Web Databases – Part 1.1

How an end user interacts with a web database:

1. In a web browser, an end user submits a request for some dynamic data to the web server.
2. The web server passes it onto the middleware (for instances, CGI, Java Servlet, JSP, …).
3. The middleware writes the request in SQL queries and sends them to a back-end database using some API such as DBI, JDBC, ODBC, ….
4. The retrieved data are sent back to the middleware.
5. The middleware generates a web page for the data.
6. The web server sends the web page to the browser of the end user.
7. The browser displays the web page in front of the end user.

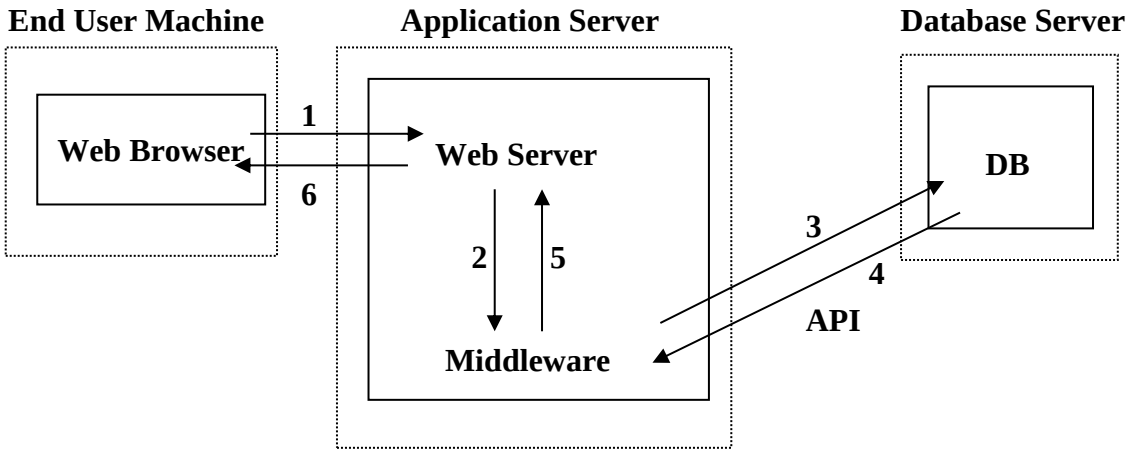


Figure 1. A generic web architecture

Common Gateway Interface (CGI)

- The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP or Web servers.

- A plain HTML document that the Web daemon retrieves is static, which means it exists in a constant state: a text file that doesn't change.

- A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information.

- C, C++, PERL, etc. can be used to write CGI programs.

Drawbacks:

- Creating a separate process for each client request is expensive in terms of processor and memory resources.

- Expensive to open and close database connections for each client request.

- At each client's request, the same CGI program has to be loaded, executed and terminated which often overloads the server.

Example:

Table Phonebook:

pnumber	Name
4053254042	Computer science

Step#1: SQL is used to create and populate the table "Phonebook":

```
create table Phonebook(
        pnumber varchar2(10) not null,
        Name varchar2(25),
        primary key(pnumber));


desc Phonebook;

Insert into Phonebook values ('4053254042', 'Computer Science');
```

Step#2: PERL and DBI are used to retrieve data from the table "Phonebook":

The following PERL code example sets up a connection to the www.yourwebsite.com database, prepares and executes an SQL statement, stores the result in a local variable, and then cleans up the connection.

```
# Use the DBI (Database Interface) module
use DBI qw(:sql_types);

# Declare local variables

my($databaseName, $databaseUser, $databasePw, $dbh);
my ($stmt, sth, @newRow);
my ($pnumber);

# Set the parameter values for the connection
my ($databaseName= "DBI:mysql:yourWebSite_com";
$databaseUser = "yourLoginId";
$databasePw = "yourLoginPassword";

# Connect to the database
# Note this connection can be used to execute more than one statement on any number of tables in the database
```

```
$dbh = DBI->connect($databaseName, $databaseUser,
   $databasePw) || die "Connect failed: $DBI::errstr\n";

# Create the statement.
$stmt = "SELECT Name FROM Phonebook
            WHERE pnumber= ’4053254042’";

# Prepare and execute the SQL query
$sth = $$dbh->prepare($$stmt)
   || die "prepare: $$stmt: $DBI::errstr";
$sth->execute || die "execute: $$stmt: $DBI::errstr";

# Get first record, If more than one record returned, put fetchrow in while loop
@record = $sth->fetchrow()

# Get the value of the first field returned.
$pnumber = $record[0];

# Clean up the record set and the database connection
$sth->finish();
$dbh->disconnect();
```

All queries follow the same basic formula. Simply replace the SELECT statement with the INSERT, UPDATE, DELETE, etc. statement you wish to use. Note that these other queries do not return records. So, the fetchrow() and assignment which follows should be deleted for them.

Many other operations such as joins, subqueries, grouping, and sorting are all supported by providing a proper SQL statement in place of the one above.

SERVLETS

- Servlets are Java applications that can perform the same work of a CGI program.

- Performance is significantly better. Creating a separate process to handle each client request isn't necessary.

- With servlets, the Java Virtual Machine stays up, and each request is handled by a lightweight Java thread, not a heavyweight operating system process.

- If N requests to the same Program come up, there will be N threads, but only a single Servlet class processes the requests.

- Optimization techniques like caching previous computations and having database connections open are some of the advantages.

- Servlets have ability to share data between each other, making database connection pools easy to implement.

- Servlets are platform-independent, because they are written in JAVA.

- Servlets are loaded and initialized only once in their life cycle.

Life Cycle of a Servlet

- It involves three methods, init(), service() and destroy().

- They are implemented by every servlet and also invoked by the server at specific times.

- Consider a typical user scenario; first assume that a user enters a URL to a web browser. The browser generates an HTTP request for this URL and sends it to the appropriate server.

- The HTTP request is received by the web server. The server maps this request to a particular servlet.

- The servlet is dynamically retrieved and loaded into the address space of the server.

- The server invokes the init() method of the servlet. This happens only when the servlet is first loaded into memory.

- The server invokes the servlet's service() method. The servlet then processes the HTTP request.

- The servlet remains in the server's address space and is available to process other HTTP requests received from the client.

- When the server decides to unload the servlet from the memory, it calls the destroy() method.

Java Database Connectivity (JDBC)

- JDBC technology is an API (Application Programming Interface) for accessing virtually any tabular data source from the Java™ programming language.

- It provides cross-DBMS connectivity to a wide range of SQL databases.

- It also provides access to other tabular data sources such as spreadsheets or flat files.

- It allows developers to take advantage of the Java platform's "Write Once, Run Anywhere" capabilities.

- With a JDBC driver, a developer can easily connect to database systems even in a heterogeneous environment.

- The JDBC API is comprised of two packages:

    - The *java.sql* package which contains classes and interfaces for manipulating relational databases;

- The *javax.sql* package which contains all the JDBC Technology Drivers.

Example for Web Database using Java Servlets and JDBC

Important Points to remember:

1. Create tables and populate the tables using SQL.
2. Then design the servlets.
3. The name of the java class should be the same as the *.java file name.

Basic Steps:

(The following assumes that you have jdbc drivers and jsdk installed and have your system set to work with those. Also you must have the source code in appropriate directory.)

- Build the database using SQL and populate the tables with your own data.
- Create the servlet source code in Java, say query1.java
- Compile the source code: javac query1.java
- Run the STARTSERVER utility:
    - Open a command prompt window and type **startserver**. This tool listens on port 8080 for incoming client requests.

- Start a web browser and request the servlet:
  - Start a web browser and enter the URL as http://localhost:8080/servlet/query1
  - Alternatively, the following URL may also be entered: http://127.0.0.1:8080/servlet/query1
  - This can be done because 127.0.0.1 is defined as the IP address of the local machine.

Example 1: the usage of Java Servlets and JDBC for inserting data into a table.

