Database Application in JSP using NetBeans

Created by: Siddharth Chatrola

INDEX

1. Basic:-

We make simple JSP application in NetBeans. It contains four jsp files.

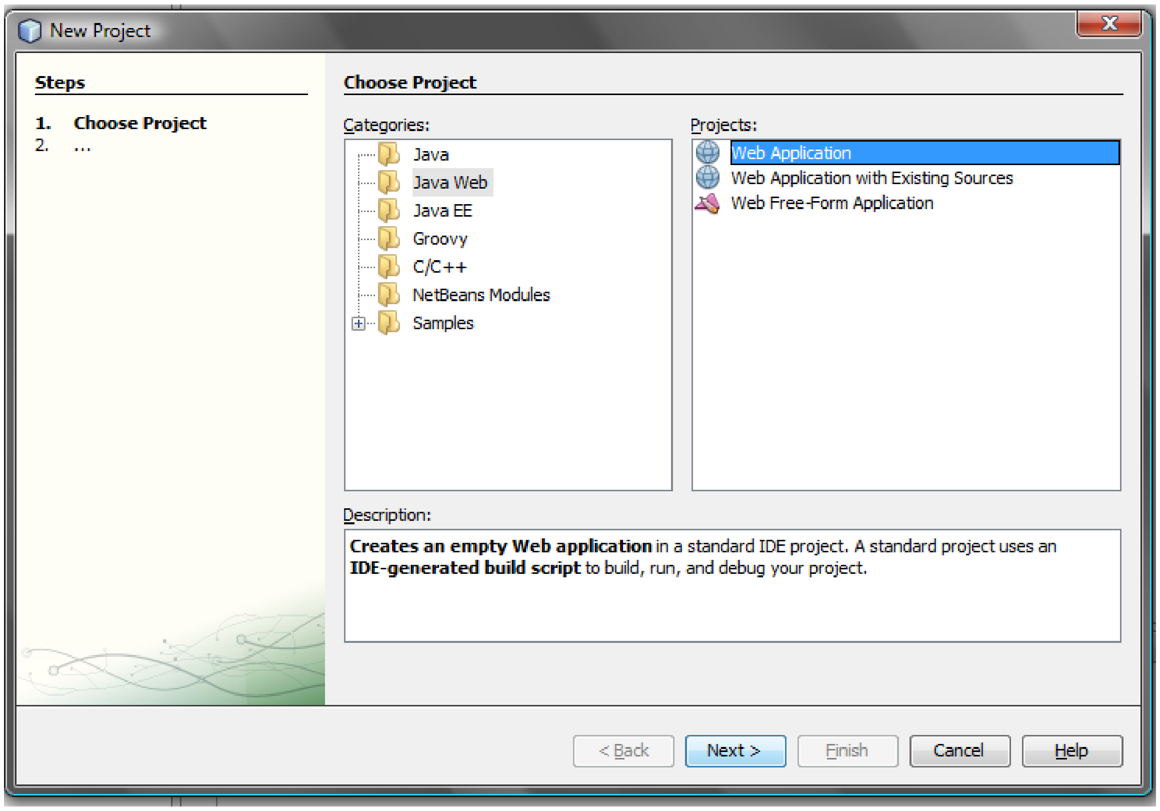
- Index.jsp : contains link for other three pages.
- Insert.jsp : Insert data into STUDENT table and display.
- Update.jsp : Update data of STUDENT table and display.
- Delete.jsp : Delete data from STUDENT table and display.

STUDENT table contains three fields and database is made using JavaDB.

COLUMN NAME	TYPE	DETAIL
ID	VARCHAR(15)	PRIMARY KEY
NAME	VARCHAR(40)	NOT NULL
BRANCH	VARCHAR(40)	NOT NULL

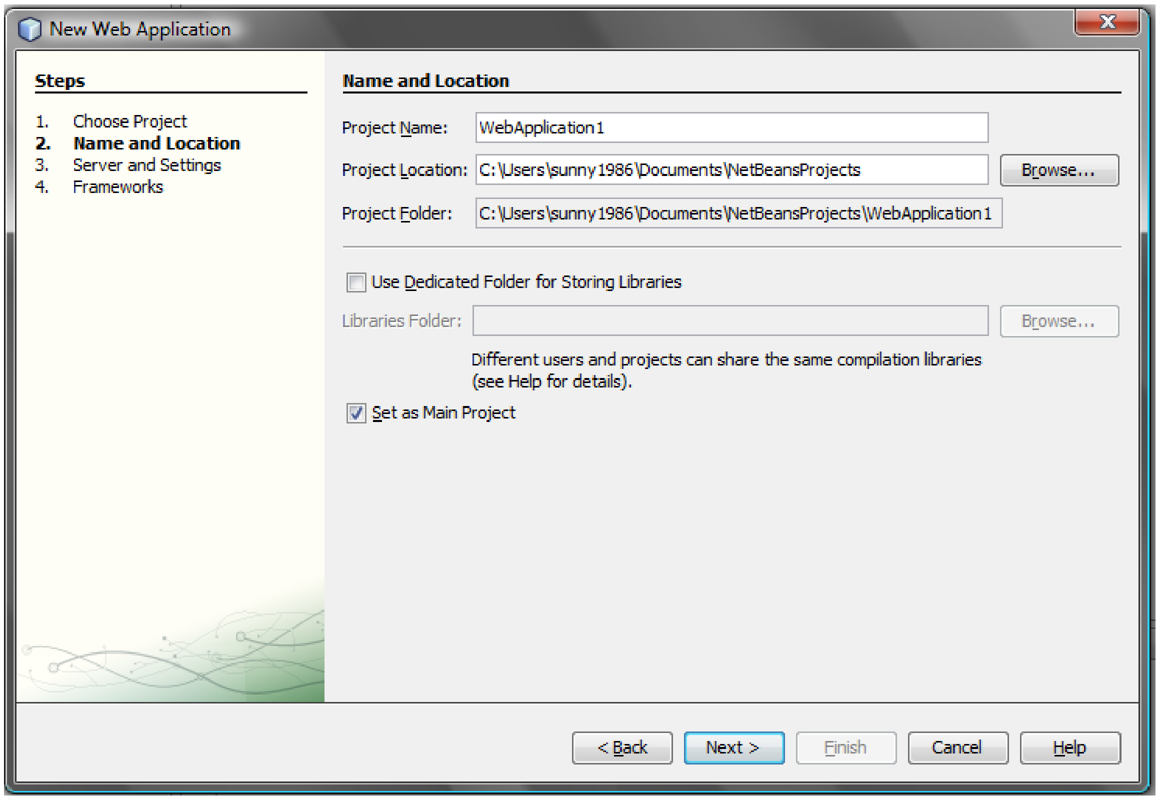
2. Creating application and adding drivers:

- Start NetBeans.
- Go to File -> New project.



