LS/EECS Building Ecommerce Applications

Lab 1: OsapCalc-V.1

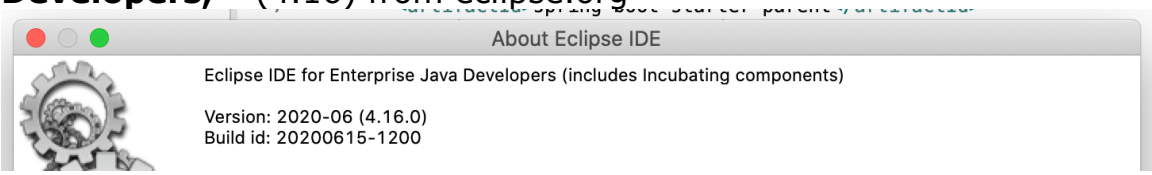
(upload it by the deadline)

Objectives

-understand the development environment
-setup the development environment
-deploy and run simple html, jsp and servlets
-understand the APIs of the Servlet's HTTP request and response
-understand the servlet session
-develop a simple client-server application

Task A: Setup and test the Dev and Runtime Environment (on Lab computers, skip 1-3)

1. Download and install Eclipse **"Eclipse IDE for Java Enterprise Developers,"** (4.16) from eclipse.org



   a. Need to install at least Java 8 on your computer, (JRE 1.8) https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html
   b. Start Eclipse
   c. Explore the Perspectives
   d. Run your first Java program, Hello.java
2. Install Apache Tomcat (v8.5), https://tomcat.apache.org/tomcat-8.5-doc/index.html
3. Download and unzip Apache derby version 10.12.1.0 https://db.apache.org/derby/releases/release-10.12.1.1.cgi
4. Configure Eclipse with Tomcat. In Java EE perspective, explore the Servers tab;
   a. Create a Web server, select Apache Tomcat 8.5 (point to the directory where you unzipped Tomcat)
5. Create your first Web Dynamic Project, **OsapCalc-v1**
   a. The project should have a **web.xml** file (last page of the create wizard)
   b. Understand the structure of a Java EE project
6. Test your environment by writing, deploying, running, your first web pages
   a. Html page: HelloFromHTML.html
   b. Jsp page: HelloFromJSP.jsp

c. Java Faces: HelloFromFaces.xhtml

Use pop-up menu to create the pages. For jsp/faces files, use the "Create JSP" menu, select the templates for xml for each.
In each page, add some HTML markup to say "Hello…"
Then use "Run on Server" pop-up menu. Notice the Server tab.

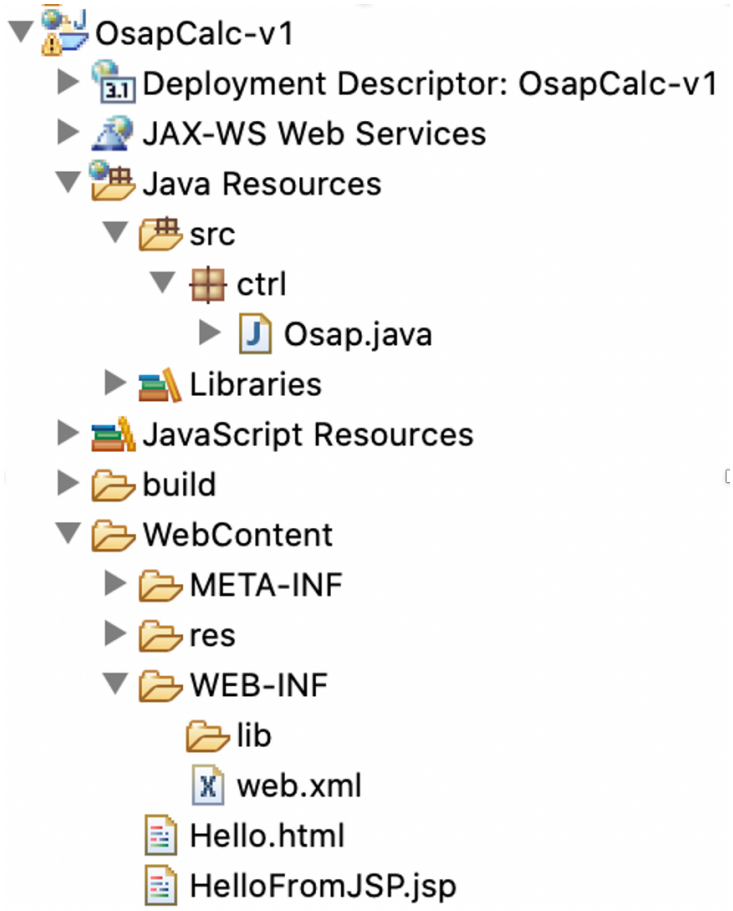


Fig. 1 A typical Java EE project in Eclipse: src folder contains the java files, including servlets; WebContent folder contains the html, jsp pages.