Table: Department

Department Name	Department Chair	Department Location
CS	Dr. Susan Smith	NB Bldg

Step#1: SQL is used to create and populate the table "Department":

```
create table Department(
        Dname varchar(10) not null,
        Chair varchar(25),
        Location varchar(15),
        primary key (Dname));

desc Department;

Insert into Department values ( 'CS', 'Dr. Susan Smith ', 'NB Bldg');
```

Step#2: Java Servlet and JDBC are used to populate the table "Department":

File Name: Query1.java

```
/**************************************************************************
 *
```

```
*                    Name:      ABC
*                    Student Id: 123
*
*********************************************************************
* This program code is for a java servlet with JDBC connection to oracle
* database. This invokes a form for logging into the database.
* It then invokes a form for inserting data into the table Department.
*
*          INPUT VARIABLES: a) login, password---- To connect to the database
*                                    b) Department name, Department chair, Department location
*                                        To be inserted into the table " Department".
*
*********************************************************************/
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public final class Query1 extends HttpServlet
{
  /* doGet simply calls doPost; doGet is called if
   * method="GET" in the HTML form.
   */
  protected void doGet(HttpServletRequest request,
                   HttpServletResponse response)
                   throws ServletException, IOException
  {
    doPost(request, response);
  }

  protected void doPost(HttpServletRequest request,
                    HttpServletResponse response)
                    throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();


    /* Some standard HTML header tags
     */
    out.println("<html>");
    out.println("<head><title>Query1</title></head>");
    out.println("<body bgcolor=\"333333\">");

    /* Get the parameters from the form data
     */
    String login = request.getParameter("login");
    String password = request.getParameter("password");

          String url = "jdbc:oracle:thin:" + login + "/" + password +
             "@tarjan.cs.ou.edu:1521:csdb";
            String Dname = request.getParameter("Dname");
            String Chair = request.getParameter("Chair");
            String Location = request.getParameter("Location");
```

```
    try
    {
      if( login==null && password==null)
      {
        /* This is the first time the servlet is "hit"
         * Show only the login form
         */
        printForm(out, null, true, false);
      }
      else
      {
        /* login &pass were given.
         * Try to connect to the database.
         */
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection conn = DriverManager.getConnection(url, login, password);


          if (Dname!=null && Chair!=null && Location!=null)
          {
              Statement stmt = conn.createStatement();
              stmt.executeUpdate("Insert into Department values (‘" + Dname + "’,’" + Chair + "’,’" +
Location + "’)");
              out.println("Row inserted " "<br>");
          }

        printForm(out, conn, true, true);
      }
    }
    catch(SQLException ex)
    {
      out.println("<pre>"+ex+"</pre><br>");
      out.println("login was: " + login + "<br>");

      // only print password if login/pass were wrong
      if( ex.getErrorCode()==1017 || ex.getErrorCode()==1005 )
      {
        out.println("password was: " + password + "<br>");
      }
    }

    /* End the HTML document
     */
    out.println("</body>");
    out.println("</html>");

    out.close();
  }


  private void printForm(PrintWriter out,
                         Connection conn,
                         boolean printLogin,
                         boolean printDept) throws SQLException

  {
```

```
    /*  The action is the Servlet itself.
     */
    out.println("<form action=\"Query1\" method=\"post\">");


    /* Print the simple login form
     */
    if( printLogin )
    {
      out.println("Login: <input name=\"login\" type=\"text\" size=\"10\">");
      out.println("Password: <input name=\"password\" type=\"password\" size=\"10\">");
    }

    out.println("<hr>");

    /* print form for insertion */
    if( printDept )
    {
      out.println("Enter the Department:<input name=\"Dname\" type=\"text\" size=\"10\">");
      out.println("Enter the Chair:<input name=\"Chair\" type=\"text\" size=\"10\">");
      out.println("Enter the Location:<input name=\"Location\" type=\"text\" size=\"10\">");

      out.println("<hr>");

    }

    /*  We must add a submit button!
     */
    out.println("<hr>");
    out.println("<input type=\"submit\" value=\"Execute\">");
    out.println("</form>");
  }
}
```

Example 2: Retrieval of data using Java Servlets and JDBC

Table: Faculty

SSN	Name	Title	Specialization	Department
000-75-1234	Le Gruenwald	Prof	Database	CS

Step#1: Use SQL to create the table Faculty and insert a row of data into it:

```
create table Faculty(
        Ssn varchar(15) not null,
        Name varchar(25),
        Title varchar(10),
```

```
        Specialization varchar(20),
        Dname varchar(10),
        primary key(Ssn));
```

desc Faculty;

Insert into Faculty values ('000-75-1234', 'Le Gruenwald', 'Prof', 'Database', 'CS');

Step#1: Use Java Servlet and JDBC to retrieve data from the table Faculty

Name of the file: `Query2.java`

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public final class Query2 extends HttpServlet
{
  /* doGet simply calls doPost; doGet is called if
   * method="GET" in the HTML form.
   */
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException
  {
    doPost(request, response);
  }

  protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();


    /* Some standard HTML header tags
     */
    out.println("<html>");
    out.println("<head><title>Query2</title></head>");
    out.println("<body bgcolor=\"333333\">");

    /* Get the parameters from the form data
     */
    String login = request.getParameter("login");;
    String password = request.getParameter("password");
    String url = "jdbc:oracle:thin:" + login + "/" + password +
              "@turing.cs.ou.edu:1521:csdb";
     String Specialization = request.getParameter("Specialization");
```

```
    try
    {
      if( login==null && password==null)
      {
        /* This is the first time the servlet is "hit"
         * Show only the login form
         */
        printForm(out, null, true, false);
      }
      else
      {
         /* Connect to database and retrieve the information required  */
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection conn = DriverManager.getConnection(url, login, password);

          if ( Specialization!=null)
          {
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT Name, Title from Faculty where Specialization =
‘" + Specialization + "’" );

              while(rs.next())
               {
                     out.println("<option>"+ rs.getString(1) + "</option>");
                     out.println("<option>"+ rs.getString(2) + "</option> <br>");
               }
          }

        printForm(out, conn, true, true);
      }
    }
    catch(SQLException ex)
    {
      out.println("<pre>"+ex+"</pre><br>");
      out.println("login was: " + login + "<br>");

      // only print password if login/pass were wrong
      if( ex.getErrorCode()==1017 || ex.getErrorCode()==1005 )
      {
        out.println("password was: " + password + "<br>");
      }

    }

    /* End the HTML document
     */
    out.println("</body>");
    out.println("</html>");

    out.close();
  }

  private void printForm(PrintWriter out,
                         Connection conn,
                         boolean printLogin,
                         boolean printFac) throws SQLException
```

```
  {
    /*  The action is the Servlet itself.
     */
    out.println("<form action=\"Query2\" method=\"post\">");

    /* Print the simple login form
     */
    if( printLogin )
    {
      out.println("Login: <input name=\"login\" type=\"text\" size=\"10\">");
      out.println("Password: <input name=\"password\" type=\"password\" size=\"10\">");
    }

    out.println("<hr>");

    /*  Specify one column name, which is used to retrieve faculty information.
     */
    if( printFac)
    {
      out.println("<Enter the specialization: <input name=\”Specialization\" type=\"text\" size=\"20\">");
    }

    /*  We must add a submit button!
     */
    out.println("<hr>");
    out.println("<input type=\"submit\" value=\"Execute\">");
    out.println("</form>");
  }
}
```

JSP

- JavaServer Pages (JSP) technology enables Web developers and designers to rapidly develop and easily maintain information-rich, dynamic Web pages that leverage existing business systems [http://www.tutorialspoint.com/jsp/jsp_overview.htm].
- As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform-independent.
- JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.
- JSP are translated and compiled into Java servlets but are easier to develop than Java servlets.

Advantages of JSP over Java Servlet

- Although JSP and Java servlet can finish the same task, it is more convenient to write and modify regular HTML than to have a zillion println statements that generate the HTML.

- By separating the look from the content, a big project can be done by different people in a perfect way. That is, web page design experts can build the HTML and leave rooms for Java servlet specialists to insert the dynamic content.

- JSP reaps all the benefits provided by Java servlets and Web container environment, but they have an added advantage of being simpler and more natural program for Web enabling enterprise developer.

Installing the JDBC driver for Eclipse

Step 1
Download JDBC driver ojdbc8.jar from the link:

https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/jdbc-ucp-183-5013470.html

Step 2
Write a *Java* program to use *Java JDBC* to connect to the Oracle database. Add *ojdbc8.jar* to the Java build path. If you are using eclipse, this can be done by right click *project name -> Properties -> Java Build Path -> Libraries -> Add External JARS* and select *ojdbc8.jar*.

Using JDBC driver for Eclipse

Step 1
Import the java.sql package.