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

- You will see the above window. Here from categories choose Java Web.
- It will show you projects as shown in fig in right side tab.
- Select Web Application and give next.
- You will see next window as shown below.


New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name: WebApplication1

Project Location: C:\Users\sunny1986\Documents\NetBeansProjects

Project Folder: C:\Users\sunny1986\Documents\NetBeansProjects\WebApplication1

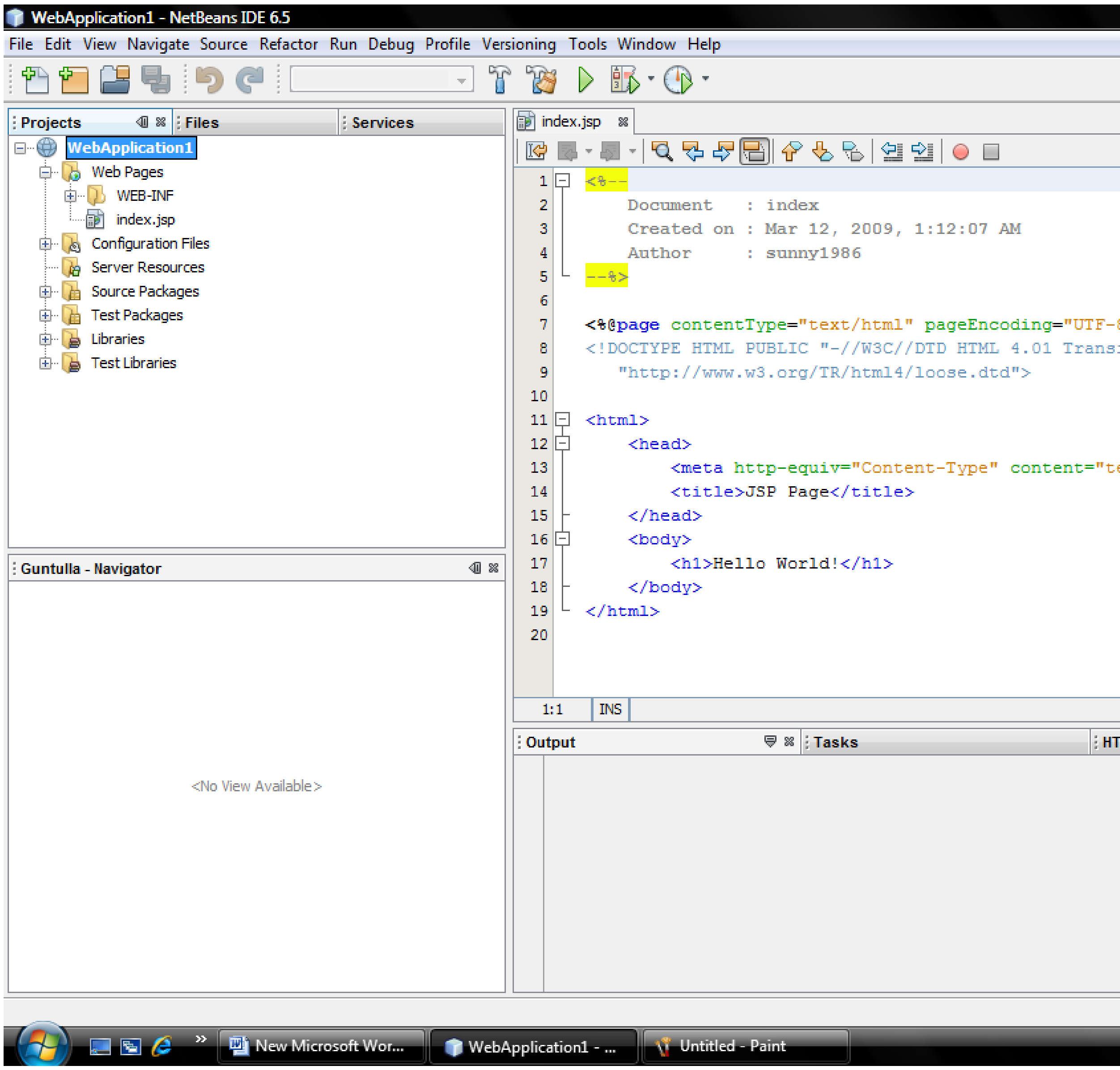
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☑ Set as Main Project

- Here you can specify project name and path where to store project.
- When you give next it will show the web server settings.
- We are using GlassFish as web server.
- After giving next it will show window to choose framework.
- We are not using framework here so don’t select anyone and click finish button. Our project is created.

