Uttara InfoSolutions

www.uttarainfo.com

Servlets Lab 1

0) Download tomcat.zip and unzip it to any folder

1) Run Eclipse -> File -> New -> Dynamic web project -> Give name as FirstApp -> choose web module version 2.5.

2) Drag and drop servlet-api.jar from Tomcats lib folder to WEB-INF/lib in FirstApp

3) Right click on FirstApp and create -> new -> HTML -> Test.html

Put some static content. Verify where Test.html is placed in the project structure. Is it within WebContent? inside or outside WEB-INF?

4) Right click on Test.html and run-as -> on server. Link server if it doesn't already exist by pointing to the unzipped folder of tomcat.

5) Add a header element, image element and verify by refreshing on the browser window if the content is being displayed. Right click on the browser window and click on View Source and verify if html is the same. Do you understand how the Test.html got served by the web server? Recollect the steps. Stop the server. Check if you can refresh the content in the browser window. Do you see the html content now? Why not?

Header element -> <h1> Heading </h1>, Image element -> <img src="name.jpg"/>

6) Right click on project and create a Sample.html. Verify if the file is getting created next to Test.html. In Test.html, create a hyperlink to link to Sample.html like this:

<a href ="Sample.html>Click to test sample</a>

What kind of linking is this? How does browser know where to send request when you click on hyperlink?

7) Right click on FirstApp -> create new -> servlet -> FirstServlet. Put in package test.

8) Add in doGet(..)

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

java.util.Date dt = new java.util.Date();

out.println("Hello "+dt);
```

9) Right click on DateServlet -> run on -> server -> link to tomcat base folder (parent of bin) and then finish. Change url-pattern to /getDate in web.xml. Right click on DateServlet, run on server. Now create a hyperlink in Test.html and href="getDate". Now see if clicking the link will trigger the servlet.

10) Override init(),destroy(), put SOPs in constructor, init(), destroy(), doXXX() methods. Right click on servlet -> run-as on server and test.

11) Create a HTML with 2 user input elements (name and age) and 1 submit. Take a name and age as input. Submit to a servlet (url-pattern in action attribute with method='get'). In servlet, access the user inputs (request.getParameter("name of input element"), respond with a welcome message depending on age of user. Do user input validations. Change method action to 'post' and see if doPost() is getting called.

Methods to use in doGet(request,response) are:

a) String val = request.getParameter("<name of input element>");

b) PrintWriter pw = response.getWriter();

c) pw.write("<html>…");

12) Build a Counter using Servlet (CounterApp).In HomePage.html create a hyperlink to send request to "count" url-pattern and register it as welcome-file in web.xml. Right click on project and verify if sending a request to the app will respond with the Home Page. Count requests that are sent for url-pattern /count. The output should be "The Count is <number>". Per refresh the count should get incremented. What type of variable will you create for holding the count? Create it as an instance variable. Override init() and destroy() in CountServlet and ensure that in destroy, the count is getting written into a file named count.txt with a single line

(count = value). Read this value in init() and initialise the value to the count instance variable. Test by restarting the server, whether the count is getting incremented per request or not.