```
import java.sql.*;
```

Step 2
Load a database driver.

```
try {
        Class.forName("oracle.jdbc.OracleDriver");

} catch (Exception x) {
          System.out.println( "Unable to load the driver class!" );
}
```

Step 3

Create an Oracle JDBC Connection. Replace the LoginName and Password in the statement below with your Oracle SQL Developer login name and password, respectively.

```
try {
    Connection dbConnection =
DriverManager.getConnection("jdbc:oracle:thin:@//oracle18.cs.ou.edu:1521/orclpdb",
”username”, ”password”);
} catch(SQLException x) {
   System.out.println( “Couldn’t get connection!” );
}
```

Step 4

Create an Oracle Statement object.

To execute SQL statements, you need to instantiate a Statement object from your connection object by using the createStatement() method.

```
Statement stmt = dbConnection.createStatement();
```

Step 5

Execute a SQL statement with the Statement object.

```
stmt.executeUpdate(sqlCreate);
or
stmt.executeQuery(query);
```

Eclipse

- Eclipse is an integrated development (IDE) environment commonly used for developing Java programs
- The latest version of Eclipse can be downloaded from https://www.eclipse.org/downloads/packages/
- Please download “Eclipse IDE for Enterprise Java Developers” which contains all the necessary packages for java web development

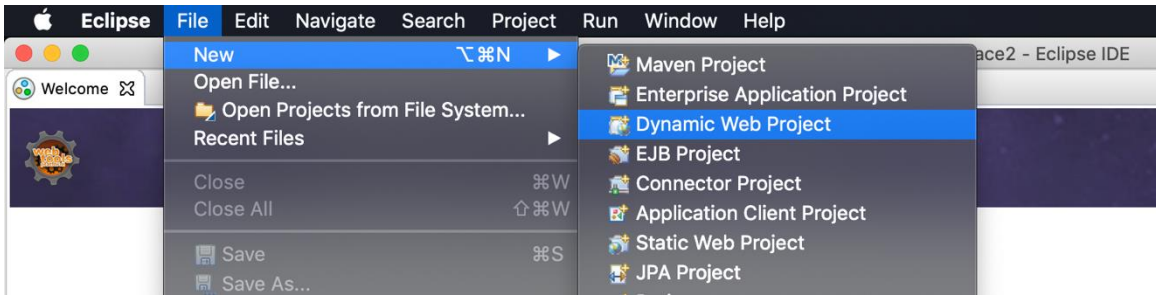
Example for Web Database using JSP and JDBC

1. Build the database using SQL and populate the tables with your own data.

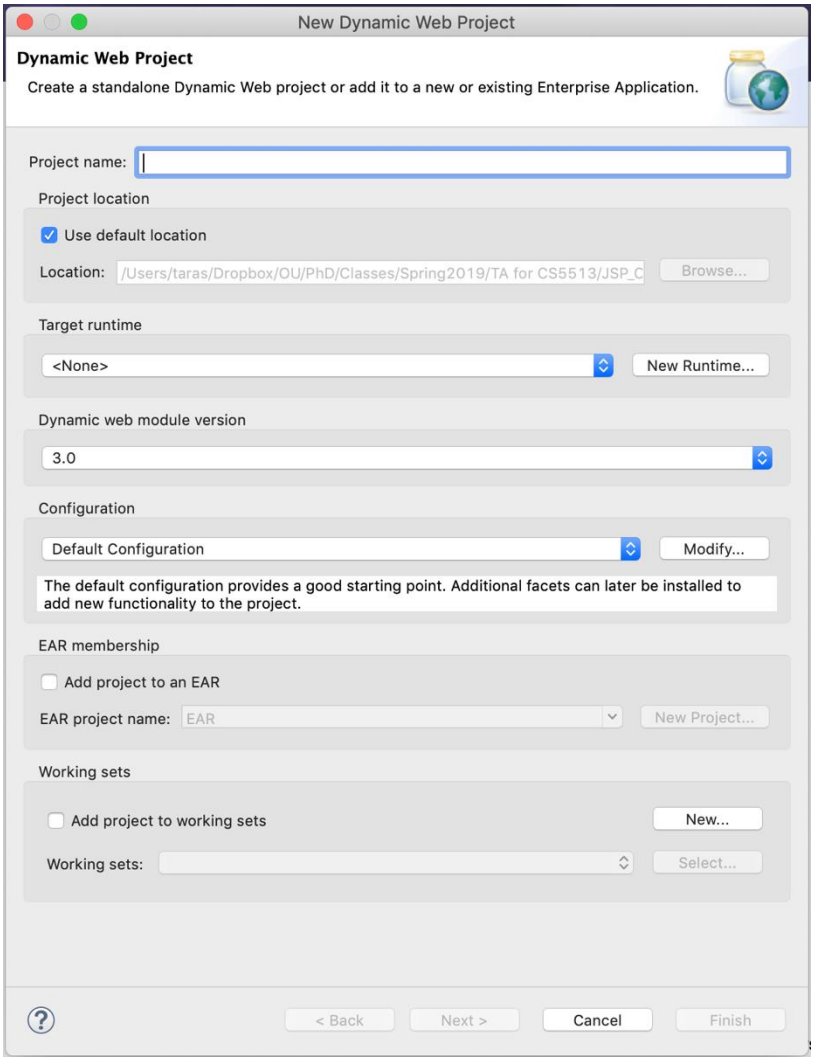