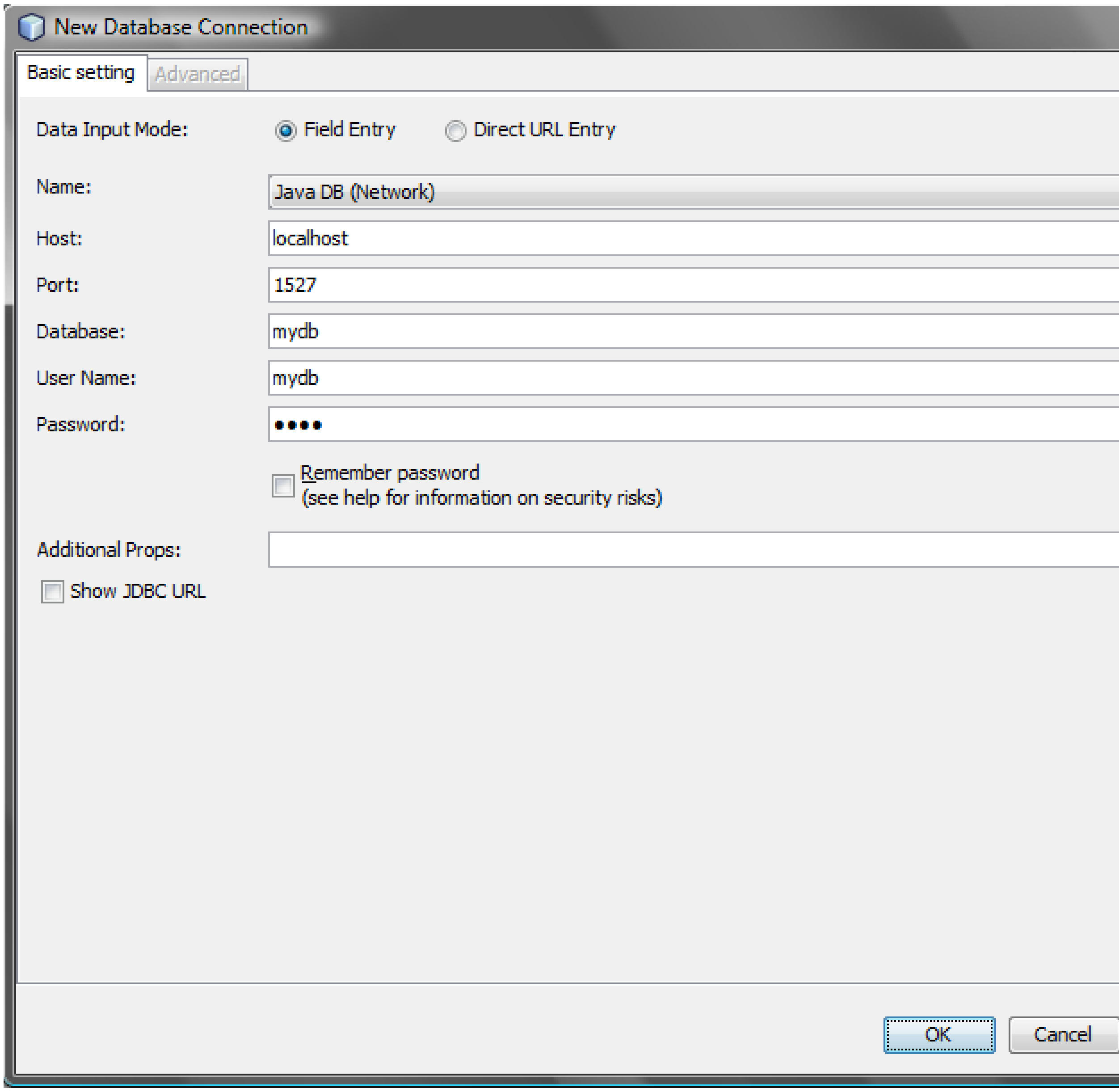


```
<%--
    Document   : index
    Created on : Mar 12, 2009, 1:12:07 AM
    Author     : sunny1986
--%>

<%@page contentType="text/html" pageEncoding="UTF-
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Trans
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="te
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

- Index.jsp page is automatically created by default. Now, you have to create other three JSP pages.
- To do it, in project window write click on Web pages -> New -> JSP.
- It will open new window which ask for page name. Give appropriate name.
- Create Insert.jsp,Update.jsp and Delete.jsp.
- I have used JavaDB as database. So, I have to include driver for that into my library. If you are using other database then you have to include appropriate drivers.
- Right click on Libraries -> Add JAR/Folders….
- It will open a window, browse through your hard disk and select appropriate JAR and click open.
- It will add drivers to access database into your project.

3. Creating Database Connection:

- Go to services tab.
- Right click on Databases and chose New Connection….
- You have to fill database details here.



New Database Connection

Basic setting | Advanced

Data Input Mode: [✓ Field Entry] Direct URL Entry