Task B: Explore the servlet, request and response API

7. Create your first Servlet, **Osap.java** to write "Hello" to the console (on server). To do that, after you create the servlet, in doGet method, add this line:

```
System.out.println("Hello, Got a GET request from Osap!");
```

- Change the annotation of your servlet so it responds to /Osap/* as well as /Osap.

Notes:
When you create the servlet Osap.java, it is created with this annotation,
@WebServlet({ "/Osap" }).
You can add additional paths to the above annotation.

- Invoke servlet at this URL: http://localhost:8080/OsapCac-v1/Osap
  - You can select the servlet, and then "Run on Server" from pop-up menu
  - Note that with the annotation "/Osap/*" you can add any path and query to your invocation. Try it!
- You can use an external browser, like Chrome, Firefox to access the servlet and access the servlet from there.

7. By exploring the API of the servlet, add code to Osap.java to do the following:

- Return the client ip and client port to the browser
  - How would you implement an IP-filtering firewall?
- Return the request's http protocol and method
- Return the request path
- Pass a query string in the URL and return the entire query string sent by the client (check the lecture slides how you pass parameters)
  - Use the getParameter of request to extract a named request parameter
  - Return the value of a parameter "foo"
- Embed spaces in the query string and note the URL encoding in the developer tool.

Notes on how to do it:
To reply to the client (browser) with a message, in the servlet *Osap.java*, *doGet()* method, you need to obtain a *writer* from *response* object and write a string…

```
response.setContentType("text/plain");
Writer resOut = response.getWriter();
resOut.write("Hello, World!\n");
```

To explore different elements of the http request message, use *get* methods on *request* object…

```
String clientIP = request.getRemoteAddr();
resOut.write("Client IP: " + clientIP + "\n");
 String clientQueryString = request.getQueryString();
String foo = request.getParameter("foo");
resOut.write("Query Param foo=" + foo + "\n");
```

At this point, you servlet response should look like in the caption below

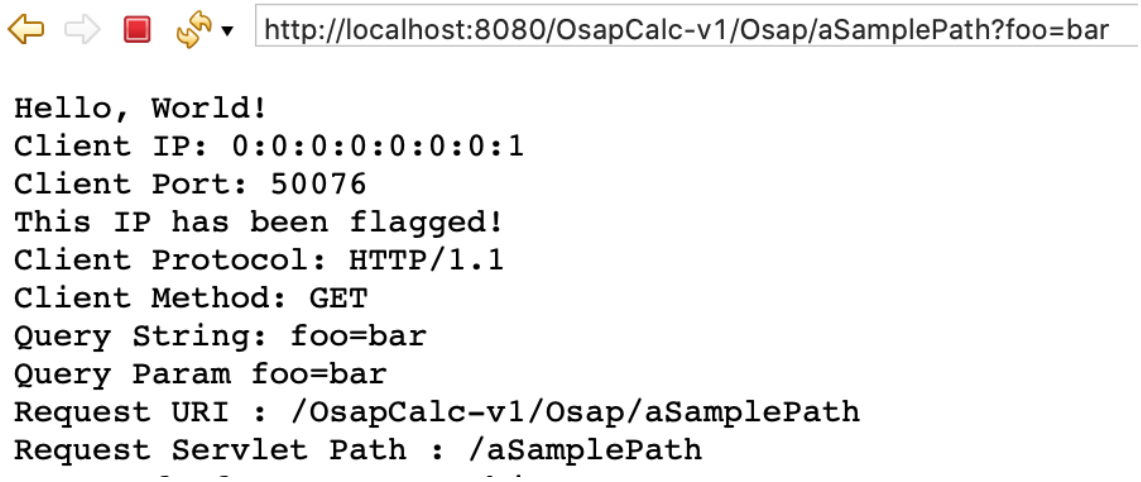


Fig. 2. Servlet response.

