click OK.

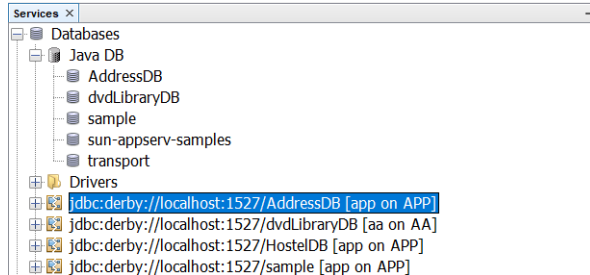



After you create the database, if you expand the Databases node in the Services window you can see that the IDE created a database connection and that the database was added to the list under the Java DB node.

Task 2: Connecting to the Database

In order to begin working with the **AddressDB** database, you need to create a connection to it. To connect to the **AddressDB** database perform the following steps.

- Expand the Databases node in the Services window and locate the new database and the database connection nodes. The database connection node () is displayed under the Databases node. The name of the database is displayed under the Java DB node.



- Right-click the **AddressDB** database connection node (jdbc:derby://localhost:1527/AddressDB [app on APP]) and choose Connect. The connection node icon appears whole (), signifying that the connection was successful.
- Create a convenient display name for the database by right-clicking the database connection node (jdbc:derby://localhost:1527/AddressDB [app on APP]) and choosing Rename. Type AddressDB in the text field and click OK.

Task 3) Creating Tables

The **AddressDB** database that you just created is currently empty. It does not yet contain any tables or data. In NetBeans IDE you can add a database table by either using the Create Table dialog, or by inputting an SQL statement and running it directly from the SQL Editor. In this lab, you will use the create table dialog.

- Expand the **AddressDB** connection node and note that there are several schema subnodes. The app schema is the only schema that applies to this tutorial. Right-click the APP node and choose Set as Default Schema.
- Expand the APP node and note that there are three subfolders: Tables, Views and Procedures. Right-click the Tables node and choose Create Table to open the Create Table dialog box.
- In the Table Name text field, type **ADDRESS**.
- Click Add Column. The Add Column dialog box appears. For Column Name, enter **firstname**. For Data Type, select **VARCHAR** from the drop-down list.
- Repeat this procedure now by specifying fields as shown in the table below:


Key	Index	Null	Unique	Column name	Data type	Size
		[checked]		firstName	VARCHAR	20
		[checked]		lastName	VARCHAR	20
		[checked]		mi	CHAR	5
		[checked]		street	VARCHAR	40
		[checked]		city	VARCHAR	20
		[checked]		state	VARCHAR	20
		[checked]		postcode	VARCHAR	8
		[checked]		telephone	VARCHAR	12
		[checked]		email	VARCHAR	30

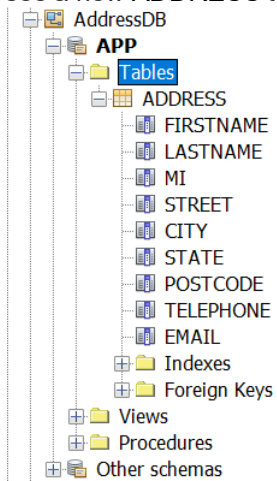
Create Table

Table name: ADDRESS

Key	Index	Null	Unique	Column name	Data type	Size	Scale
		<input checked="" type="checkbox"/>		firstname	VARCHAR	20	
		<input checked="" type="checkbox"/>		lastname	VARCHAR	20	
		<input checked="" type="checkbox"/>		mi	VARCHAR	5	
		<input checked="" type="checkbox"/>		street	VARCHAR	40	
		<input checked="" type="checkbox"/>		city	VARCHAR	20	
		<input checked="" type="checkbox"/>		state	VARCHAR	20	
		<input checked="" type="checkbox"/>		postcode	VARCHAR	8	
		<input checked="" type="checkbox"/>		telephone	VARCHAR	12	
		<input checked="" type="checkbox"/>		email	VARCHAR	30	

OK Cancel Help

- f. When you are sure that your Create Table dialog contains the same specifications as those shown above, click OK. The IDE generates the ADDRESS table in the database, and you can see a new ADDRESS table node () display under the Tables node.



2. Creating web components.

Task 1: Create a web project

- Create the DBServlet Web application project with the following characteristic:
 Project: DBExample
 Location: C:\Users\User\Documents\NetBeansProjects
 Folder: C:\Users\User\Documents\NetBeansProjects\DBExample
 Server : Personal glassfish
 Java EE version: Java EE 7
 Contextpath: /DBExample