Name: Java DB (Network)

Host: localhost

Port: 1527

Database: mydb

User Name: mydb

Password: ●●●●

☐ Remember password (see help for information on security risks)

Additional Props:

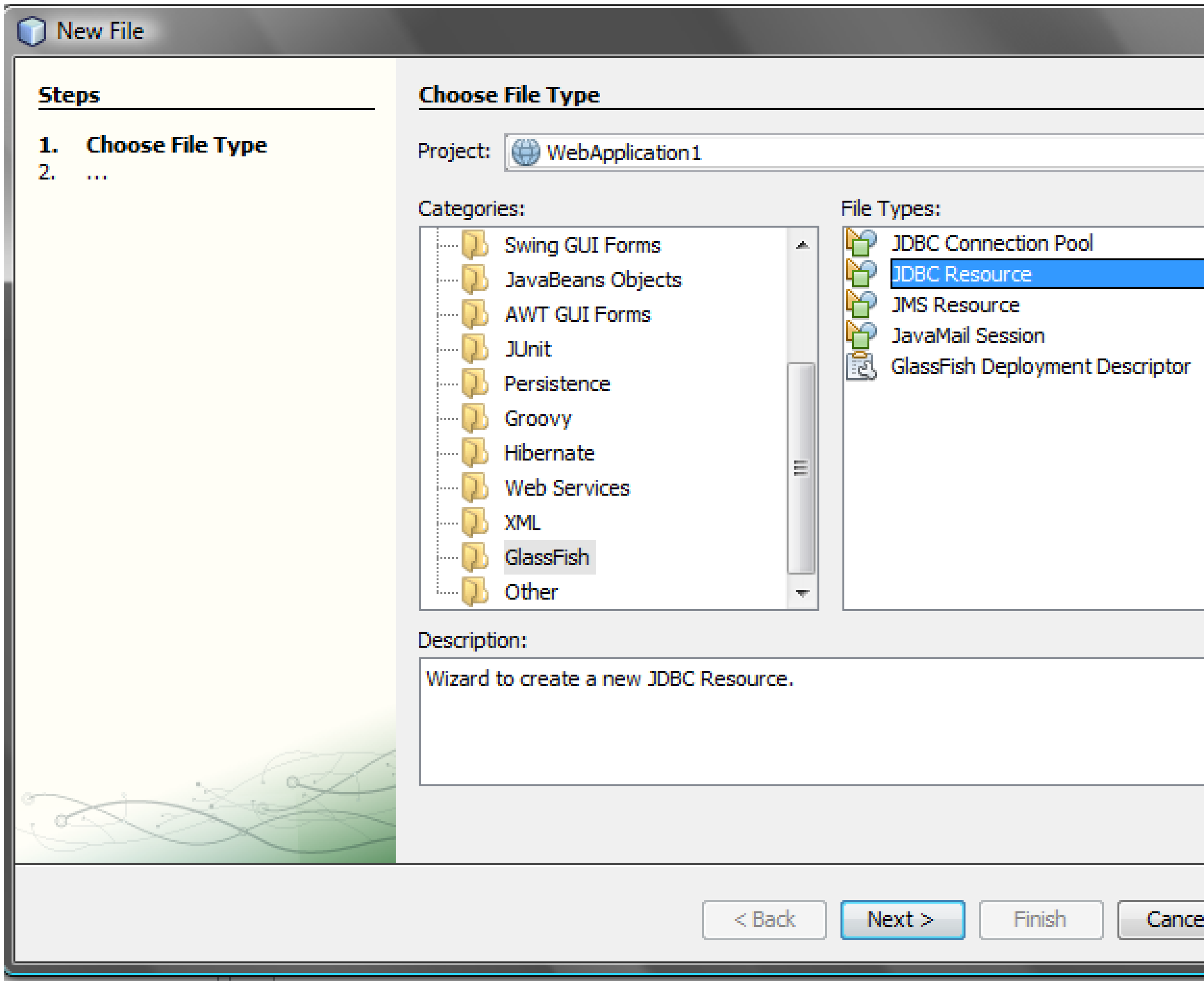
☐ Show JDBC URL

- Details for these fields are as follows.
  - Name:- database type in this case it is JavaDB.
  - Host:- It is the name of computer on which database is resides.
  - Port:- Port number through which you connect to database
  - Database, User Name and Password is self explanatory.

- When you will click ok it will create connection to your database.
- I hope you have created STUDENT table in your database.

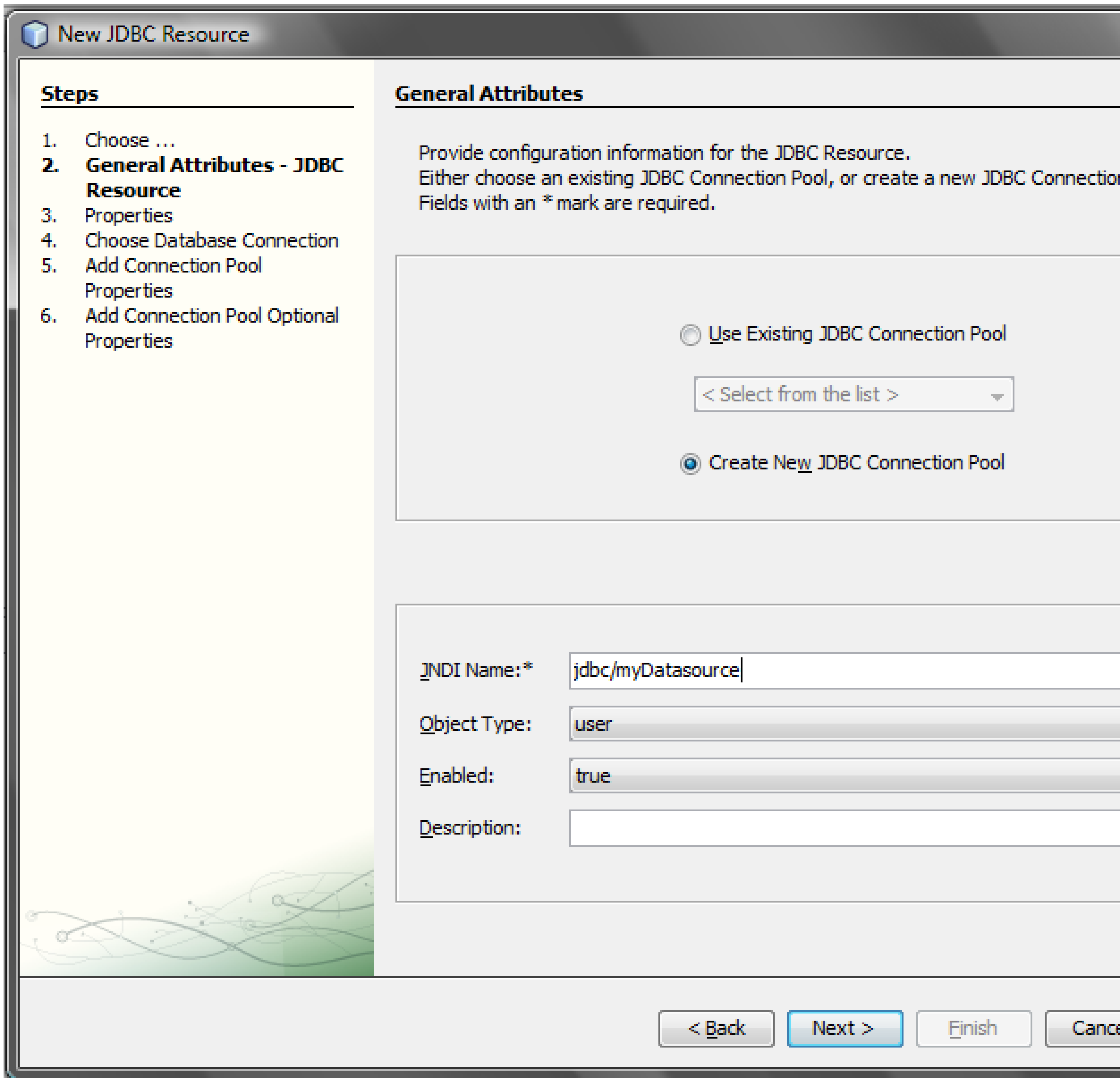
4. Creating Database Connection pool in application:

- Go to project window.
- Right click on server resources -> New -> Others. It will open a window as shown below.



Wizard to create a new JDBC Resource.

- Select your project.
- In Categories chose GlassFish , File Type select JDBC Resource and click next.


New JDBC Resource

Steps

1. Choose ...
2. **General Attributes - JDBC Resource**
3. Properties
4. Choose Database Connection
5. Add Connection Pool Properties
6. Add Connection Pool Optional Properties

General Attributes