Task C: Understanding Java EE Application Descriptor and ServletContext object

8. Deployment Descriptor consists of the file web.xml. The content of this file is accessible from your application, through the *ServletContext* class instances.

In this task, you

- Add Osap to the welcome file list and test it.
- Add a *context parameter* and verify that you can access it in the servlet.
- Explore the APIs of ServletContext class
- Add an error-page for error-code 404 and point it at location */res/my404.jspx*. Test by visiting a non-existent page.
- Add an error-page for exception-type java.lang.Exception and point it at location */res/myException.jspx*. Test by triggering any exception in your servlet.

Notes on how to do it:

You should have a *web.xml file* in your project.

Edit *web.xml* using the xml editor that opens the file

- add Osap to the file list, e.g.

```
<welcome-file-list>
  <welcome-file>Osap</welcome-file>
  <welcome-file>index.html</welcome-file>
…..
```

- Add context parameter children. The parameters have name and value

  Ex:

```
<context-param>
   <param-name>applicationName</param-name>
   <param-value>OSAP Calculator 2020</param-value>
 </context-param>
 <context-param>
   <param-name>applicantName</param-name>
   <param-value>Joe Doe</param-value>
 </context-param>

<context-param>
   <param-name>principal</param-name>
   <param-value>1000</param-value>
 </context-param>
```

Note: web.xml is loaded when the application is deployed, you might need to restart the server after you edit it.
-you can access the ServletContext from doGet with

ServletContext context= this.getServletContext();

You can read the context parameters from within the servlet, using the method getServletContext().getInitParameter("parameterName")//it returns a String. This might need to be converted, e.g.

```
principal = Double.parseDouble
(this.getServletContext().getInitParameter("principal"));
```

- Retrieve the following parameters and return them to client
  -applicantName;
  -applicationName;
  -principal

- Explore the following methods of the ServletContext: getContextPath() and getRealPath(). To better understand them, check what they return as in the example below

```
String contextPath=context.getContextPath();
String realPath=context.getRealPath("Osap");
```

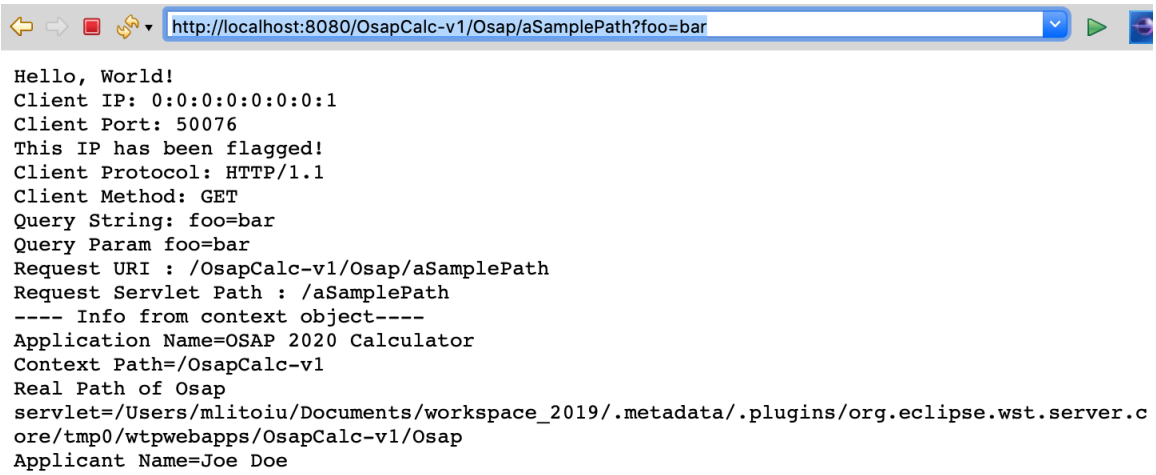


Fig. 3. Servlet response with context info.

9. Error pages… (see Fig 1.)

a) you have to create them in webContent/res folder in your project

b) you have to edit the web.xml and add error page children, like these:

```
<error-page>
    <error-code>404</error-code>
    <location>/res/my404.jspx</location>
  </error-page>
  <error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/res/myException.jspx</location>
  </error-page>
```

c) then create the pages as shown in Fig. 1. The pages should display some meaningful content when triggered.

