CSSE 477 : Milestone 1 + 2 : Web Server

You are provided with a working web server that can handle GET requests. The web server, at the moment, does not handle POST, PUT, DELETE, or HEAD requests. Note that the main purpose of this homework is to help you understand how a web server works so that you can extend this homework to build a Web Application Server such as Tomcat and Glassfish servers that provide web containers for Java applications in Milestone 3. Please refer to Section 9 of HTTP RFC [http://tools.ietf.org/html/rfc2616] and response codes [http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html] to understand the details of POST, PUT, and DELETE.

You will work in a **group of three** (ideal) to **four** (max) for the course project. There are altogether four weekly milestones in the project.

Important Note

No specification can be bullet-proof and cover all possible details of the system being specified. That is why we advocate communicating with different project stakeholders to clarify requirements in our field. In that spirit, if you have a confusion about any parts of this spec, please feel free to contact your instructor.

Functional Requirements (Milestone 2)

You will implement all the features specified under this section in MS2. Furthermore, you will also revisit the Quality Attributes Requirements section (below) after implementing these features and satisfy the requirements specified there for MS2.

F1 – Redesign and Refactor

Redesign and refactor the existing project by implementing appropriate design patterns to **avoid modifying the same source code files multiple times** for implementing the next four requests and corresponding responses. Also, make other appropriate design improvements where applicable.

F2 – Basic Support for the HEAD Request

Your HEAD request will provide necessary info about the requested resource without returning the resource.

F3 – Basic Support for the POST Request

Your POST request will **create** a new file if one does not exist or **append** the contents in the body of the request to an existing file. Assume plain text files for this milestone.

F4 – Basic Support for the PUT Request

The POST request must **create** a new file or **overwrite** an existing file in the root directory hosted by the web server with the content in the body (payload) of the request.

F5 – Basic Support for the DELETE Request

Your DELETE request will delete an existing file.

Note that for these requests, your server must return an appropriate status response back to the client. Also, it should be able to handle nested files and folders starting at the directory it is serving.

Quality Attribute Requirements (Milestone 1)

These are the requirements for both MS1 and MS2.

Q1 – Build System

Currently, the project is an Eclipse project. Configure the project to use Gradle. Your build script should support importing project in Eclipse and IntelliJ IDE.

Q2 – Automated Unit Testing

You project must have automated JUnit-based tests that test the core functionality of your code. The build script must also show the code coverage based on the JUnit tests you have written.

[**Note: For MS1, having a few tests as a proof-of-concept is just fine. For MS2, please provide enough test cases to test important pieces of your code.**]

Q3 – Automated Black-box (End-To-End) Testing

Your project must have automated JUnit-based tests that sets up necessary resources for the web server to serve, spins the web server as a back-box, and use a HTTP client (such as Google Http Java Client) to perform GET/PUT/POST/DELETE/HEAD tests programmatically, and finally stops the server and clears the allocated resources within the JUnit tests.

[Note: For MS1, you may just test the GET request/response. For MS2, you must test all supported HTTP verbs.]

Q4 – Continuous Integration and Delivery

Your project must be hosted on Ada (GITLab) and equipped with .gitignore, .gitlab-ci.yml, README.md, and other necessary files. Your project must automatically build in one of the GITLab runners on a GIT push. You must also perform the following two deployments:

1. **Staging Deployment**
   You should use one of the VM assigned to a team member as the staging server. After a successfully build on **git push**, you should deploy the web server jar to the staging server and run end-to-end tests using an HTTP client such as **curl**. You must create necessary Linux script to run your jar as a Linux daemon service in the staging server. Here is a 5-minute tutorial to help you with that (https://blog.terminal.com/using-daemon-to-daemonize-your-programs/).

2. **Production Deployment**
   You should use another VM assigned to a team member as the production server. After a successfully build on **git tag**, you should deploy the web server jar to the production server and run end-to-end tests using an HTTP client such as **curl**. Again, the Linux daemon service is a requirement for the production server as well.

Q5 – Late Bindings

Your web server must load settings from a configuration file (you may use a java properties file if you like). The two settings that must be configurable now are: i) The web server root directory, and ii) Port number. Feel free to add more config as you see fit.

Q6 – Logging

The outputs and errors of the server are currently being printed on the console. Change this to use standard logging framework (such as SL4J/Log4j). There must be two logs: error and access. The error log file should have logs of the error that the server generates and the access log file should have logs of the request/reply that the server handles.

Q7 - Report (README.md)

1. The code coverage and build status badge must be shown in the README.md file on the Ada server.
2. You have already developed architecture (module diagram) for the web server as a part of Paper Review 3. You must have a new found understanding of how the web server works through these exercises. Use this new found understanding to refine your architecture diagram and present it in README.
3. You have also developed a detailed design of the web server as a part of Paper Review 3. Refine the detailed design as well and present it in README.
4. Present other standard details one should know about your project in the README.

Deliverables

Just enter the URL of your project here:
https://docs.google.com/spreadsheets/d/1UzH5w1AvLFJOmFLtLZ7imZuLFtoazeAiIv6emcie6Uk/edit?usp=sharing