Provide configuration information for the JDBC Resource.
Either choose an existing JDBC Connection Pool, or create a new JDBC Connectio
Fields with an * mark are required.

( ) Use Existing JDBC Connection Pool

< Select from the list >

(●) Create New JDBC Connection Pool

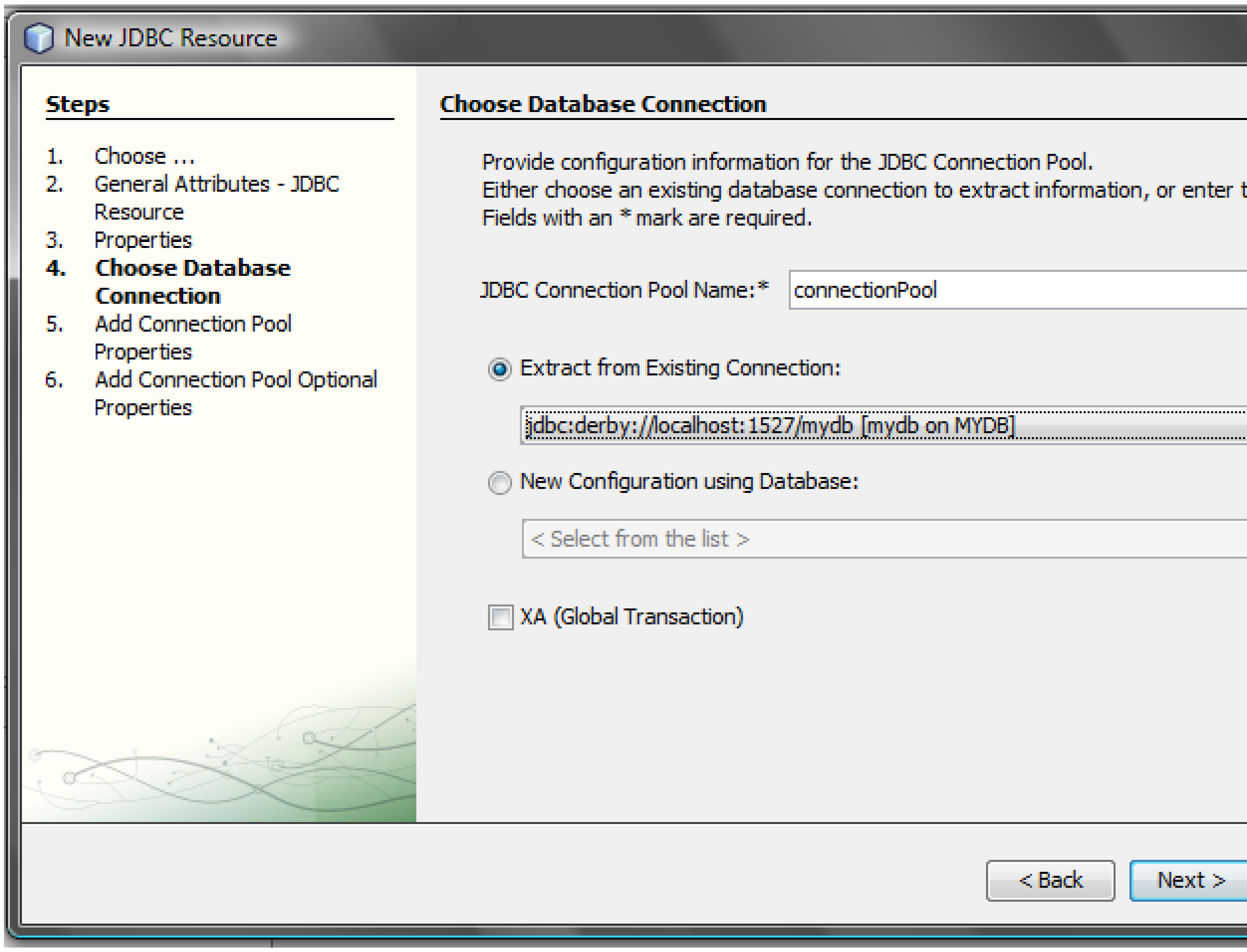
JNDI Name:* jdbc/myDatasource

Object Type: user

Enabled: true

Description:

- Select create New_JDBC Connection Pool, Give JNDI name.
- Here, I have given pool name jdbc/myDatasource.
- Then give next, so it will ask for properties at that window just click next.
- You will see window shown below.


New JDBC Resource

Steps

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
4. **Choose Database Connection**
5. Add Connection Pool Properties
6. Add Connection Pool Optional Properties

Choose Database Connection

Provide configuration information for the JDBC Connection Pool.
Either choose an existing database connection to extract information, or enter t
Fields with an * mark are required.

JDBC Connection Pool Name:* connectionPool

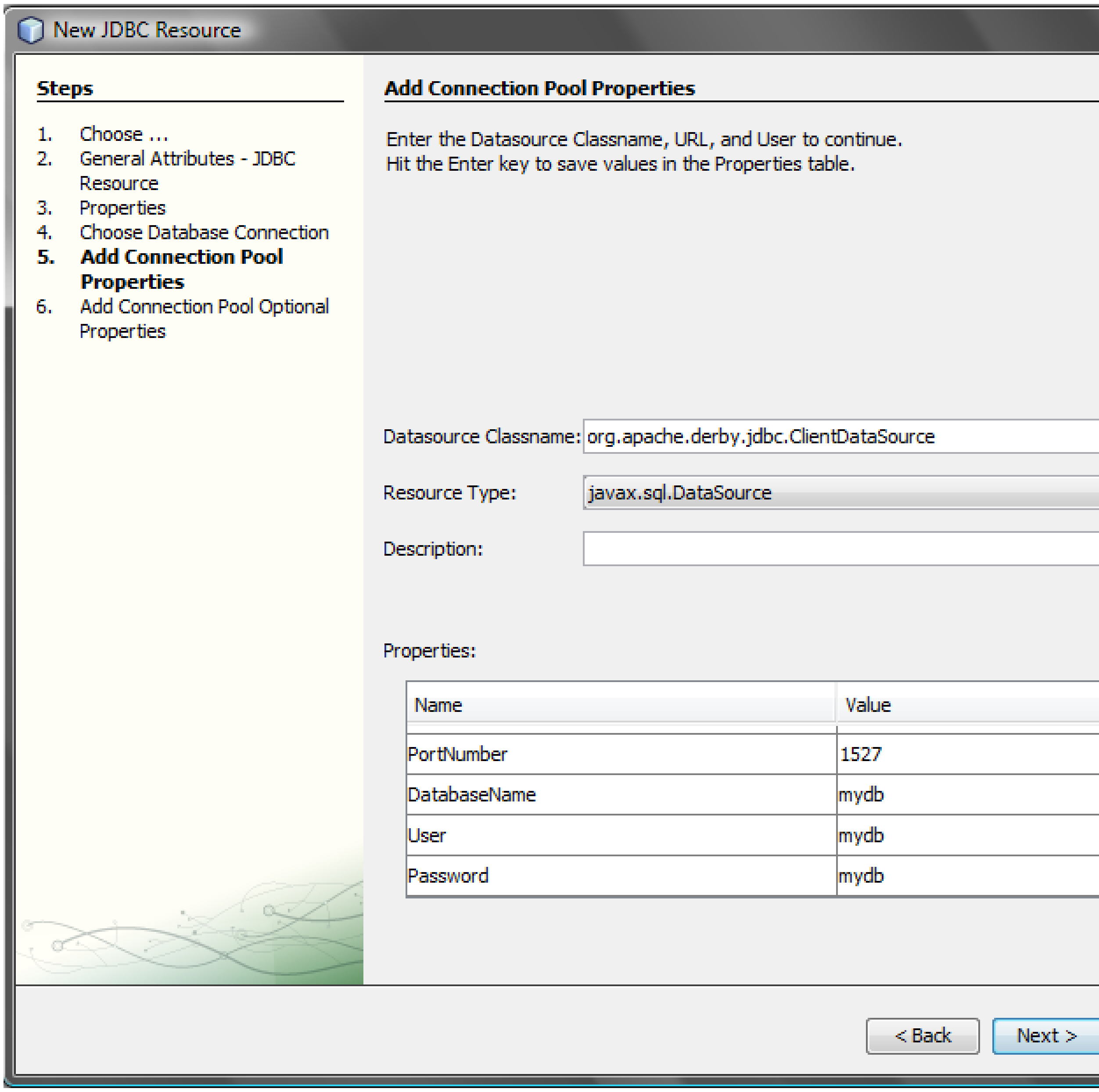
[✓ Extract from Existing Connection:] jdbc:derby://localhost:1527/mydb [mydb on MYDB]

New Configuration using Database: < Select from the list >

XA (Global Transaction)

< Back   Next >