Task D: OSAP Calculator and Parameters

10. Modify Osap.java to compute the OSAP monthly payments:

- Extract the values of the following request parameters: principal, period, and interest (the parameters come via URL). You can assume that any client-supplied value is valid; i.e. do not validate.
- If a parameter is missing, supply a default value obtained from a context parameter. This way, one can change the defaults w/o recompiling the servlet.
- Compute the monthly payment using the formula: $(r/12)*A/[1 - (1+ (r/12))^{-n}]$, where r is the annual interest rate, A is the present value (principal), and n is the period measured in months.
- Send the computed payment with an appropriate message to the client but format the amount so it is rounded to the nearest cent.

Notes:

-You can pass the parameters to the servlet in the url of the get method
Ex:
http://localhost:8080/Osap?principal=10000&interest=10&period=24
-In the servlet, doGet method, you retrieve the parameters using request.getParameter("parameterName") method
-make sure you have the default values for principal, interest and period in the context-param of the web.xml…and use them if no parameters are passed in the query.

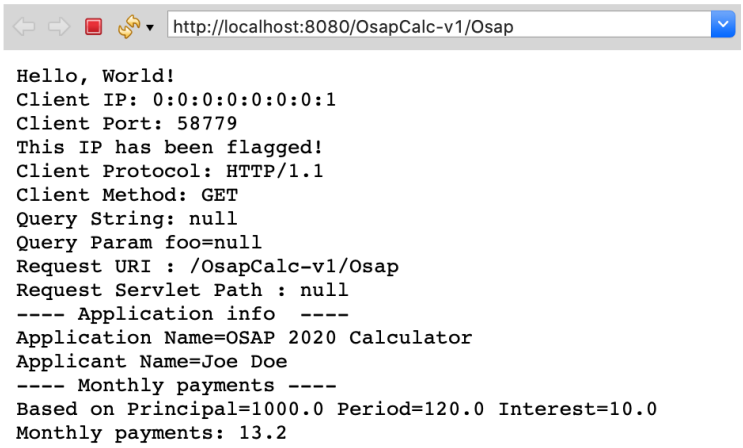


Fig 4. A snapshot of the output in the browser. When invoked with no query string and no path, uses the default values from web.xml

http://localhost:8080/OsapCalc-v1/Osap?principal=2000&interest=12&period=140

```
Hello, World!
Client IP: 0:0:0:0:0:0:0:1
Client Port: 55895
This IP has been flagged!
Client Protocol: HTTP/1.1
Client Method: GET
Query String: principal=2000&interest=12&period=140
Query Param foo=null
Request URI : /OsapCalc-v1/Osap
Request Servlet Path : null
---- Info from context object----
Application Name=OSAP 2020 Calculator
Context Path=/OsapCalc-v1
Real Path of Osap
servlet=/Users/mlitoiu/Documents/workspace_2019/.metadata/.plugins/org.eclipse.wst.server.c
ore/tmp0/wtpwebapps/OsapCalc-v1/Osap
Applicant Name=Joe Doe
---- Monthly payments ----
Based on Principal=2000.0 Period=140.0 Interest=12.0
Monthly payments: 26.6
```

Fig 5. A snapshot of the browser wen invoked with a query string and a path.

Task E: Remembering data, Sessions (optional)

Can you make you application to remember the data you submitted in the last invocation?
Hint: use the HttpSession object associated with your application
-from your request object, getSession()
-use setAttribute(), getAttribute methods to handle the parameters.

Below is a sample of

http://localhost:8080/OsapCalc-v1/Osap

```
Hello, World!
Client IP: 0:0:0:0:0:0:0:1
Client Port: 57617
This IP has been flagged!
Client Protocol: HTTP/1.1
Client Method: GET
Query String: null
Query Param foo=null
Request URI : /OsapCalc-v1/Osap
Request Servlet Path : null
---- Info from context object----
Application Name=OSAP 2020 Calculator
Context Path=/OsapCalc-v1
Real Path of Osap
servlet=/Users/mlitoiu/Documents/workspace_2019/.metadata/.plugins/org.eclipse.wst.server.c
ore/tmp0/wtpwebapps/OsapCalc-v1/Osap
Applicant Name=Joe Doe
---- Session info  ----
Application Name=OSAP 2020 Calculator
Applicant Name=null
principal=2000.0
period=140.0
interest=12.0
---- Monthly payments ----
Based on Principal=2000.0 Period=140.0 Interest=12.0
Monthly payments: 26.6
```

Fig 6. Invocation with no query, still the application remembers the last invocation parameters..