****

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Fall, Year:2024), B.Sc. in CSE (Day)**

**Lab Report NO: 01**

**Course Title: Computer Networking Lab**

**Course Code: CSE-304 Section:221-D21**

**Lab Experiment Name: Implementation of HTTP POST and GET methods.**

**Student Details**

| **Name** | | **ID** |
| --- | --- | --- |
| **1.** | Masum Hossain | 221902164 |

**Lab Date : 15/09/2024**

**Submission Date : 15/11/2024**

**Course Teacher’s Name : Md. Saiful Islam Bhuiyan**

| **Lab Report Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |
| --- |

**1. TITLE OF THE LAB REPORT EXPERIMENT**

Implementation of HTTP POST and GET methods.

**2. OBJECTIVES**

This lab experiment involves creating a Java program to perform HTTP GET and POST requests using the HttpURLConnection class.As we are concerned that, the GET request fetches data from a specified endpoint, while the POST request sends data to a server. The aim is to understand and implement fundamental REST API(web service interface) request methods using Java as well as implementing the following objectives:

1. To understand the fundamentals of HTTP communication.
2. To implement a GET request to retrieve data from an external API.
3. To implement a POST request to send data to an external API.
4. To process and display the response from the server.

**3. ANALYSIS**

In this experiment both GET and POST methods have been implemented for fetching data from the server and sending data to the server respectively. Here we connect to an online REST API (https://jsonplaceholder.typicode.com) to perform GET and POST requests using Java.

Algorithm of the Program:

1.Define two URLs for GET and POST requests.

2.Create a method, sendGET for the GET request:

* Open an HTTP connection to the specified URL.
* Set the request method to GET.
* Check the response code; if successful (200) read and display the response content.

3.Create a method, sendPOST for the POST request:

* Open an HTTP connection to the specified URL.
* Set the request method to POST and define the content type as JSON.
* Enable output on the connection and write the JSON string to the request body.
* Check the response code; if successful (201), read and display the response content.

4.Execute the sendGET and sendPOST methods in the main function and handle any exceptions.

**4. IMPLEMENTATION**

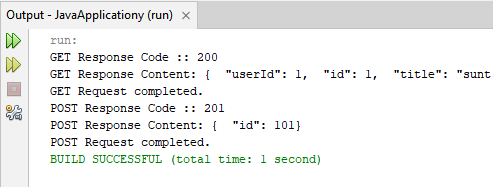
The program is implemented in Java using the HttpURLConnection class for network communication. We define two methods, sendGET and sendPOST. In sendGET, the program connects to a specified URL with a GET request, retrieves the data, and reads the response. In sendPOST, the program establishes a POST request connection to the URL and sends a JSON payload in the request body. The server's response code and response body are read and displayed for each request, validating the success of both operations.

**Implemented Code**

| import java.io.BufferedReader;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.net.HttpURLConnection;  import java.net.URL;  public class GetPostMainClass {  private static final String GET\_URL = "https://jsonplaceholder.typicode.com/posts/1";  private static final String POST\_URL = "https://jsonplaceholder.typicode.com/posts";  public static void main(String[] args) {  try {  sendGET();  System.out.println("GET Request completed.");    sendPOST();  System.out.println("POST Request completed.");  } catch (Exception e) {  e.printStackTrace();  }  }  private static void sendGET() throws Exception {  URL url = new URL(GET\_URL);  HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();  httpURLConnection.setRequestMethod("GET");  int responseCode = httpURLConnection.getResponseCode();  System.out.println("GET Response Code :: " + responseCode);  if (responseCode == HttpURLConnection.HTTP\_OK) { // successGET  BufferedReader in = new BufferedReader(new InputStreamReader(httpURLConnection.getInputStream()));  String inputLine;  StringBuilder content = new StringBuilder();  while ((inputLine = in.readLine()) != null) {  content.append(inputLine);  }  System.out.println("GET Response Content: " + content.toString());  in.close();  } else {  System.out.println("GET request failed.");  }  }  private static void sendPOST() throws Exception {  URL url = new URL(POST\_URL);  HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();  httpURLConnection.setRequestMethod("POST");  httpURLConnection.setRequestProperty("Content-Type", "application/json; utf-8");  httpURLConnection.setRequestProperty("Accept", "application/json");  httpURLConnection.setDoOutput(true);  String jsonInputString = "{ \"title\": \"foo\", \"body\": \"bar\", \"userId\": 1 }";  try (OutputStream os = httpURLConnection.getOutputStream()) {  byte[] input = jsonInputString.getBytes("utf-8");  os.write(input, 0, input.length);  }  int responseCode = httpURLConnection.getResponseCode();  System.out.println("POST Response Code :: " + responseCode);  if (responseCode == HttpURLConnection.HTTP\_CREATED) {  BufferedReader in = new BufferedReader(new InputStreamReader(httpURLConnection.getInputStream()));  String inputLine;  StringBuilder content = new StringBuilder();  while ((inputLine = in.readLine()) != null) {  content.append(inputLine);  }  System.out.println("POST Response Content: " + content.toString());  in.close();  } else {  System.out.println("POST request failed.");  }}} |
| --- |

**5. OUTPUT**

For these three programs,these have been tested several times and we have got the desired output each time according to data sent by the server.



| **Fig 01: Showing the GET and POST response.** |
| --- |

**6. ANALYSIS AND DISCUSSION**

The experiment was successful in demonstrating how to implement GET and POST requests using Java. The program made use of HttpURLConnection to establish the connection and retrieve responses, handling different response codes for validation. Through this exercise, we explored the importance of HTTP status codes in evaluating the success or failure of network requests and gained insights into handling JSON payloads in POST requests.

**7. SUMMARY:**

A REST API is a web service interface that allows communication between systems using standard HTTP methods to perform actions on resources identified by URLs.