2. Download a ZIP archive with the latest version of Apache Tomcat 9 Web-Server (9.0.16 at the time of this writing) available at the link below:
    a. https://tomcat.apache.org/download-90.cgi
    b. Unzip the archive, record the resulting directory location

3. Launch Eclipse (preferably, create and select a new workspace directory when prompted)

4. Navigate the menu and click “File” -> “New” -> “Dynamic Web Project” (see screenshot below)



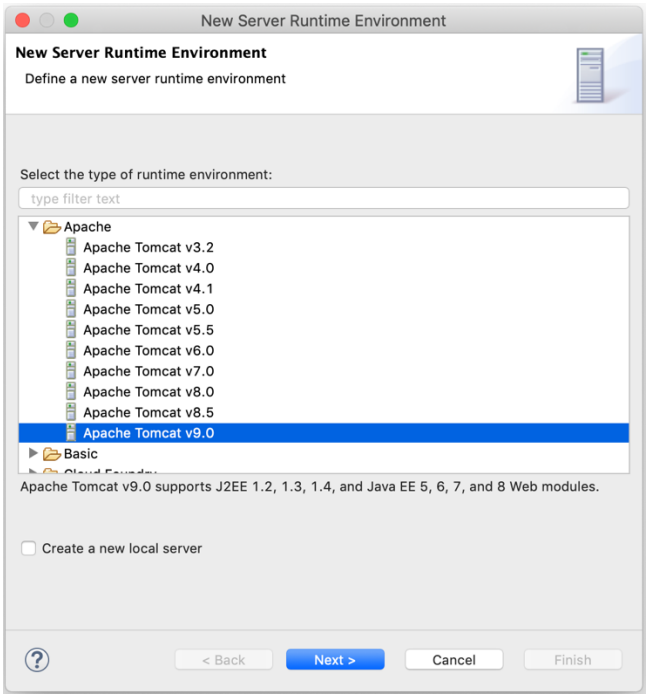
5. You should now see the below project creation window



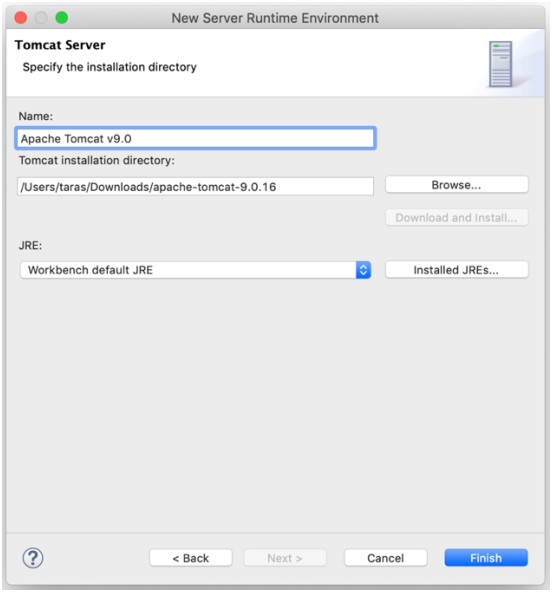
6. Give project a name, “jsp_oracle_test” for example

7. Click on “New Runtime” button

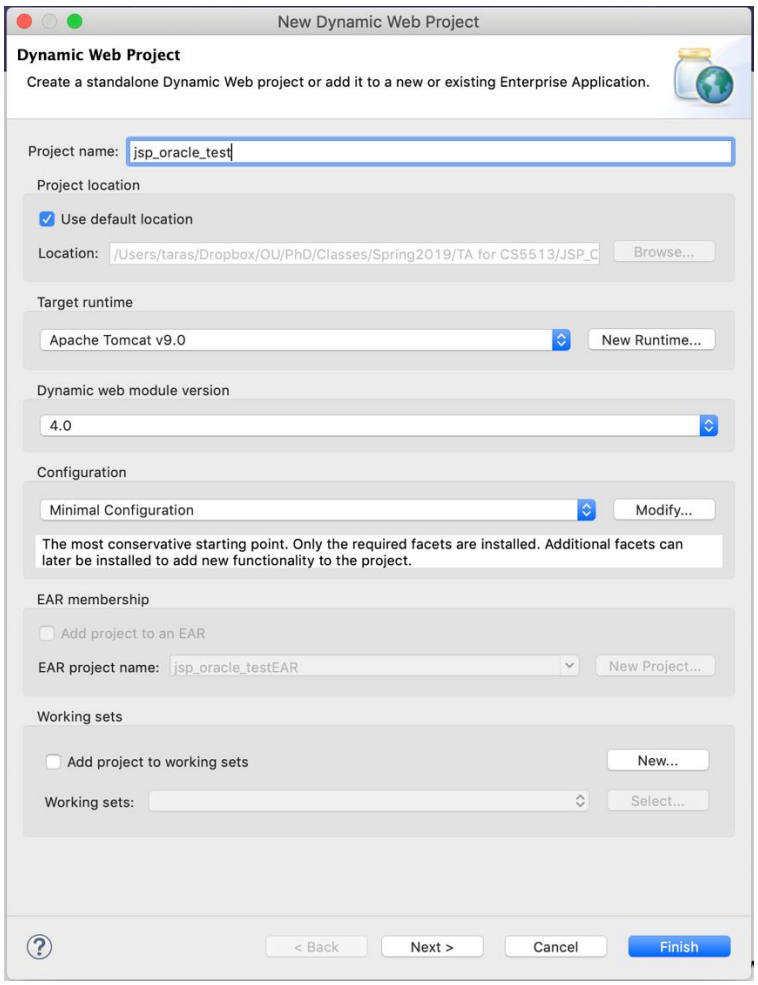
8. In a pop-up window, select “Apache Tomcat v9.0” in the “Apache” folder, click on “Next” button (see the screenshot below)



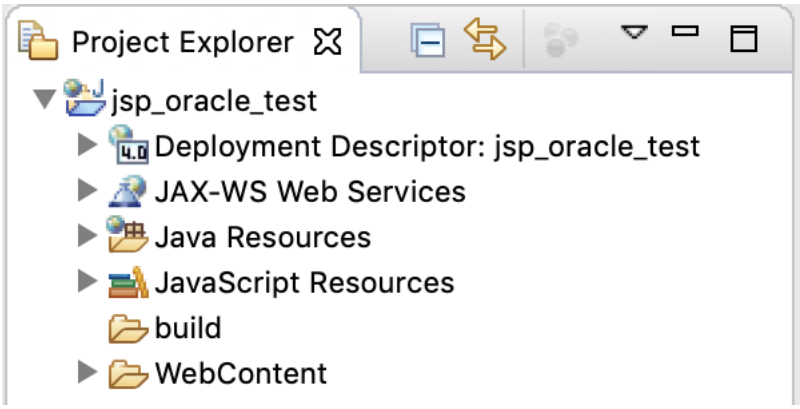
9. Click on the “Browse…” button, navigate to and select the directory where you unzipped your Apache Tomcat 9 archive, click “Open” button. Your “New Server Runtime Environment” should now look like the screenshot below. Click the “Finish” button.



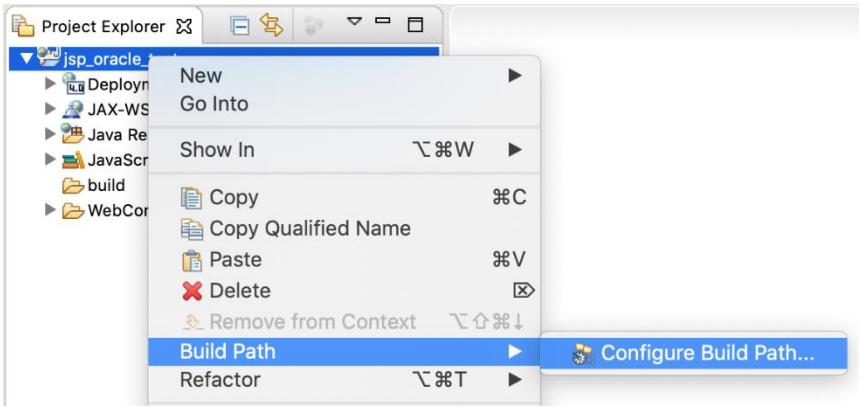
10. Now, back at the “New Dynamic Web Project” window (see screenshot below), select “Minimal Configuration” under “Configuration” tab, click on “Finish” button



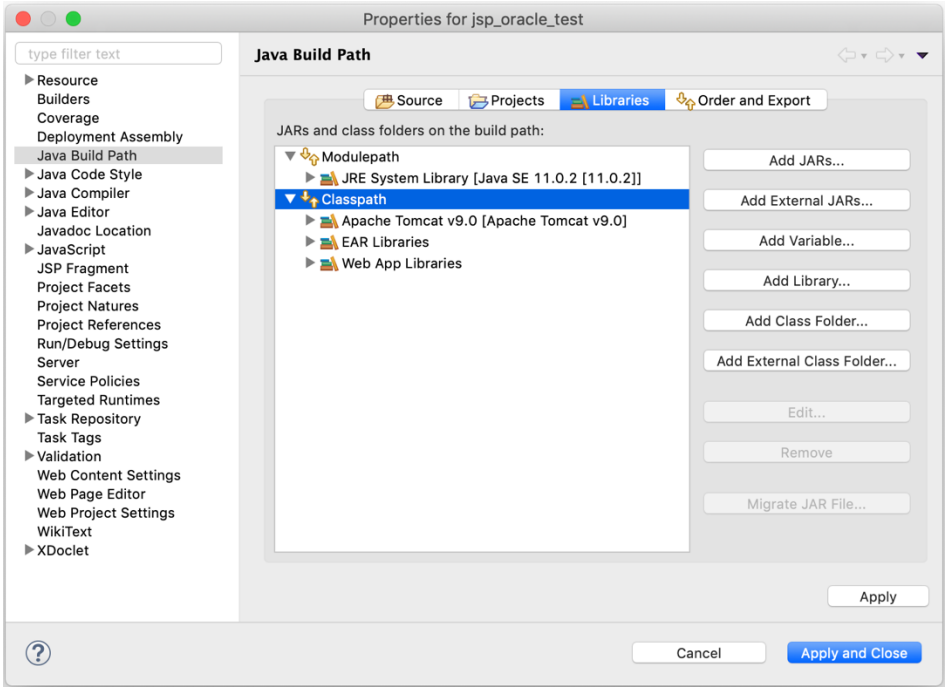
11. Your project explorer window should now look like the screenshot below (you might need to close the “Welcome” window first to see it)



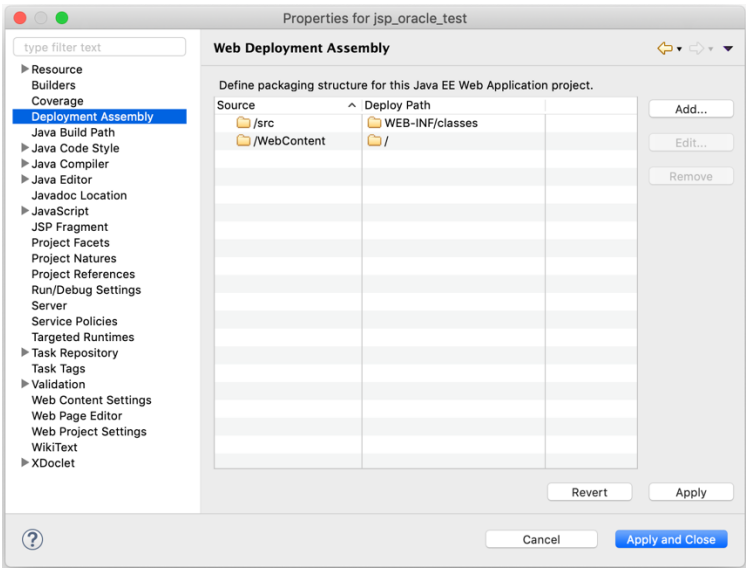
12. Right-click on the project name in the Project Explorer tab, select “Build Path” -> “Configure Build Path…” (see screenshot below)



