# **Installing a Web Server that supports servlets**

1. Install a sample web server, which supports Servlets/JSPs. A light weight web server is Apache Tomcat server. You can get the server from <http://tomcat.apache.org/>

A comparison of different web servers is available at Wikipedia at <http://en.wikipedia.org/wiki/Comparison_of_web_server_software>

1. Follow the installation directions and install the web server on your machine. Also check how to start, stop, restart your web server.
2. Check that your web server is working by going to an appropriate website based on your configuration – if you used the default configuration, then you want to check the webpage: <http://localhost:8080/>

If you are using Eclipse IDE and Tomcat webserver, there are alternate ways of configuring your Eclipse to work with Tomcat server. If this is your configuration, follow the steps explained in

<http://courses.coreservlets.com/Course-Materials/pdf/csajsp2/01-Overview-and-Setup.pdf>

<http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat-7-with-eclipse.html> The only modification is that the port is specified to be 8080, therefore you will use the URL <http://localhost:8080/> for testing.

Also test the sample app test-app.zip given and see that it works.

Main aspects to note:

1. Make sure you choose Java SDK and not the java that comes with eclipse (see slide 21 on <http://courses.coreservlets.com/Course-Materials/pdf/csajsp2/01-Overview-and-Setup.pdf>)
2. For writing your first servlet, check <http://courses.coreservlets.com/Course-Materials/pdf/csajsp2/02-Servlet-Basics.pdf>
3. Note the mapping specified using @WebServlet

**Example Servlet**

We will use Eclipse IDE and Servlets. Follow the instructions given: (Note that we need Eclipse IDE for Java EE Developers for developing servlets).

Main steps include (look at <http://courses.coreservlets.com/Course-Materials/pdf/csajsp2/01-Overview-and-Setup.pdf>)

1. Create a new Project (Dynamic Web Project).
2. Add code
3. After we write our code, we will deploy our project

Code for a simple servlet is given below. I created this java class file in a package called testServletPkg which was created in Java Resources:src as mentioned above.

**package** testServletPkg;

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.annotation.\*;

**import** javax.servlet.http.\*;

@WebServlet("/test")

**public** **class** TestServlet **extends** HttpServlet {

@Override

**public** **void** doGet (HttpServletRequest request,

HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<HTML>\n" +

"<HEAD><TITLE>Hello</TITLE></HEAD>\n" +

"<BODY BGCOLOR=\'WHITE\'>\n" +

"<H1>Hello</H1>\n");

out.println ("</BODY></HTML>");

}

}

Note the following:

1. The package declaration is because it is created inside the package mentioned above.
2. @WebServlet says how the servlet can be accessed from a web browser. Suppose our project is called myServletTest and our web servet can be accessed as: <http://localhost:8080/>, then our servlet can be accessed as <http://localhost:8080/myServletTest/test>
3. Your servlet must extend HttpServlet.
4. @Override is optional, and can be omitted.
5. doGet method describes what is the response from the web server to the client request. The code specifies that the response is a HTML file with Hello in the title, and a heading Hello in the body. I presume that you know HTML already. Otherwise find a simple HTML tutorial. The doGet method must have exactly the same parameters and exceptions as above.

# **HTML Forms and Oracle JDBC**

For HTML forms, check a tutorial. Also for using HTML forms with servlets, check <http://courses.coreservlets.com/Course-Materials/pdf/csajsp2/03-Form-Data.pdf>. We will use POST method below. We will also use a database in the backend.

We will create a simple HTML form. Let us create a new Dynamic Web Project called, testFormProj in Eclipse. We will create HTML files in Web Content. Let us create a new HTML file called testFormHTML.html – you can right click on Web Content and say that you want a new HTML file. Choose an appropriate template (I chose the default template – HTML transitional).

Now we will edit the testFormHTML.html file as shown below (we will use post method in HTTP to submit the form data to the servlet).

<!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>Testing HTML Form with Oracle</title>

</head>

<body>

<H1>HTML Form Test</H1>

<FORM METHOD=*"POST"* ACTION=*"http://localhost:8080/testFormProj/testFormServlet"*>

<input type=*"text"* name=*"sNumber"*/>&nbsp;sNumber<br>

<input type=*"text"* name=*"sName"*/>&nbsp;sName<br>

<input type=*"text"* name=*"professor"*/>&nbsp;professor<br>

<p><INPUT TYPE=*SUBMIT* value=*"Submit"*/></p>

</FORM>

</body>

</html>

Note the following: The FORM tag with METHOD, and ACTION – the ACTION specifies the servlet which we will write below.

Also note the element: INPUT type = SUBMIT. This brings up the Submit button on your form, and when you click Submit the form data will be sent to the servlet.

Now deploy the project to Tomcat (as we did before), and restart Tomcat. Now open the web page: <http://localhost:8080/testFormProj/testFormHTML.html> and see that your HTML file is opening fine.

We will now create the Java servlet file to handle the form. We will create a package in Java Resources:src, let me call it as testFormPkg. I will create a java file called TestFormServlet within this package. Once you have compiled the Java class restart the web server.

We need to do one more thing, which is to add oracle’s ojdbc6.jar to Tomcat launch configuration. For this, right click on Tomcat v 7.0 Server at localhost and select Open. Now select Open launch configuration. Here select Classpath, User Entries, Add External Jars and add ojdbc6.jar (the last couple of steps are exactly like how you add the jdbc driver to your project).

Note that the code only shows a simple functionality: inserting values in a form into a table (of course we need the table in the database for this to work).

**package** testFormPkg;

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**import** javax.servlet.annotation.\*;

**import** java.sql.\*;

@WebServlet ("/testFormServlet")

**public** **class** TestFormServlet **extends** HttpServlet {

Connection conn = **null**;

Statement stmt = **null**;

**public** **void** doPost(HttpServletRequest request,

HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<HTML>\n" +

"<HEAD><TITLE>Test Form</TITLE></HEAD>\n" +

"<BODY BGCOLOR=\'WHITE\'>\n" +

"<H1>Hello</H1>\n");

**try** {

getDBConnection (out);

String sNumber = request.getParameter ("sNumber");

String sName = request.getParameter ("sName");

String professor = request.getParameter ("professor");

String sql = "INSERT INTO Student (sNumber, sName, professor)"

+ " VALUES (" + sNumber + ",'" + sName + "','" + professor + "')";

out.println (sql);

out.println ("\n<br/>");

stmt.executeUpdate (sql);

} **catch** (Exception e) {

out.println ("some error happened");

out.println (e.getMessage());

}

out.println ("</BODY></HTML>");

}

**void** getDBConnection (PrintWriter out) **throws** SQLException {

**try** {

conn = DriverManager.*getConnection*

("jdbc:oracle:thin:@db.csep.umflint.edu:1521:csep",

"mmani", "mmani");

out.println("Got connected");

stmt = conn.createStatement ();

} **catch** (SQLException e) { **throw** e; }

}

}

If you are interested in using other tools such as JSP, with servlets, please read up on them at <http://courses.coreservlets.com/Course-Materials/csajsp2.html>

<http://courses.coreservlets.com/Course-Materials/msajsp.html>