SOEN 387 – Fall 2019
Assignment-I

6% of the Final Grade

Due Date: October 8, 2019 – 11:59 PM
Group Assignment: A maximum of three per team

Servlet Hello World

Objectives
Understanding HttpServer, servlets, request and response objects, headers, content-types, and encoding.

Project Outline
In this assignment, you are asked to create a web project containing static html pages, a dynamic servlet, and an additional http server. The following directory structure outlines the web project:

+your_app_root
    +WEB-INF/…
    +src/your-java-packages-and-classes (i.e. com.soen387.hw1.servlets.*)
    +lib/any-external-jars (if applicable)
    +index.html (xhtml)
    +index2.html (xhtml)
    +css/css-file1.css
    +css/css-file2.css
+readme.md

Tasks
The assignment consists of the following five tasks:

1. Creating a Basic Servlet
2. Creating a simple static web client using HTML static files
3. Creating a simple HTTP Server
4. Using Curl
5. The readme.md file (required but not marked)

Task 1. Creating a Basic Servlet
Target topics: request and response objects, headers, content-types, and encoding.

Create a simple servlet that displays the following information:

- the request method,
- the client (request) headers, as well as
- the query string parameters

For instance, opening the url

"http://localhost:port/your-app/your-servlet?format=text&param1=val1&param2=val2",

in a browser may result in the following output:

```
Request Method: GET
Request Headers:
      -- display each header in a separate line
Query String:
      format: text
      param1: val1
      param2: val2
```

The servlet checks for the existence of the 'format' parameter and in case it is provided, it displays the above information in corresponding format:

- format=text: displays the information in plain text format, i.e. above example.

- format=html (default): formats the output using html. It displays the information using three html tables, i.e.:

Request Method:	GET

Request Headers:	
…	…

…

- format=xml: in this format, the output of the servlet is an xml document, i.e.

```
<response>
      <request-method>GET</request- method>
      <request-headers>
            <header name="...">...</header>
            <header name="...">...</header>
      </request-headers>
      <query-string>
            <format>xml</format>
            <param1>value1</param1>
            <param2>value2</param2>
      <query-string>
      .
      .
      .
</response>
```

Notes:

- Make sure your query string is properly decoded.
- In case the format is not specified, the html format is used by default.
- In case any other values is specified for the format, an appropriate server error is generated.
- In case of html / xml, make sure the output is escaped properly (html-encoded / xml-encoded).
  Hint: search for java utilities on how to do properly perform html encoding.
- Make sure the output content type is set properly.

Task 2. Creating a simple static web client using HTML static files.
Target topics: xhtml, html form, GET vs POST, using CSS.

Create two static html files namely index.html and index2.html. The html files are in xhtml format. They use separate css files. Both html files display a single html form which consists of the following entries:

- name: text box
- email: text box
- format: dropdown with four options:
  { not specified (blank), html, plain text, xml }

The file index.html is the default page and uses css-file1. The file index2.html is a secondary page that displays the above form using a different style (css-file2). Both files submit their form to the servlet in Task 1. The page index.html uses the GET method whereas the page index2.html uses POST. Both index files have hrefs to each other (form index.html the user may navigate to index2.html and vice versa).
The index pages are almost identical except for the following items:

- links to different css files
- form method (GET vs POST)
- the hyperlinks to the other page

They indeed must look different. Use your own design for implementing two different looks. You may use different fonts, colors, placements, etc.

Note: Using different styles must be solely done by css in external files.

Task 3. Creating a simple HTTP Server.
Target topics: java builtin http server, file io, try with resoures.

Using java HttpServer, create a simple server class that simulates the http server over the static content. The static content in here refers to the index as well as the css files. Your HttpServer upon receiving the request, outputs the content of the corresponding file to the output. For instance the following urls may be mapped to the static files:

/:	index.html	*content-type: text/html*
/index.html:	index.html	*content-type: text/html*
/index2.html:	index2.html	*content-type: text/html*
/css/css-file1.css:	css/css-file1.css	*content-type: text/css*
/css/css-file2.css:	css/css-file2.css	*content-type: text/css*

Notes:

- You may hardcode the folder-path that the static content is stored in.
- In case an invalid url/file is given, a 404 error may be generated.
- Use try-with-resources to access files

An overview of try with resource may be found here:
https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html

A sample http server implementation may be found here:
https://www.logicbig.com/tutorials/core-java-tutorial/http-server/http-server-basic.html

Task 4. Using Curl
Target topics: using curl, sending headers and data, viewing header and data.

Using curl, perform the following operations and report the result (in Task 5).

- perform a GET method on the servlet and specify the format to html; provide user agent request header; report full response including the response headers
- perform a POST method on the servlet and specify the format to xml; provide user agent request header; report full response including the response headers
- perform a GET method on index.html; provide user agent request header; report full response including the response headers
- perform a POST method on the http-server and specify the format to text; provide user agent request header; report full response including the response headers

Hint: use `curl -v` for full response view

Task 5. The readme.md file
Target topics: understating wen and http, building and deployment.

In task, you are required to create an MD file that provides the following information. The information in this task is not marked, however, they are used for individual assessment. Each team member must list their contribution to the project. This will be used for individual assessment. Additionally,

- Examine if a war file is generated for your project. How was it generated? Where is the location of the generated war file?
- How do you run the HttpServer class in Task 3? What are the deployment requirements (i.e. in case the application is run on a different machine)?
- What happens when you submit the html forms in Task 3?
- Provide the four curl outputs in Task 4.

A detailed information on how to create md files may be found here:
https://www.markdownguide.org/getting-started/

Deliverables

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). You and your teammate must be in the same section. Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted for both members. You must make sure that you upload the assignment to the correct directory of **Assignment 1** using EAS. Assignments uploaded to the wrong directory will be discarded and no resubmission will be allowed.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention:
The zip file should be called *a#_studentID*, where # is the number of the assignment *studentID* is your student ID(s) number. For example, for the first assignment, student 12345678 would submit a zip file named *a1_12345678.zip*. If you work on a team and your IDs are 12345678 and 34567890, you would submit a zip file named *a1_12345678_34567890.zip*.

Submit your assignment electronically via EAS based on the instruction given by your instructor as indicated above. **Please see course outline for submission rules and format, as well as for the required demo of the assignment.** A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. **You must keep a record of your submission confirmation.** This is your proof of submission, which you may need should a submission problem arises.

Grading Scheme

T#	MX	MK	
1	/5		*servlet, different outputs, content type, response, and encoding*
2	/2		*xhtml , css*
3	/2		*http server, file/stream IO, content type, and response*
4	/1		*curl, headers and data, POST and GET*
5	/0		
Total:	/10		

(T# - task number, MX - max (out of), MK - your mark)

References

1. https://www.markdownguide.org/getting-started/
2. https://www.logicbig.com/tutorials/core-java-tutorial/http-server/http-server-basic.html
3. https://way2java.com/servlets/java-made-clear-response-setcontenttype-method-example/
4. https://www.dummies.com/programming/java/how-to-use-the-javac-command/
5. https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html