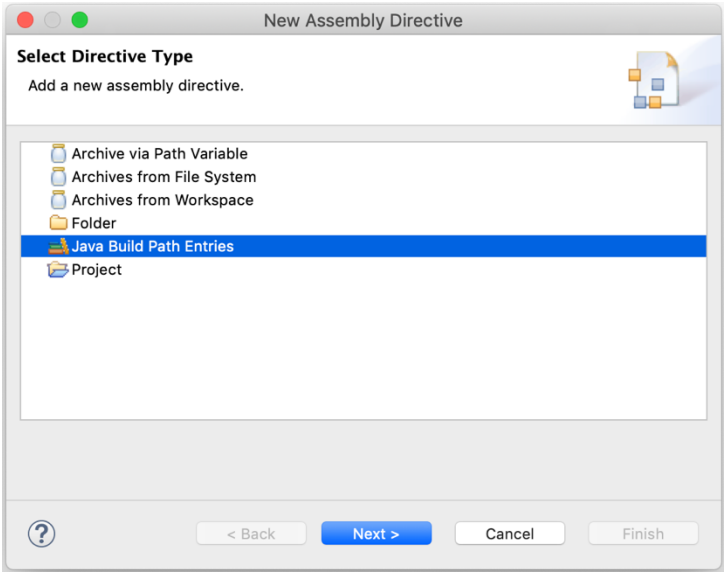
13. Select “Libraries” tab, click on “Classpath”, click on “Add External JARs…” button



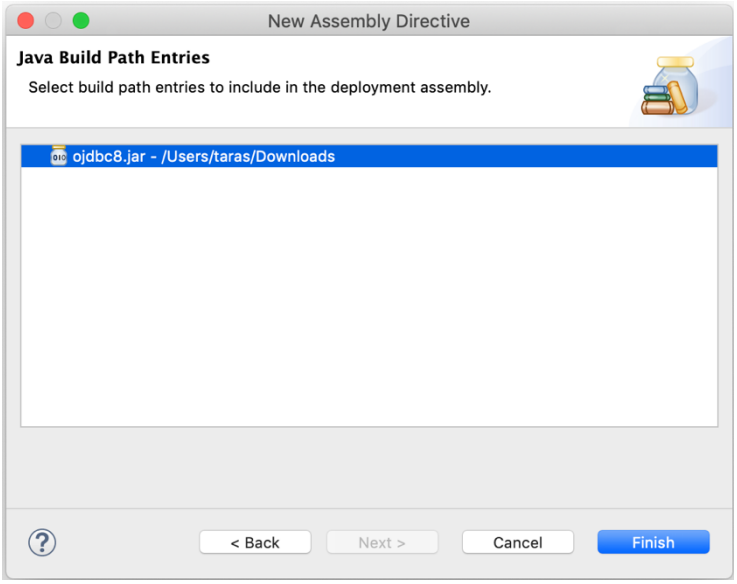
14. Navigate to and select the ojdbc8.jar you downloaded earlier, click on “Open” button. Now click on “Apply” button.
15. Don’t close project properties window yet, select “Deployment Assembly” tab on the left, click on the “Add…” button (see screenshot below).



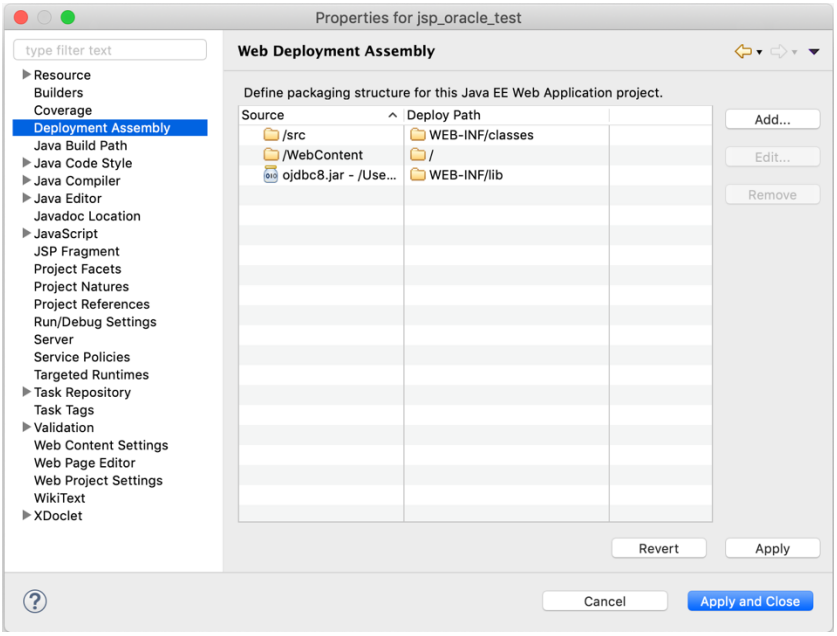
16. In the “New Assembly Directive” window select “Java Build Path Entries”, click “Next >” button (see screenshot below).



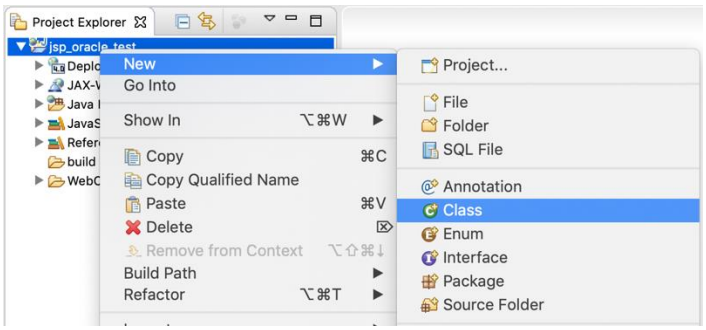
17. Select “ojdbc8.jar” and click on “Finish” button.



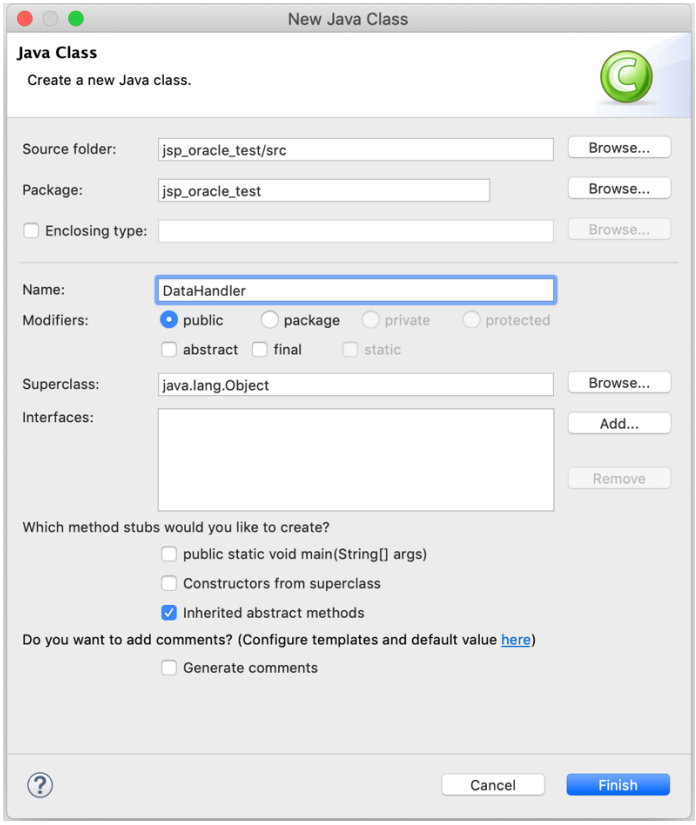
18. Your project properties window should now look like the window below, click on “Apply and Close” button.



19. Right click on the project name, click on “New” -> “Class” (see screenshot below).

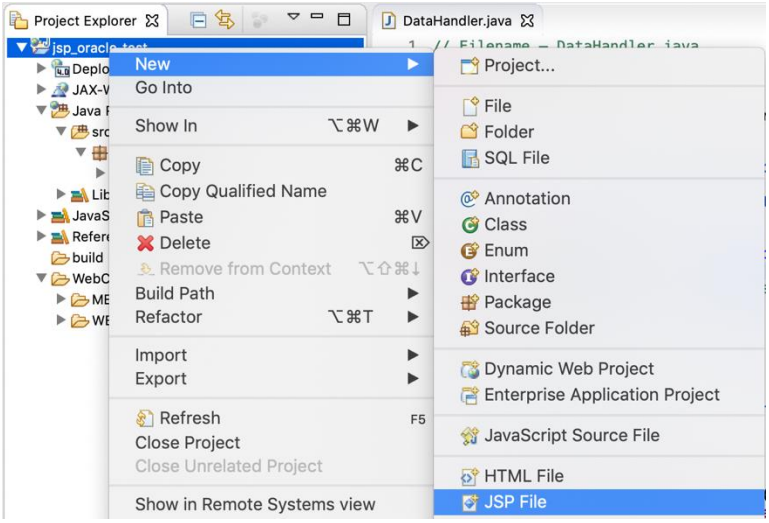


20. Name your new Java class DataHandler and click “Finish” button (see screenshot below).

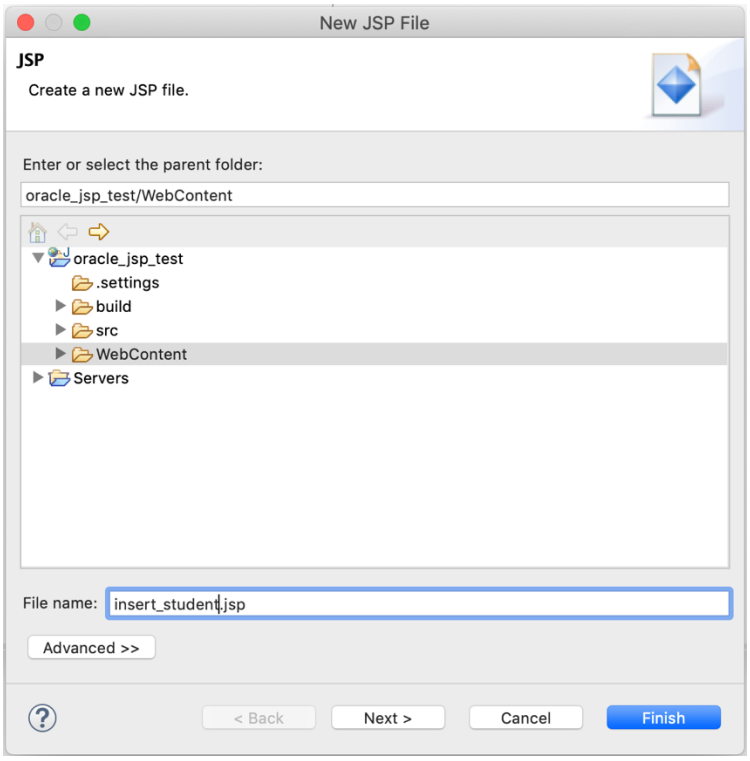


21. This file contains all the methods that are used to implement the important functions of the application. It includes methods that validate user credentials, connect to the database, retrieve data, insert data, update data, handle exceptions, and so on.

22. Right click on the project name, click on “New” -> “JSP File” (see screenshot below)

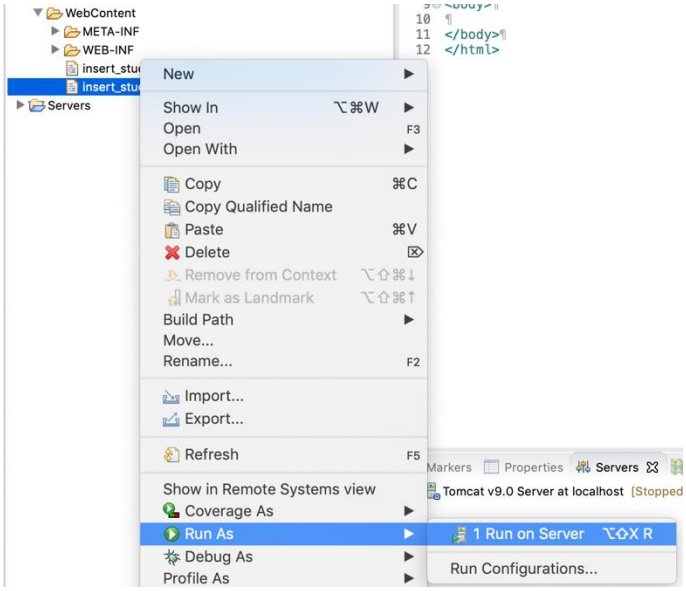