- Here, Give connection pool name.
- Select a connection that you have created previously in services and click next.


New JDBC Resource

Steps

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
4. Choose Database Connection
5. **Add Connection Pool Properties**
6. Add Connection Pool Optional Properties

Add Connection Pool Properties

Enter the Datasource Classname, URL, and User to continue.
Hit the Enter key to save values in the Properties table.

Datasource Classname: org.apache.derby.jdbc.ClientDataSource

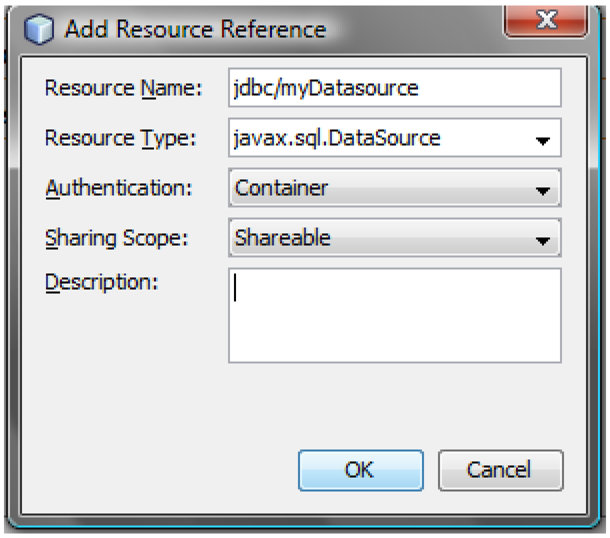
Resource Type: javax.sql.DataSource

Description:

Properties:

Name	Value
PortNumber	1527
DatabaseName	mydb
User	mydb
Password	mydb

- You will see connection pool properties. Now click finish button.
- Open Web pages -> WEB-INF -> web.xml
- Click on references tab, in Resource references click on Add.



Add Resource Reference

Resource Name: jdbc/myDatasource

Resource Type: javax.sql.DataSource

Authentication: Container

Sharing Scope: Shareable

Description:

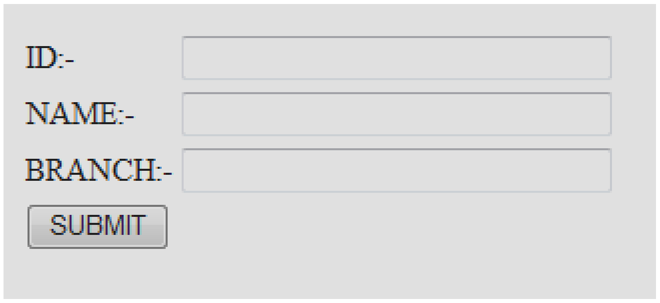
- Give resource name in the textbox. This is JNDI resource that we already created.
- Open Web pages -> WEB-INF -> sun-web.xml
- Between </jsp-config> and </sun-web-app> tag enter following lines.