Task 2: Create an HTML file

- Create an HTML file name **SimpleRegistration.html** for collecting the data and sending it to the database using the post method.

```

1 <html>
2 <head>
3 <title>Simple Registration without confirmation</title>
4 </head>
5 <body>
6 Please register to your instructor's student address book
7 <form method = "post" action = "/DBExample/SimpleRegistration">
8 <p> Last Name <font color = "#FF0000">*</font>
9 <input type = "text" name = "lastName">&nbsp;
10 First Name <font color = "#FF0000">*</font>
11 <input type = "text" name = "firstName">&nbsp;
12 MI <input type = "text" name = "mi" size = "3">
13 </p>
14 <p>Telephone
15 <input type="text" name="telephone" size = "20">&nbsp;
16 Email
17 <input type="text" name="email" size = "30">&nbsp;
18 </p>
19 <p>Street <input type = "text" name="street" size = "40"></p>
20 <p>
21 City <input type = "text" name = "city" size = "20">&nbsp;
22 Postcode <input type = "text" name = "postcode" size = "6">
23 </p>
24 <p> State
25 <select size = "1" name = "state">
26 <option value="Selangor">Selangor</option>
27 <option value="Perak">Perak</option>
28 <option value = "Kedah">Kedah</option>
29 <option value = "Melaka">Melaka</option>
30 <option value = "Johor">Johor</option>
31 <option value = "Negeri Sembilan">Negeri Sembilan</option>
32 </select>&nbsp;
33 </p><br>
34 <input type = "submit" value = "Submit">
35 <input type = "reset" value = "Reset">
36 </form>
37 <p><font color="#FF0000">* Required fields</font></p>
38 </body>
39 </html>

```

- Save the file in Web Pages folder.

Task 3: Create a Servlet

- a. Create a Java Servlet with the following characteristic: Class name: **SimpleRegistration**

Project: DBExample

Location:Source Package Package: default

package Servlet name: SimpleRegistration

URL Pattern : /SimpleRegistration

Initialization Parameters: None

Tick (✓) Add information to deployment descriptor (web.xml)

- b. Add the following code in the SimpleRegistration Servlet.

```

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  import java.sql.*;
8
9  public class SimpleRegistration extends HttpServlet {
10     //Use a prepared statement to store a student data into database
11     private PreparedStatement pstmt;
12     private Connection conn; //use Connection statement to connect to database
13
14     public void init() throws ServletException{
15         initializeJdbc();
16     }
17     @Override
18     protected void doPost(HttpServletRequest request, HttpServletResponse response)
19         throws ServletException, IOException {
20         response.setContentType ("text/html") ;
21         PrintWriter out = response.getWriter();
22
23         //obtain parameter from the client
24         String lastName = request.getParameter("lastName");
25         String firstName = request.getParameter("firstName");
26         String mi = request.getParameter("mi");
27         String phone = request.getParameter("telephone");
28         String email = request.getParameter("email");
29         String address = request.getParameter("street");
30         String city = request.getParameter("city");
31         String postcode = request.getParameter("postcode");
32         String state = request.getParameter("state");
33
34         try {
35             if (lastName.length() == 0 || firstName.length() == 0) {
36                 out.println("Last Name and First name are required");
37                 return; // End the method
38             }
39             storeStudent (lastName, firstName, mi, phone, email, address,
40                 city, state, postcode);
41             out.println(firstName+" "+lastName+" is now registered in the database");
42         } catch (Exception ex) {
43             out.println("Error: " + ex.getMessage());
44         } finally {
45             out.close(); //close stream
46         }
47     }

```

```

48
49 private void initializeJdbc(){
50     try {
51         //Declare driver and connection string
52         String driver = "org.apache.derby.jdbc.ClientDriver";
53         String connectionString = "jdbc:derby://localhost:1527/AddressDB";
54         String usr="app", pass="app";
55
56         //Load the driver
57         Class.forName(driver);
58
59         //Connect to the sample database
60         conn = DriverManager.getConnection(connectionString,usr,pass);
61
62         } catch (Exception ex){
63             ex.printStackTrace();
64         }
65     }
66
67     /**
68     * Store a student record to the database
69     */
70     private void storeStudent(String lastName, String firstName, String mi,
71         String phone,String email,String address,
72         String city,String state,String postcode) throws SQLException {
73         // create a statement
74         String sql = "insert into Address "
75             + "(lastname, firstname,mi,telephone,email,street,city, "
76             + "state,postcode) values (?, ?, ?, ?, ?, ?, ?, ?)";
77         pstmt = conn.prepareStatement(sql);
78         pstmt.setString(1, lastName);
79         pstmt.setString(2, firstName);
80         pstmt.setString(3, mi);
81         pstmt.setString(4, phone);
82         pstmt.setString(5, email);
83         pstmt.setString(6, address);
84         pstmt.setString(7, city);
85         pstmt.setString(8, state);
86         pstmt.setString(9, postcode);
87         pstmt.executeUpdate();
88     }

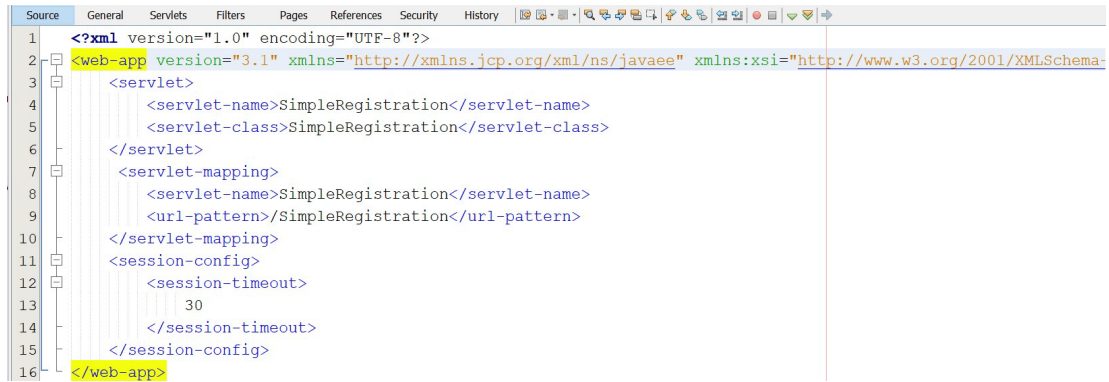
```