23. Name your JSP file (“insert_student.jsp”, in this example), click on “Finish” button (see the screenshot below)

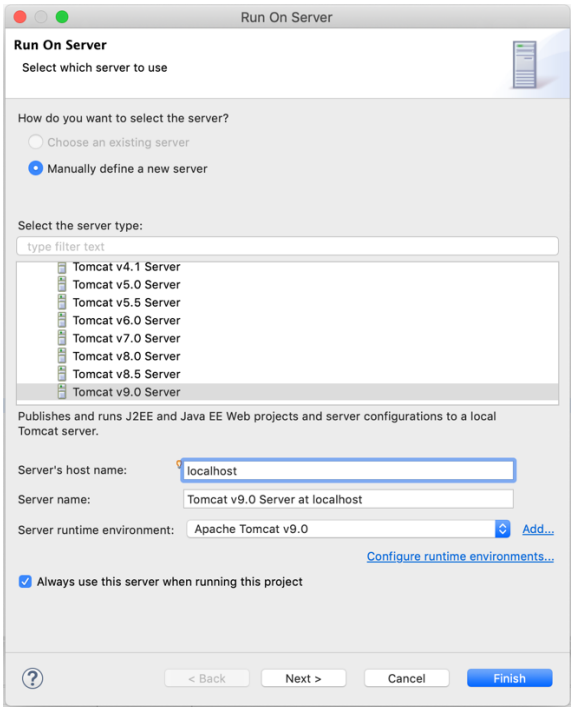


24. JSP files allow end users to login the database, retrieve data and manipulate data. (insert_student.jsp in the example)
25. Now, create a corresponding action jsp that carries out the action with the data collected in the previous jsp. (insert_student_action.jsp in the example)

26. To test your files, right click on insert_student.jsp in the Project Explorer tab, select “Run As” -> “Run on Server” (see screenshot below)

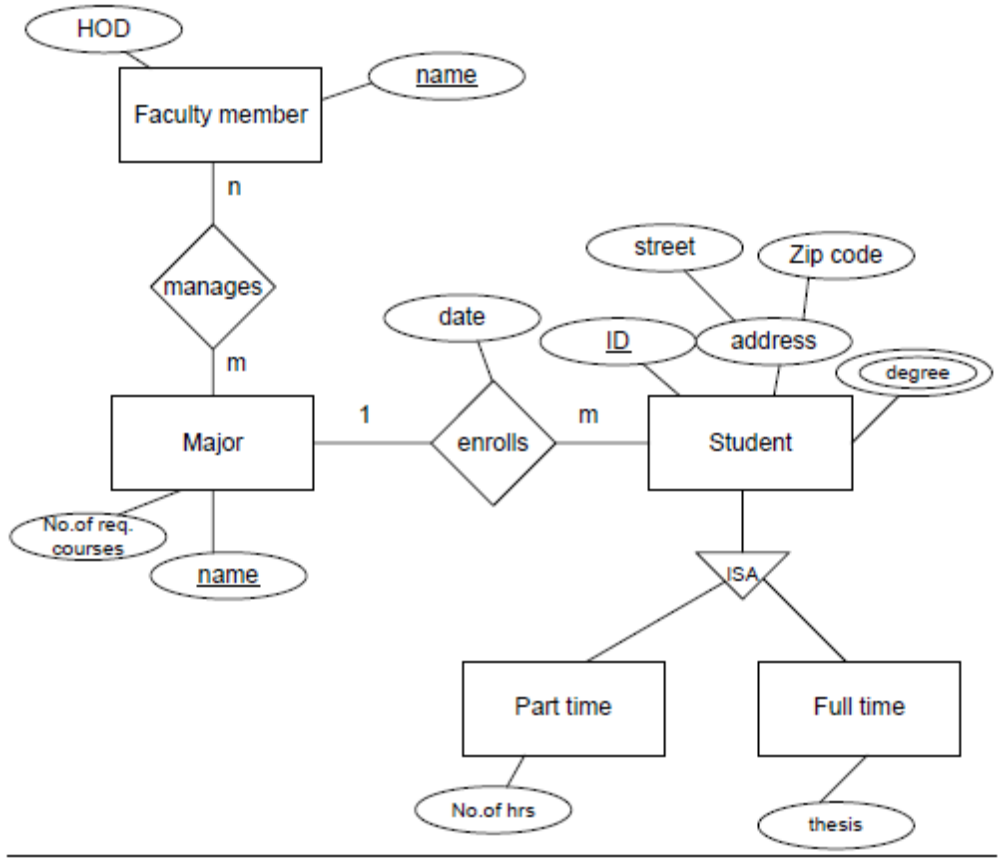


27. First time you do the above, you will be asked to select a web-server (see below screenshot). Select “Always use this server…”, click on “Finish” button.



Example 1: the usage of JSP and JDBC for inserting and retrieving data.

ER Diagram:



Step#1: SQL is used to create some tables, so that we can insert a new student

CREATE TABLE student(id number(9), street varchar(25), zip_code number(5), PRIMARY KEY (id));

CREATE TABLE degree(id number(9), degree varchar2(25), PRIMARY KEY (id,degree), FOREIGN KEY (id) REFERENCES student ON DELETE CASCADE);

CREATE TABLE student_fulltime(id number(9),thesis varchar2(50),PRIMARY KEY (id),FOREIGN KEY (id) REFERENCES student ON DELETE CASCADE);

CREATE TABLE student_parttime(id number(9),no_of_hours number(3),PRIMARY KEY (id),FOREIGN KEY (id) REFERENCES student ON DELETE CASCADE);

Step#2: JSP and JDBC are used to populate the tables

DataHandler.java // connect to the database and perform user authentication