```
<resource-ref>
<res-ref-name>jdbc/myDatasource</res-ref-name>
<jndi-name>jdbc/myDatasource</jndi-name>
</resource-ref>
```

- It will create reference name correspond to JNDI name.
- You can chose any arbitrary name as reference name.

5. Creating Insert page:

- Open Insert page, if not already open.
- Inside body part create one form. Put action of this form as insert.jsp.
- Inside that form tag create tax boxes and submit button.
- Put all things in such a way so that final view will look like as follows.
-



ID:-

NAME:-

BRANCH:-

- When you click submit button, request will go to page it self.
- So page will be reloaded. Code for this html form is as follows.

```
<form action="insert.jsp" method="POST">
```

```
<table border="0">
<tbody>
<tr>
<td>ID:-</td>
<td><input type="text" name="id" value="" size="30" /></td>
</tr>
<tr>
<td>NAME:-</td>
<td><input type="text" name="name" value="" size="30" /></td>
</tr>
<tr>
<td>BRANCH:-</td>
<td><input type="text" name="branch" value="" size="30" /></td>
</tr>
<tr>
<td colspan="2"><input type="submit" value="SUBMIT" /></td>
</tr>
</tbody>
</table>
</form>
```

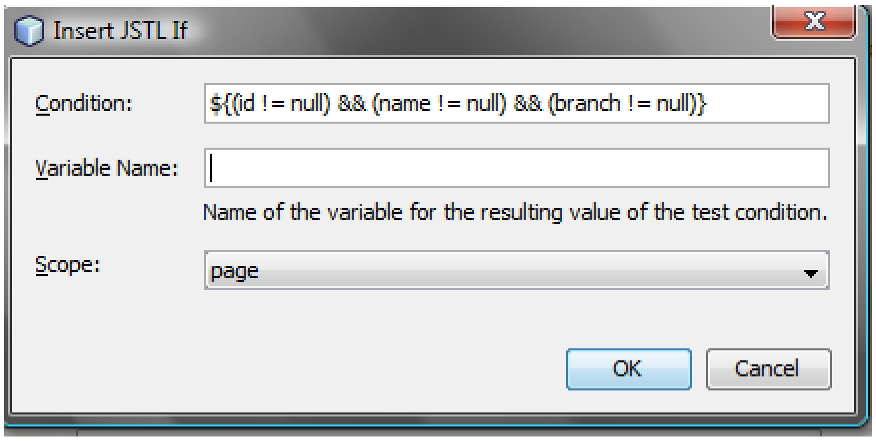
- Above <form> tag, put three c:set tags to get value of id, name and text.
- First time when this page is called these values are null.
- When we come to this page after click on submit, we get parameters from tax boxes that we want to enter into database.

```
<c:set var="id" value="${param.id}"/>
<c:set var="name" value="${param.name}"/>
<c:set var="branch" value="${param.branch}"/>
```

- For using standard tag library, you have to include following two lines at top in JSP page.

```
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

- After this c:set statements insert one JSTL IF tag.
- To do this, go to palette window, under that go to JSP tag, drag and drop JSTL If from there to application after last c:set tag.
- It will show you following window.



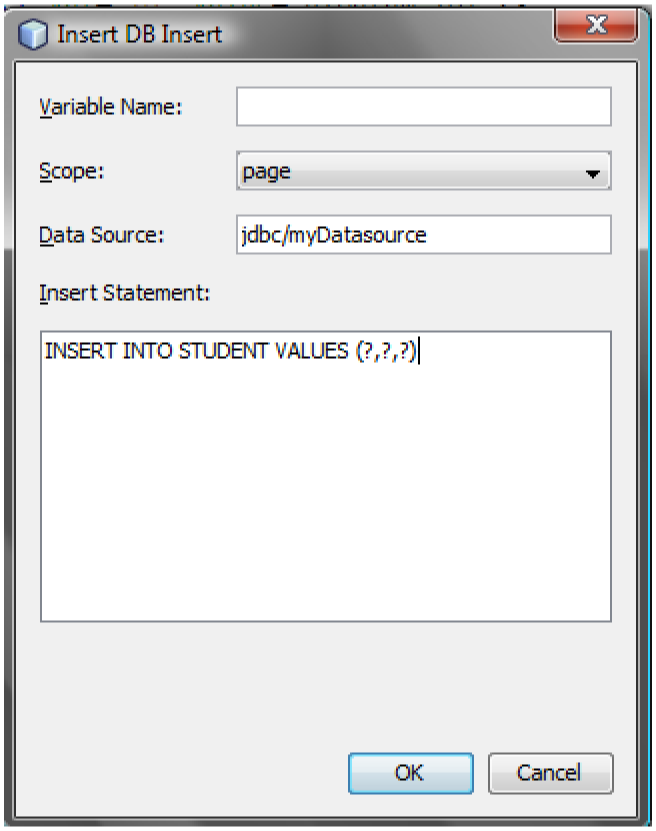
Condition: ${(id != null) && (name != null) && (branch != null)}

Variable Name:

Name of the variable for the resulting value of the test condition.

Scope: page

- Write condition as shown in fig.
- If this condition is true then and only then we want to insert data into database.
- So all the code inside c:if tage will be executed if and only if condition is true.
- Inside c:if tag write code for insert query.
- To do this, go to palette window, under that go to Database tag, drag and drop DB Insert from there to application inside c:if tag.
- It will show you following window. In which, you have to insert data source name that we have created and insert query as shown in fig.



Variable Name:

Scope: page

Data Source: jdbc/myDatasource

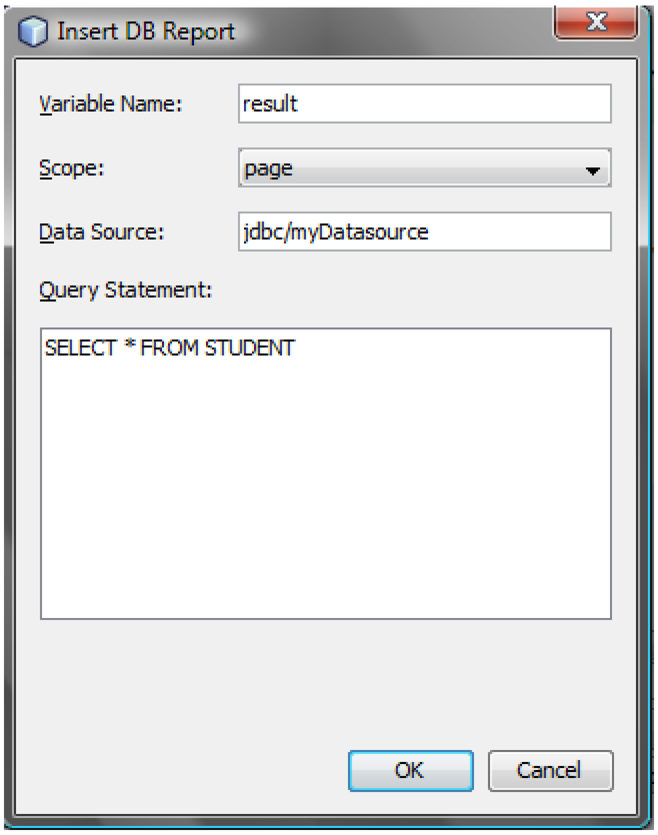
Insert Statement:

```
INSERT INTO STUDENT VALUES (?,?,?)
```

