Green University of Bangladesh
Department of Computer Science and Engineering(CSE)
Faculty of Sciences and Engineering
Semester: (Spring Year:2024), B.Sc. in CSE (Day)

LAB REPORT # 02
Course Title: Computer Networking Lab
Course Code: CSE 312 Section: D16

Lab Experiment Name: Implementation of HTTP GET and POST Methods.

Student Details

Name		ID
	Md. Moshiur Rahman	221902324

Lab Date : 21/9/2024
Submission Date : 24/9/2024
Course Teacher's Name : Maisha Muntaha
[For Teachers use only: Don't Write Anything inside this box]

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

1. TITLE OF THE LAB EXPERIMENT

HTTP (Hypertext Transfer Protocol) is a fundamental protocol used for communication on the World Wide Web. HTTP encompasses several methods, including GET and POST, which are employed to request and transmit data between clients (typically web browsers) and web servers. This lab focuses on implementing these methods in a Java application to illustrate their functionality.

2. OBJECTIVES/AIM

HTTP (Hypertext Transfer Protocol) defines various methods for communication between clients and web servers. Two of the most commonly used methods are HTTP GET and HTTP POST. GET is used to request data from a web server.Retrieving web pages, fetching resources (images, scripts), querying data from web services.POST is used to submit data to a web server for processing or storage.Submitting web forms, uploading files, creating or updating resources.

3. PROCEDURE / ANALYSIS / DESIGN

In this experiment we perform two operations for HTTP GET and POST method.Here given the algorithm to understand how this method works.

1. **Algorithm(GET Method)**
   - Step 1: Create a url (https://jsonplaceholder.typicode.com/posts/)
   - Step 2: HTTP connection (openConnection())
   - Step 3: Configure connection setRequestMethod("POST")
   - Step 4: Prepare Data "create string conating"
   - Step 5: Send Data "outputStream and write string content"
   - Step 6: verifying "responseCode = = HTTP_CREATED
   - Step 7: Read response: "bufferReader"
   - Step 8: execute posted content
   - Step 9: END
2. **Algorithm(POST Method)**
   - Step 1: Create URL "HTTPConnection, openConnection();
   - Step 2: Configure connection "setRequestMethod("GET");
   - Step 3: Check response "if responseCode = = HTTP_OK
     - **Read content;** print the string
     - **Else;** print error message
   - Step 4: EXIT

4. IMPLEMENTATION

- **HTTP(POST) Method**

```
package cse312;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.*;
public class HTTP_GET {
   public static void main(String[] args) throws MalformedURLException,
         IOException {
      URL url = new URL("https://jsonplaceholder.typicode.com/posts/");
      HttpURLConnection conn = (HttpURLConnection) url.openConnection();
      conn.setRequestMethod("POST");
      conn.setDoOutput(true);
      OutputStream out = conn.getOutputStream();
      String str = "Hi!!! Connection Successfully ";
      out.write(str.getBytes());
      int responseCode = conn.getResponseCode();
      if(responseCode == HttpURLConnection.HTTP_CREATED){
         System.out.println( responseCode);
         System.out.println(conn.getResponseMessage());
      }else{
         System.out.println("Go to Home");
      }
      InputStreamReader in = new InputStreamReader(conn.getInputStream());
      BufferedReader buffer = new BufferedReader(in);
      String eachline = null;
      StringBuffer strBuffer = new StringBuffer();
         while((eachline = buffer.readLine())!=null){
            strBuffer.append(eachline);
         }
      System.out.println(strBuffer);
   }
}
```

- **HTTP(POST) Method**

```
package cse312;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.*;
public class HTTP_POST {
  public static void main(String[] args) throws MalformedURLException,
         IOException {
      URL url = new URL("https://jsonplaceholder.typicode.com/posts/");
      HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

```
        conn.setRequestMethod("POST");
        conn.setDoOutput(true);
        OutputStream out = conn.getOutputStream();
        String str = "Hi!!! Connection Successfully ";
        out.write(str.getBytes());
        int responseCode = conn.getResponseCode();
        if(responseCode == HttpURLConnection.HTTP_CREATED){
           System.out.println( responseCode);
           System.out.println(conn.getResponseMessage());
        }else{
           System.out.println("Go to Home");
        }
        InputStreamReader in = new InputStreamReader(conn.getInputStream());
        BufferedReader buffer = new BufferedReader(in);
        String eachline = null;
        StringBuffer strBuffer = new StringBuffer();
           while((eachline = buffer.readLine())!=null){
              strBuffer.append(eachline);
           }
        System.out.println(strBuffer);
    }
}
```

5. TEST RESULT / OUTPUT

Fig-1.1(POST Method)

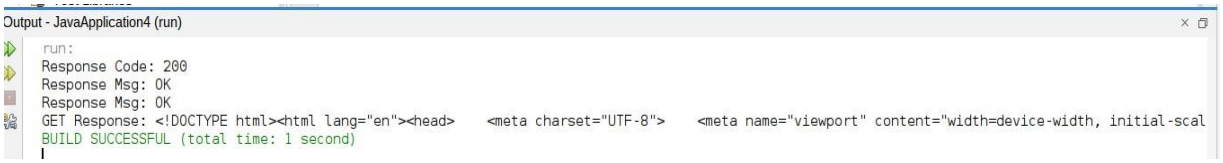

Fig-1.2(GET Method)

6. ANALYSIS AND DISCUSSION

1. **Testing the GET Method**
   - Create an instance of `HTTPMethodsExample`.
   - Use the `sendGetRequest` method to send a GET request to a web server.
   - Capture and display the response.
2. **Testing the POST Method**
   - Create an instance of `HTTPMethodsExample`.

- Use the `sendPostRequest` method to send a POST request to a web server.
- Capture and display the response.

☐ **Analysis and discussion of the result / output**

➢ Successfully completing this report was a significant achievement. The program executed as intended, making an HTTP GET request to webcode.me The obtained response code and message were displayed

☐ **What can be better?**

➢ The HTTP GET program was tested in multiple programming environments, providing valuable insights into different response values under varied conditions.But HTTP POST is a more secure purpose used.

☐ **What did we learn from it?**

➢ The assignment provided valuable insights into creating custom HTTP GET and POST methods and establishing internet connections.We also know about the basic difference and mechanism of these two methods.

7. SUMMARY:

The lab report on the "Implementation of HTTP GET and POST Methods in Java" describes a practical exploration of using these HTTP methods within a Java application. HTTP GET and POST methods are fundamental for web communication, enabling clients to request data from servers (GET) or send data to servers (POST).
Overall, this lab served as a valuable hands-on experience in implementing and comprehending HTTP GET and POST methods within a Java application, highlighting their significance in web development and data exchange.

1. ***Why do you think that after implementing GET HTTP method, an entire HTML structure of the requested web page comes as output? Explain to your instructor.***

2. In Algorithm 1, there is an extra step in line 4 for POST method implementation. Why do you think there is no need for such a step (as can be seen in Algorithm 2) in implementing the GET HTTP method for any web page?

Ans:

1) When using the GET HTTP method, the entire HTML structure of the requested web page is returned because GET is designed to retrieve and display resources from the server. It takes the data without changing it, which is why you see the entire content as HTML in the response. This content is intended to be rendered in a browser.

2) In the POST method, there is an additional step (as in line 4 of the algorithm) to send the data to the server. POST is used to send data (eg form items) for processing. However, GET does not involve sending data, so this step is not necessary as shown in Algorithm 2. GET is primarily for retrieving data, while POST involves both sending and receiving.