```
package oracle_jsp_test;

// import all necessary libraries
import java.io.FileNotFoundException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import oracle.jdbc.pool.OracleDataSource;
import java.sql.Statement;
import java.sql.ResultSet;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import javax.servlet.http.HttpSession;


public class DataHandler {

//specify the database connection string and log in information

	String jdbcUrl = "jdbc:oracle:thin:@//oracle18.cs.ou.edu:1521/orclpdb";
	String userid = "<username>"; //your Oracle username
	String password = "<password>"; // your Oracle password
	Connection conn;
	Statement stmt;
	ResultSet rset;
	String query;
	String sqlString, sqlString2;

//create a new database connection

	public void getDBConnection() throws SQLException{
		OracleDataSource ds;
		ds = new OracleDataSource();
		ds.setURL(jdbcUrl);
		conn=ds.getConnection(userid,password);
	}

//authenticate user; if valid then connect to the database otherwise display
//error message

	public boolean authenticateUser(String jdbcUrl, String userid, String
password,
			HttpSession session) throws SQLException {
		this.jdbcUrl= jdbcUrl;
		this.userid = userid;
		this.password = password;
		try {
			OracleDataSource ds;
			ds = new OracleDataSource();
```

```
                        ds.setURL(jdbcUrl);
                        conn = ds.getConnection(userid, password); //connect to DB
                        return true;
                }
                catch ( SQLException ex ) {
                        System.out.println("Invalid user credentials");
                        session.setAttribute("loginerrormsg", "Invalid Login. Try
Again...");
                        this.jdbcUrl = null;
                        this.userid = null;
                        this.password = null;
                        return false;
                }
        }
// add a new student to the student table
public String addStudent(int id, String street, int zip_code, String
degree1,String degree2,
                String thesis, int no_of_hours) throws
SQLException,FileNotFoundException {
        getDBConnection();

//create a statement object, the first argument "TYPE_SCROLL_SENSITIVE"
//indicates the type of a ResultSet object: scrollable and generally sensitive
//to changes made by others; the second argument "CONCUR_READ_ONLY" indicates
//the concurrency mode of the ResultSet object: the concurrency mode that may
//not be updated. Both of the arguments are optional.