- We want to insert parameter value that we set using c:set.
- So after INSERT statement, we place three sql:param tags to set value of question mark.
- Add following three statements after INSERT statement but inside sql:update tag.

```
<sql:param value="${id}"/>
<sql:param value="${name}"/>
<sql:param value="${branch}"/>
```

- These statements set value of id in place of first question mark, value of name in place of second question mark and so on.
- Now, code to insert data into database table is over.
- We want to print STUDENT table detail.
- To generate report, go to palette window, under that go to Database tag, drag and drop DB Report from there to application after </form> tag.
- Here, insert query, data source name and variable as shown in fig and click ok.



Variable Name: result

Scope: page

Data Source: jdbc/myDatasource

Query Statement:

```
SELECT * FROM STUDENT
```

- It will add some code.
- When you built and run application for first time it will show empty html table with only column names in report.
- Because initially database table is empty.

- After you insert data into database, report show data into html table.

6. Creating Update page:

- We made update page such that, we can not update id (primary key).
- We choose any id and enter updated value for name and branch for that.
- All things are same as Insert.jsp page except followings.
- After clicking submit button, we run UPDATE query instead of INSERT query.
- You have to change action of form to update.jsp
- Instead of using text box for id, we use combo box. Code for that is as shown below.

```
<select name="id">
<sql:query var="result" dataSource="jdbc/myDatasource">
SELECT ID FROM STUDENT
</sql:query>
<c:forEach var="row" items="${result.rowsByIndex}">
<c:forEach var="column" items="${row}">
<option> <c:out value="${column}"/></option>
</c:forEach>
</c:forEach>
</select>
```

- Instead of insert query we put update query as shown below.

```
<sql:update var="res" dataSource="jdbc/myDatasource">
UPDATE STUDENT SET NAME = ?, BRANCH = ? WHERE ID
= ?
<sql:param value="${name}"/>
<sql:param value="${branch}"/>
<sql:param value="${id}"/>
</sql:update>
```

- This completes our update page.

7. Creating Delete page:

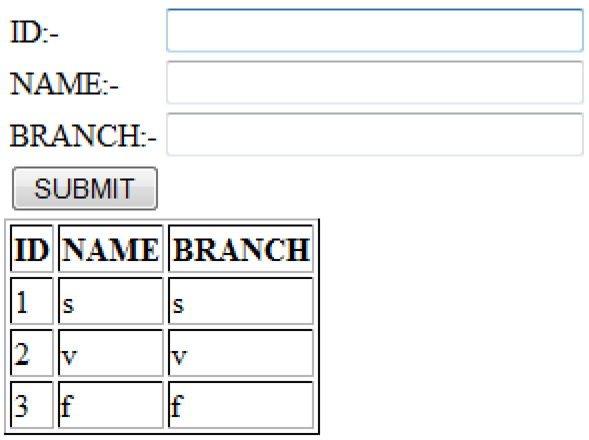
- All things are same as update page except following.
- We have to change action of form tag to Delete.jsp
- We remove tax box and label for name and branch.
- We also remove c:set for name and branch.
- Name and branch are also removed from every where.
- We will add one more option name as select in combo box for id.
- Now, if id is null or select then we have to do nothing.
- Otherwise we have to run DELETE query as shown below.

```
<c:set var="id" value="${param.id}"/>
<c:if test="${(id != null) && (id != 'select') }">
```

```
<sql:update var="res" dataSource="jdbc/myDatasource">
DELETE FROM STUDENT WHERE ID = ?
<sql:param value="${id}"/>
</sql:update>
</c:if>
```

8. Final Screenshots:

- Insert page

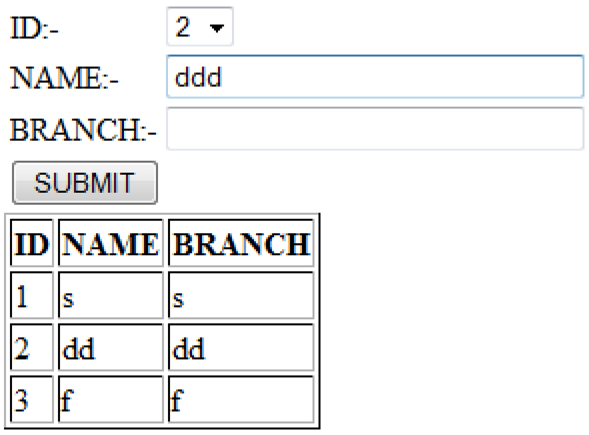


ID:-	
NAME:-	
BRANCH:-	

SUBMIT

ID	NAME	BRANCH
1	s	s
2	v	v
3	f	f

- Update Page

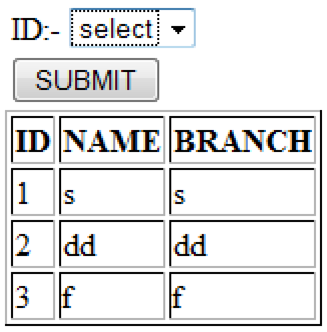


ID:-	2
NAME:-	ddd
BRANCH:-	

SUBMIT

ID	NAME	BRANCH
1	s	s
2	dd	dd
3	f	f

- Delete Page



ID:- select

ID	NAME	BRANCH
1	s	s
2	dd	dd
3	f	f