## Lab Assignment – 9

### **Name:** Srujan Patwardhan

### **STD:** TY-D

### **Roll no.:** 02

### **PRN:** 12210847

### **Subject:** Computer Networks

#### Q1. Develop a client-server using C++ or JAVA to demonstrate the behaviour of HTTP1.0, HTTP1.1, HTTP1.2 and HTTP2.0 protocols along with all success and error messages. Use Firefox as client browser.

##### CODE:

###### DynamicHTTPClient.java:

import *java.io.\**;

import *java.net.\**;

import *java.util.\**;

*public* *class* DynamicHTTPClient {

*public* *static* void main(String[] args) *throws* IOException {

*// Example input for dynamic testing*

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter server host: ");

        String host = scanner.nextLine();

        System.out.print("Enter server port: ");

        int port = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter HTTP method (e.g., GET): ");

        String method = scanner.nextLine();

        System.out.print("Enter path (e.g., /index.html): ");

        String path = scanner.nextLine();

        System.out.print("Enter HTTP version (e.g., HTTP/1.1): ");

        String httpVersion = scanner.nextLine();

        System.out.print("Do you want to add custom headers? (yes/no): ");

        boolean addHeaders = scanner.nextLine().equalsIgnoreCase("yes");

        Map<String, String> headers = new HashMap<>();

        if (addHeaders) {

            while (true) {

                System.out.print("Enter header key (or 'done' to finish): ");

                String key = scanner.nextLine();

                if (key.equalsIgnoreCase("done")) break;

                System.out.print("Enter header value: ");

                String value = scanner.nextLine();

                headers.put(key, value);

            }

        }

        sendRequest(host, port, method, path, httpVersion, headers);

        scanner.close();

    }

*private* *static* void sendRequest(String host, int port, String method, String path, String httpVersion, Map<String, String> headers) *throws* IOException {

        Socket socket = new Socket(host, port);

        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        String request = method + " " + path + " " + httpVersion;

        System.out.println("Sending request: " + request);

        out.println(request);

        out.println("Host: " + host);

        for (Map.Entry<String, String> header : headers.entrySet()) {

            out.println(header.getKey() + ": " + header.getValue());

        }

        out.println("Connection: close");

        out.println("");

        String responseLine;

        while ((responseLine = in.readLine()) != null) {

            System.out.println(responseLine);

        }

        socket.close();

    }

}

###### DynamicHTTPServer.java:

import *java.io.\**;

import *java.net.\**;

import *java.util.\**;

*public* *class* DynamicHTTPServer {

*public* *static* void main(String[] args) *throws* IOException {

        int port = 8080;

        try (ServerSocket serverSocket = new ServerSocket(port)) {

            System.out.println("Server started on port " + port + "...");

            while (true) {

                Socket clientSocket = serverSocket.accept();

                new ClientHandler(clientSocket).start();

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

*class* ClientHandler *extends* Thread {

*private* Socket clientSocket;

*public* ClientHandler(Socket socket) {

        this.clientSocket = socket;

    }

*public* void run() {

        try (BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

             PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

            String requestLine = in.readLine();

            if (requestLine == null) return;

            System.out.println("Request: " + requestLine);

            Map<String, String> headers = new HashMap<>();

            String headerLine;

            while (!(headerLine = in.readLine()).equals("")) {

                String[] headerParts = headerLine.split(": ", 2);

                if (headerParts.length == 2) {

                    headers.put(headerParts[0], headerParts[1]);

                }

            }

            String[] requestParts = requestLine.split(" ");

            if (requestParts.length != 3) {

                sendErrorResponse(out, "400 Bad Request");

                return;

            }

            String method = requestParts[0];

            String path = requestParts[1];

            String httpVersion = requestParts[2];

            handleRequest(method, path, httpVersion, out);

            clientSocket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

*private* void handleRequest(String method, String path, String httpVersion, PrintWriter out) {

        String responseBody;

        String statusLine;

        if (!httpVersion.matches("HTTP/\\d\\.\\d")) {

            sendErrorResponse(out, "505 HTTP Version Not Supported");

            return;

        }

        switch (httpVersion) {

            case "HTTP/1.0":

            case "HTTP/1.1":

            case "HTTP/2.0":

                if (method.equals("GET")) {

                    responseBody = "<html><body><h1>Response from " + path + " using " + httpVersion + "</h1></body></html>";

                    statusLine = httpVersion + " 200 OK";

                } else {

                    sendErrorResponse(out, "405 Method Not Allowed");

                    return;

                }

                break;

            default:

                sendErrorResponse(out, "400 Bad Request");

                return;

        }

        out.println(statusLine);

        out.println("Content-Type: text/html");

        out.println("Content-Length: " + responseBody.length());

        out.println("");

        out.println(responseBody);

    }

*private* void sendErrorResponse(PrintWriter out, String status) {

        String responseBody = "<html><body><h1>" + status + "</h1></body></html>";

        out.println(status);

        out.println("Content-Type: text/html");

        out.println("Content-Length: " + responseBody.length());

        out.println("");

        out.println(responseBody);