        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
        sqlString = "INSERT INTO student VALUES (" + id + ",'" + street + "'," +
zip_code + ")";
        System.out.println("\nInserting: " + sqlString);
        stmt.execute(sqlString); //execute the query

        // got here, it is time to return the success status
        return "success";
        }
}
```

Insert_student.jsp //create a web page for a student insertion (see fig. 7)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1252"/>
        <title>insert a student record</title>
</head>
<body>
        <h2 align="center">
                CS5513 Web Application
        </h2>
```

```
	<h3 align="center">
		Insert a New Student Record
	</h3>
	<h3 align="center">
		<jsp:useBean id="empsbean" class="oracle_jsp_test.DataHandler"
scope="session"/>
	</h3>

<!-- Pass the form info to insert_student_action.jsp program and execute it
when the button “Insert Student” is pressed -->

	<form action="insert_student_action.jsp">
		<div align="center">
			<p> </p>
			<table cellspacing="2" cellpadding="3" border="1"
width="369">
				<tr>
					<td width="38%">
						<strong>Student ID:</strong>
					</td>
					<td width="62%">
						<input type="text" name="id"/>
					</td>
				</tr>
				<tr>
					<td width="38%">
						<strong>Address Street:</strong>
					</td>
					<td width="62%">
						<input type="text" name="street"/>
					</td>
				</tr>
				<tr>
					<td width="38%">
						<strong>Zip Code:</strong>
					</td>
					<td width="62%">
						<input type="text" name="zip_code"/>
					</td>
				</tr>
				<tr>
					<td width="38%">
						<strong>Degree#1:</strong>
					</td>
					<td width="62%">
						<input type="text" name="degree1"/>
					</td>
				</tr>
				<tr>
					<td width="38%">
						<strong>Degree#2:</strong>
					</td>
					<td width="62%">
						<input type="text" name="degree2"/>
					</td>
```

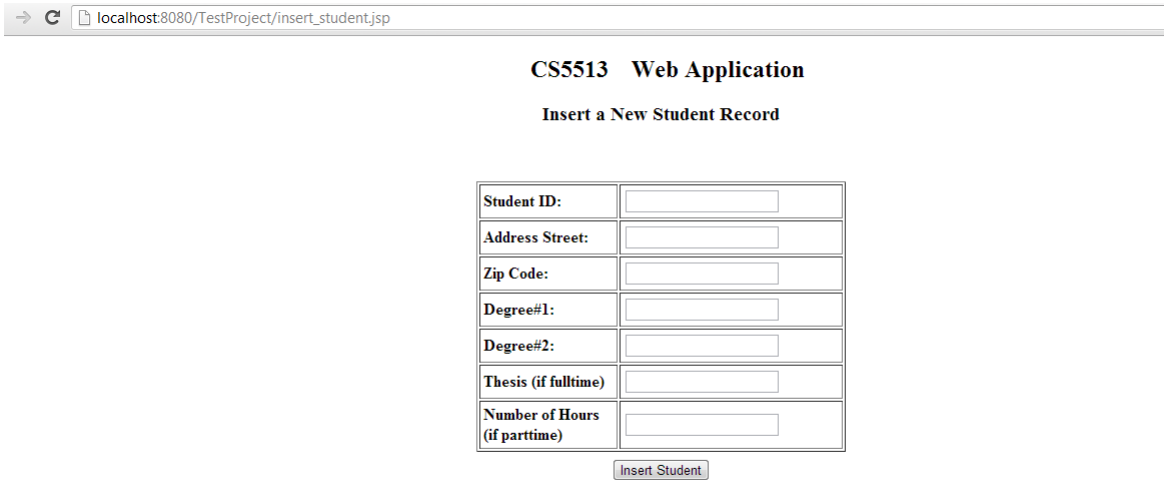
```
                        </tr>
                        <tr>
                                <td width="38%">
                                        <strong>Thesis (if fulltime)</strong>
                                </td>
                                <td width="62%">
                                        <input type="text" name="thesis"/>
                                </td>
                        </tr>
                        <tr>
                                <td width="38%">
                                        <strong>Number of Hours (if
parttime)</strong>
                                </td>
                                <td width="62%">
                                        <input type="text" name="no_of_hours"/>
                                </td>
                        </tr>
                </table>
                <table cellspacing="3" cellpadding="2" border="0">
                        <tr>
                                <td>
                                        <input type="submit" value="Insert
Student"/>
                                </td>
                        </tr>
                </table>
                <p>
                         
                </p>
        </div>
    </form>
    </body>
</html>
```

Insert_student_action.jsp //insert a student into the student table

```
<%@page import="oracle_jsp_test.DataHandler"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <%
        DataHandler handler = new DataHandler();
        handler.addStudent(Integer.parseInt(request.getParameter("id")),
                request.getParameter("street"),
                Integer.parseInt(request.getParameter("zip_code")),
                request.getParameter("degree1"),
                request.getParameter("degree2"),
```

```
                        request.getParameter("thesis"),
        Integer.parseInt(request.getParameter("no_of_hours")));
        %>
</body>
</html>
```

Running the insert_student.jsp file on server should produce the below screenshot:



Example 2: the usage of JSP and JDBC for retrieving data from tables.

Step#1: SQL is used to create tables

Skipped because it is done in Example 1.

Step#2: JSP and JDBC are used to retrieve information of students whose id is 123 from the table "student"

File Name: DataHandler.java: at the end of function public String addStudent(…), ONLY add the following new functions getStudentByID(int id) and getAllStudents()

```
public ResultSet getStudentByID(int id) throws SQLException {
            getDBConnection();
            stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
            query = "SELECT S.id,street,zip_code,degree, F.thesis,
P.no_of_hours " +
                        "FROM student S, degree D, student_fulltime F,
student_parttime P " +
                        "WHERE S.id = D.id(+) " +
                        "AND S.id = F.id(+) " +
                        "AND S.id = P.id(+) " +
                        "AND S.id =" + id;
            System.out.println("\nExecuting query: " + query);
```

```
            //execute the query and store the result in the ResultSet rset
            rset = stmt.executeQuery(query);
            return rset;
      }

public ResultSet getAllStudents() throws SQLException {
      getDBConnection();
      stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
      query = "SELECT S.id,street,zip_code,degree, F.thesis, P.no_of_hours " +
                  "FROM student S, degree D, student_fulltime F,
student_parttime P " +
                  "WHERE S.id = D.id(+) " +
                  "AND S.id = F.id(+) " +
                  "AND S.id = P.id(+) ";
      System.out.println("\nExecuting query: " + query);
      rset = stmt.executeQuery(query);
      return rset;
      }
```

Student.jsp //create a web page for the retrieval query in example 2 (see fig. 8)

```
<%@page import="java.sql.ResultSet"%>
<%@page import="oracle_jsp_test.DataHandler"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-
1252"/>
    <title>Students</title>
</head>
<body>
    <h2 align="center">
        CS5513 Web Application
    </h2>
    <h3 align="center">
        Student Information
    </h3>
    <!--
        <jsp:useBean id="student_bean" class="oracle_jsp_test.DataHandler"
scope="session">
        </jsp:useBean>
    -->
    <form action="student.jsp">
    <div align="center">
    Filter by Student's ID
    <input type="text" name="studentID"/>
    <input type="submit" value="Filter"/>
    </div>
    </form>
```

```
    <p>
        <%
            DataHandler handler = new DataHandler();
            ResultSet rset;
            String query = request.getParameter("studentID");

//when student id is entered and submit button is pressed
            if (query != null)
                rset = handler.getStudentByID(Integer.parseInt(query));

//when the page is first loaded, there is no info in the student id box and
//the query is null since no info submission, then display all students.
            else
                rset = student_bean.getAllStudents();
        %>
    </p>
    <div align="center">
    <table cellspacing="2" cellpadding="3" border="1">
    <tr>
        <td align="center">
            <h4>ID</h4>
        </td>
        <td align="center">
            <h4>Street</h4>
        </td>
        <td align="center">
            <h4>Zip code</h4>
        </td>
        <td align="center">
            <h4>Degree</h4>
        </td>
        <td align="center">
            <h4>Thesis (if fulltime)</h4>
        </td>
        <td align="center">
            <h4>Number of Hours (if part time)</h4>
        </td>
        <td> </td>
    </tr>
    <%
    while (rset.next ())
    {
        out.println("<tr>");
        out.println("<td align=\"center\">" +
        rset.getInt("id") + "</td><td align=\"center\"> " +
        rset.getString("street") + "</td><td align=\"center\"> " +
        rset.getInt("zip_code") + "</td><td align=\"center\"> " +
        rset.getString("degree") + "</td><td align=\"center\"> " +
        rset.getString("thesis") + "</td><td align=\"center\"> " +
        rset.getInt("no_of_hours") + "</td><td align=\"center\"><a
href=\"delete_action.jsp?id=" +
        rset.getInt("id") + "\">Delete</a> <a href=\"edit_student.jsp?id=" +
        rset.getInt("id") + "\">Edit</a></td>");
        out.println("</tr>");
    }
```

```
    %>
    </table>
</body>
</html>
```

Running the student.jsp file on server should produce the below screenshot: