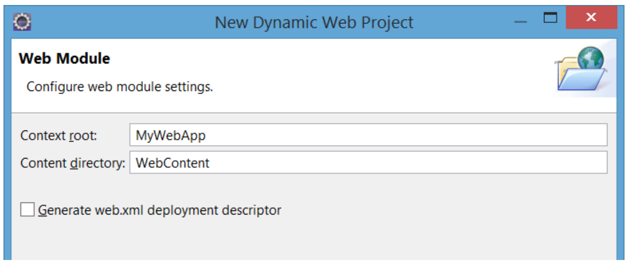
Exercise 12 – Web Applications

Objective

The objective of this exercise is to build a very simple web application that is able to display objects as part of a JSP page. This should give you some of the building blocks for looking into web applications in general for the future.

Overview

1. Start by creating a new dynamic web project in eclipse
    a. Give it any name you like
    b. In the target runtime box click “New Runtime”
    c. Look for Tomcat7 under the apache option
        i. You don’t have to use Tomcat as a webserver, glassfish is a good alternative. The built-in one for J2EE libraries can have issues with setup.
    d. Give it a name and click on the “Download and install” button. This will install a new copy of tomcat on your machine for you.
    e. Click Finish, you can now select this new server as your default runtime
    f. Click next twice. The window you are now on will allow you to set your context root. This is where the web application will be hosted on the server. You can change this to anything you like, or leave it as the default value.


New Dynamic Web Project

Web Module

Configure web module settings.

Context root: MyWebApp

Content directory: WebContent

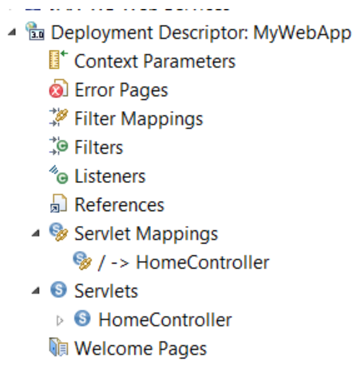
☐ Generate web.xml deployment descriptor

    g. Click finish
2. Now we have a new web project, you can view it using the Java EE perspective if you want. This will change the layout of eclipse to one optimised for web application development.
3. Create a new Servlet by right clicking the project and selecting Servlet from the new menu
    a. This will automatically generate the boiler plate code for you

b. Convention is to name the servlets after the pages they control. For the default context we use “HomeController”.

c. Change the `@WebServlet(“/HomeController”)` line to read `@WebServlet(“/”)`

i. This will ensure that any request on the root context will be handled by this file. You can see web addresses are mapped and what controllers are handling them in the eclipse menu to the side under “Deployment Descriptor”



4. Have the doGet() and doPost() methods call a processRequest() method. Make sure to pass the request and response parameters to the processRequest method so we still have access to them
5. In the processRequest(request, response) method print “Hello world” to the screen using the response writer.
6. Run your web application on the server, make sure you select the server we created earlier to run on.
7. Now we want to change the home controller to return a JSP page instead. Create a new JSP page in the WebContent folder called home.jsp. Add some text to the webpage so you can identify it as yours
8. In the HomeController respond to the request by forwarding it to the JSP Page. Run the project on the server again to check that the request is being forwarded to the right place.
9. Finally, we want to look at adding objects to the response. Create a new Java file called “User.java”

   a. Implement a simple User. A user should have an int ID, a string name and a string email. Create a constructor, getters, setters and a toString method for the class.

   b. In the home controller, create a new user object before you forward the response

c. Add the user object to the *request* object

d. These should now be forwarded to home.jsp when it is called

e. In home.jsp add some code to get the values of the object from the request object.

   Hint: remember we can add java code to jsp files by using ${ } the object name will be the same as the one you assigned it when adding it to the request, not the one in the home controller!

- If you have time…
  - Have a look at the JSTL library, this gives us more options for control flow in JSP pages
  - Try adding an arraylist of users to the request object and printing them out using the <c: foreach …> expression
    - You will need to download and add the JSTL library to your eclipse project
    - You will also need to export it as part of the Java Build settings. These can be found by right clicking the project, going to properties, then under Java Build Path in the Order and Export tab.

Glossary of key terms

Web applications

Web applications run over the internet responding to requests from desktop applications or browsers. These applications exchange messages, deliver and respond to information.

Servlet

It is a Java web component which can serve the dynamic content

Java Server Pages (JSP)

It is another form of servlet which contains HTML and JSP tags. JSP separates the presentation login from the application logic

Webservers

Webservers are the machines and programs that service requests coming into them