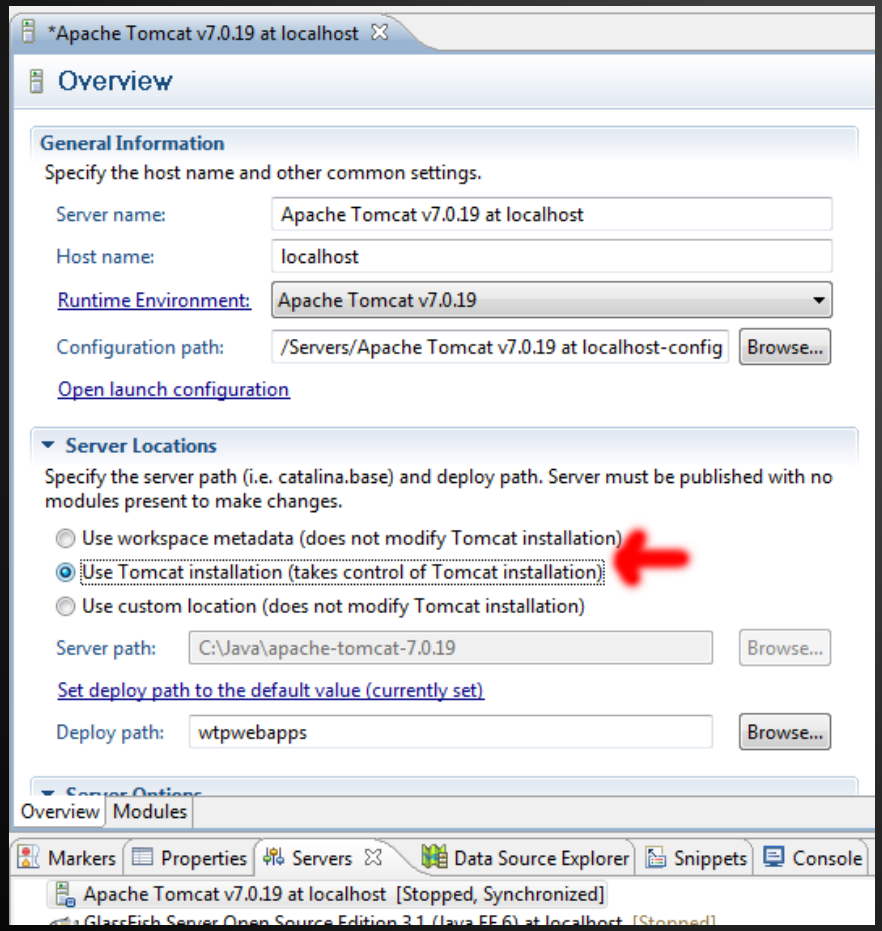
Java EE

J2EE Day 2

Tomcat Configuration

Double-click on Servers tab and in the window that opens, make sure you select Use Tomcat installation (takes control of Tomcat installation)



Create Dynamic Web Project

File -> New -> Dynamic Web Project

Java Resources holds the java source files

Libraries contains Apache Tomcat Libraries

Web App Libraries contains libraries we need

Create Servlet

META-INF, WEB-INF hold essential config files

Right-click on Project -> File -> New -> Servlet

Java Package : com.example.servletdemo

Class Name : ServletDemo

Click Next

Create Servlet Step2

Enter Name : ServletDemo

Change URL mapping to lowercase :

/servletdemo

Click Next

Create Servlet Step3 - Specify Modifiers

Under which method stubs would you like to create?

Uncheck the doPost and doGet methods and select the service method

Click Finish

web.xml

The following block will get added to web.xml

```
<servlet>
        <description></description>
        <display-name>ServletsDemo</display-name>
        <servlet-name>ServletsDemo</servlet-name>
        <servlet-class>com.example.servletsjspexample.ServletsDemo</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>ServletsDemo</servlet-name>
        <url-pattern>/servletsdemo</url-pattern>
</servlet-mapping>
```

What is a Servlet

A simple java class which has the ability to communicate via HTTP protocol by extending the HTTPServlet class.

It can request and send messages back to the client.

Modify ServletDemo

Create an index.html in WebContent

Remove the constructor from ServletsDemo

Override the service method with the following code :

```
PrintWriter out = response.getWriter();

out.println("This is a Servlet");
```

Add .jsp file

In WebContent

File -> New -> JSP File

Name it index.jsp and add a form

```
<form id="contactForm" action="servlettest" method=”post”>
        <table border=”0”>
        <tr>
        <td> First Name : </td> <td><input type=”text” name=”firstname”/></td>
        </tr>
        <tr>
        <td>Last Name : </td> <td><input type=”text” name=”lastname”/></td>
        </tr>
        <tr>
        <td colspan=”2”> <input type=”submit” value=”submit”/></td>
        </tr>
        </table>
</form>
```

Run on Server

Right-click on Server and say run on server

Click Next -> Finish

Modifying Servlet for MVC Pattern

Add RequestDispatcher to ServletsDemo.java

```
if ((request.getParameter("firstname") == null)
                || (request.getParameter("lastname") == null))
 {

        getServletContext().getRequestDispatcher("/index.jsp").forward(
                        request, response);
 }
```

Refresh

http://localhost:8080/ServletsDemo/servletsdemo

Create output.jsp

Right-click on WebContent

File -> New -> JSP -> output.jsp

Use session scope variable as follows :

```
<h1>Your first and last name is : </h1>
<%
String firstName = (String) session.getAttribute("firstname");

String lastName = (String) session.getAttribute("lastname");
%>
```

Call requestDispatcher in ServletTest.java again as follows :

```
request.setAttribute("firstname", firstName);
request.setAttribute("lastname", lastName);            getServletContext().getRequestDispatcher("/output.jsp").forward(request, response);
```

Maven Project Step1

New -> Project -> Other -> Maven -> Maven Project

Check the checkbox Create a Simple Project

Use defaults

Maven Project Step2

Group Id : com.example.j2eeapp

Artifact Id : j2eeapplication

Version : default

Packaging : war

Name : J2EE Application

Click Finish

Right-click on Project -> Select Spring Tools -> Add Spring Project Nature

Now it’s a Spring Framework project

Project Structure

Spring Elements -> Contains Spring configuration

The other 4 folders are quickpaths to java folders which contain java source files

Resources folder contains additional resources and configurations

src/test/java and src/test/resources contains unit tests

We can delete the test folders to keep the folder structure clean

Change JRE System Library to 1.6 by right-clicking -> Properties -> select 1.6

Delete test under src

src/main/webapp is the public web folder

target folder will contain the compiled and packaged project

The web-archive that can be deployed on the web-server and run

pom is the project-object-model

The pom.xml can be edited

maven is more clever than ant and much more robust

web.xml in WEB-INF

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
    <display-name>Demo</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Compile and package the project

Right-click project -> Run as -> Maven...

Maven clean means delete target folder and all the contents with all packages

Maven install includes other maven commands such as build and package

At the end of it, we get the web-archives

Automate loading of war in Tomcat

Undeploy the demo maven project from

Tomcat Web Application Manager

In Servers ->

Tomcat v6.0 Server at localhost-config

->

In server.xml

just before </Host> tag add the following configuration :

<Context docBase="/Users/das/.m2/repository/Demo/com.example.demo/0.0.1-SNAPSHOT/com.example.demo-0.0.1-SNAPSHOT.war" path="/ServletsDemo" reloadable="true"/>

to tell Tomcat to load the application directly from where it is compiled and assembled

Project Setup using Maven

Right-click on Project -> Maven -> Add Dependency

Type hibernate-core into the Enter groupId field and the Search Results will show all the jars

Look for org.hibernate Select the latest Final jar

Configure pom.xml

You will see that the pom.xml has changed to reflect the hibernate-core library

```
<dependency>
 <groupId>org.hibernate</groupId>
 <artifactId>hibernate-core</artifactId>
 <version>4.2.4.Final</version>
</dependency>
```

Expand Maven Dependencies to see what was added

Configure pom.xml

Add hibernate-validator to project

Add other dependencies in pom.xml

Oracle and Primefaces will show errors