Homework 4

ECE567: Software Engineering 1

Tanya Sharma
tds104

23 October 2023

Design

This homework assignment creates an HTTP server that listens on port 3000. It is designed to handle incoming POST requests to the "/echo" endpoint. The primary purpose of this server is to receive data in various content types (text, JSON, XML), and then echo the received data back to the client in the same format. It checks the "Content-Type" header of incoming requests to determine the data format, processes the data accordingly, and responds with the echoed data. If the content type is not supported or there are any issues with the request, appropriate HTTP status codes and error messages are returned to the client. This code can be used to set up a simple HTTP server that echoes data in different formats back to clients making POST requests to the "/echo" endpoint.

Commands

In order to execute the both the program as well as test cases the following instructions can be followed:

1. **npm install**
Install all necessary dependencies

2. **node main.js**
This command will execute the main function where all our logic takes place

3. **jasmine**
This command will execute the test suite

Postman Requests

Sending Plain Text Data:

- Open Postman.
- Create a new POST request.
- Set the request URL to http://localhost:3000/echo.
- In the request headers, add Content-Type: text/plain.
- In the request body, select "raw" and enter some plain text, for example: "This is plain text data."
- Send the request.
- You will receive a response with a status code of 200 OK, and the response body will contain the same plain text data you sent.

Sending JSON Data:

- Open Postman.
- Create a new POST request.
- Set the request URL to http://localhost:3000/echo.
- In the request headers, add Content-Type: application/json.
- In the request body, select "raw" and enter some JSON data, for example:{ "key": "value" }
- Send the request.
- You will receive a response with a status code of 200 OK, and the response body will contain the same JSON data you sent.

Sending XML Data:

- Open Postman.
- Create a new POST request.
- Set the request URL to http://localhost:3000/echo.
- In the request headers, add Content-Type: application/xml.
- In the request body, select "raw" and enter some XML data, for example:root> <element>XML data</element> </root>
- Send the request.
- You will receive a response with a status code of 200 OK, and the response body will contain the same XML data you sent.

Test Cases

Here is a brief description of the test cases in the provided code:

- **404 for a GET with a valid content type and a valid message:**

Verifies that a GET request to the "/echo" endpoint with a "text/plain" content type results in a 404 Not Found response.

- **404 for POST to an unknown route:**

Ensures that a POST request to an unknown route ("/unknown-route") with "text/plain" content type also results in a 404 Not Found response.

- **200 for a POST with valid content type and a valid message (text/plain):**

Tests that a POST request to the "/echo" endpoint with a "text/plain" content type and the message "Hello" results in a 200 OK response.

- **200 for a POST with valid content type and a valid message (application/json):**

Validates that a POST request to the "/echo" endpoint with an "application/json" content type and a JSON message results in a 200 OK response.

- **200 for a POST with valid content type and a valid message (application/xml):**
Tests a POST request with "application/xml" content type and a message in the form of XML data. Expects a 200 OK response.

- **415 for a POST with a valid message but an unsupported content type:**
Ensures that a POST request with a supported message but an unsupported content type results in a 415 Unsupported Media Type response.

- **415 for a POST with a missing content type:**
Tests a POST request with a missing "Content-Type" header, expecting a 415 Unsupported Media Type response.

- **400 for a POST with a valid content type but an invalid message:**
Validates that a POST request with a valid content type but an invalid message results in a 400 Bad Request response.

- **400 for a POST with a valid content type but an empty message:**
Verifies that a POST request with a valid content type but an empty message results in a 400 Bad Request response.

- **Handles a POST with a very large message without error:**
Ensures that the server can handle a POST request with a very large message (10MB) without errors and returns a 200 OK response.

- **200 for a valid text/plain message with matching content:**
Tests that a POST request with a "text/plain" message returns a 200 OK response, and the echoed content matches the original message.

- **200 for a valid application/json message with matching content:**
Validates that a POST request with an "application/json" message returns a 200 OK response, and the echoed JSON content matches the original JSON.

- **200 for a valid application/xml message with matching content:**
Ensures that a POST request with an "application/xml" message returns a 200 OK response, and the echoed XML content matches the original XML.

- **Handles multiple concurrent requests without crashing:**
Demonstrates that the server can handle multiple concurrent POST requests (up to 10 in this case) without crashing and validates the echoed content for each request.