    }

}

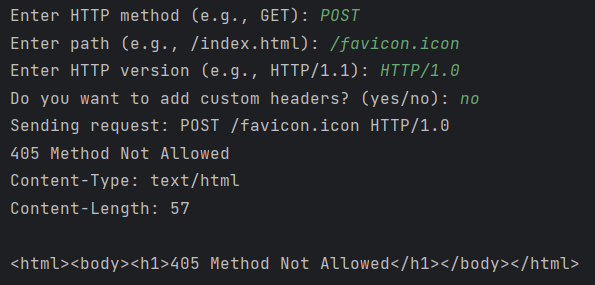
###### OUTPUT:

###### Client

###### HTTP/1.0:

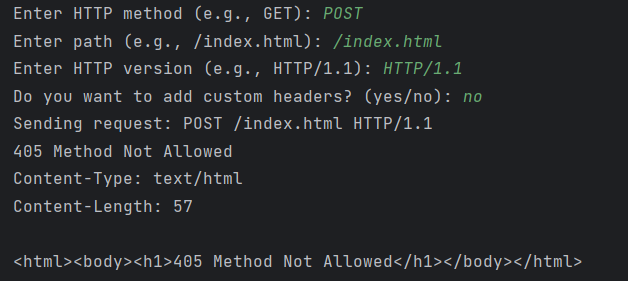


###### Fig. 1. GET request using HTTP/1.0 with OK response and index.html returned.



###### Fig. 2. POST request with 405 response code as server doesn’t support POST in code

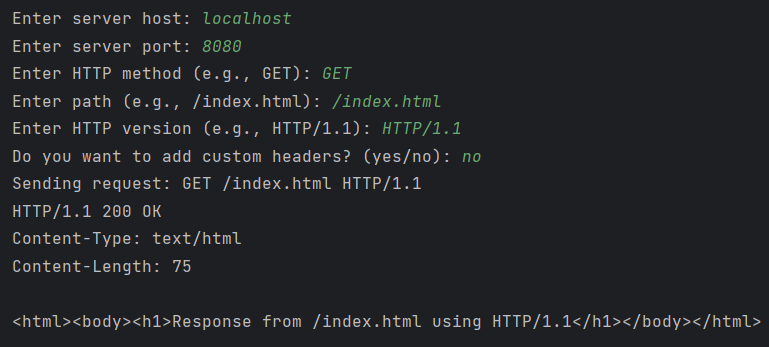
###### HTTP/1.1:



###### Fig. 3. POST request with 405 response code as server doesn’t support POST in code

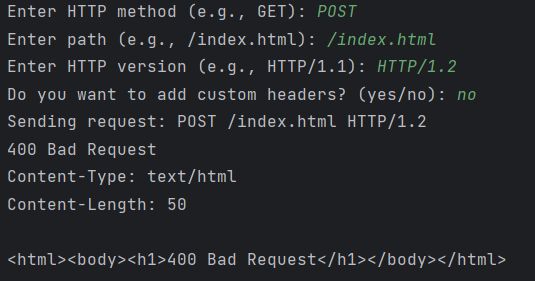


###### Fig. 4. GET request using HTTP/1.1 with custom headers and OK response and index.html returned.



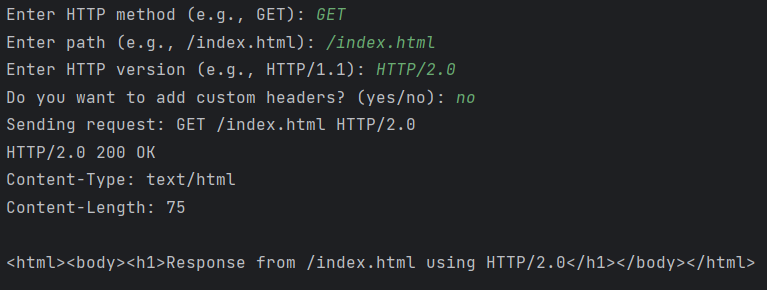
###### Fig. 5. GET request using HTTP/1.1 with OK response and index.html returned.

###### HTTP/1.2:



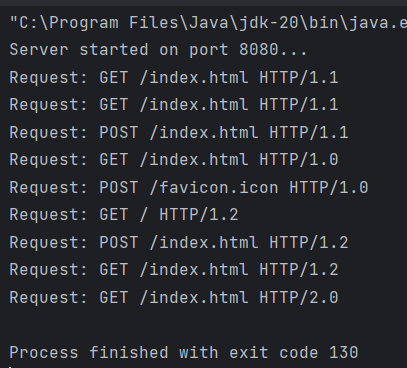
###### Fig. 6. POST request using HTTP/1.2 with Bad request response as Java doesn’t support HTTP/1.2

1. **HTTP/2.0:**



###### Fig. 7. GET request using HTTP/2.0 with OK response and index.html returned.

1. **Server:**



###### Fig. 8. All the requests using various HTTP at server side.