- c. Compile the SimpleRegistration Servlet.

3. Configuring the Web application

Task 1 : Verify the view servlet configuration

- a. Open the web.xml deployment descriptor in the Web Pages – Web INF folder.
- b. View the definition and the mapping for the **SimpleRegistration** servlet.



Task 2: Create the homepage

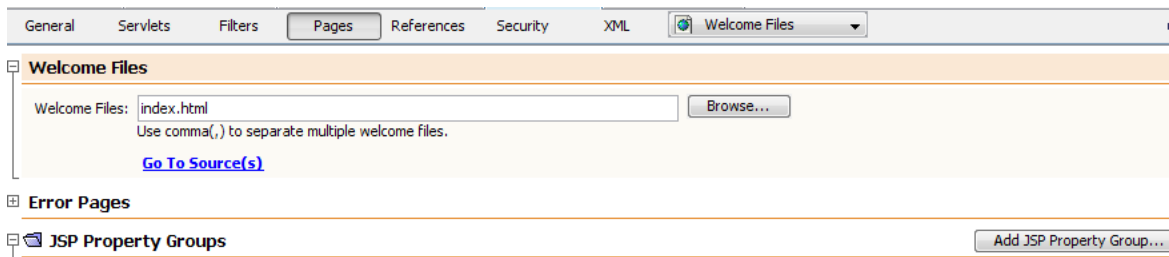
- Create an HTML file with the following characteristic:
Project : DBExample
HTML file name: index
Location:Web Pages
- Edit the index.html file so that it displays the following text.

Register new student

Edit the index.html file to add a link to the SimpleRegistration.html page.

```
<html>
<head>
  <title>Home Page</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <div><a href="SimpleRegistration.html">Register new student</a></div>
</body>
</html>
```


- Open the web.xml deployment descriptor and set index.html as the welcome file.



Task 3: Deploy the web application

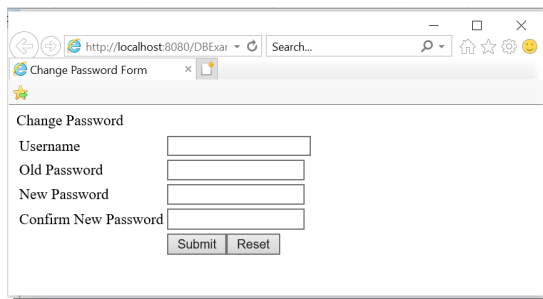
- Build the DBExample web application. Correct any errors you encounter
- Deploy the DBExample web Application
- Run the web application project
- Click the register new student link to register student.

Postlab Exercise

Write a servlet that enables the user to change the password from an HTML form, as shown in the **Figure 1** below. Suppose that the user information is stored in a table named Account with three columns, username, password and name, where name is the real name of the user (data can be inserted manually by right click at **Account** table→**View Data**→then click **Insert Records** icon  in Services tab). The servlet performs the following task:

- Verify all data must be enter by the user. If not, report the error for null entry data and redisplay the HTML form.
- Verify that the username and old password are in the table. If not, report the error (Username & password is incorrect) and redisplay the HTML form.
- Verify that the new password and the confirmed password are the same. If not, report the error (New password and the confirmed password are not same) and redisplay the HTML form.
- If the user information is entered correctly, update the password and report the status of the update to the user, as shown in **Figure 2**.

Note: to report error and redisplay HTML form, please refer form validation in Lab 5 (form validation) and Post Lab 5 (**Task 3: Repopulating Form Data**) as a guide.



A screenshot of a web browser window titled "Change Password Form". The browser's address bar shows "http://localhost:8080/DBExai". The form contains four input fields: "Username", "Old Password", "New Password", and "Confirm New Password". Below the input fields are two buttons: "Submit" and "Reset".

Figure 1

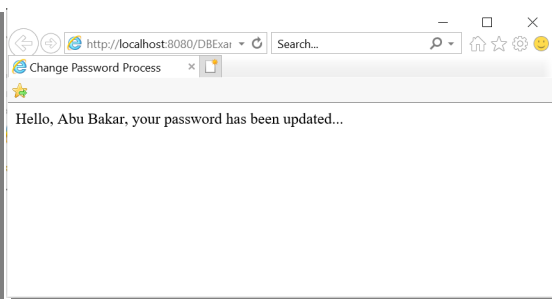


Figure 2