Department of
Computer Science and Engineering

Title: Implementation of HTTP POST and GET methods

Computer Newtorking Lab

CSE 312

Green University of Bangladesh

1 Objective(s)

- To gather knowledge of different types of HTTP methods and the main concept.
- To implement the widely-used HTTP POST and GET methods and their uses.

2 Problem analysis

Every HTTP server website has several web pages inside it. Each web page can be separately accessed by the clients getting connected to the web server. These accesses can be of many various forms and ways. Separate HTTP methods exist for each separate applications from the clients' side for the HTTP server. Two such type of operations can be requested by the client to the server using POST and GET methods.

2.1 POST method

Every web page has a specific address that a client can get attached to. This attachment with that specific web page is done with the help of a specific **connection**. This connection must be created with any type of operations from the client towards the web server's any web page.

Whenever a client wants to provide some information to help or improve a certain web page of a certain web server, **POST** method can be used. The HTTP POST method sends data to the server. It is also used when uploading a file or when submitting a completed web form. The information inside the form, or the entire file can be sent to the web page only if the request from the client contains this HTTP "POST" method in its request message's method field. So this can be established that if a client wants to write any information on a certain web page this POST HTTP method can be used by the client.

3 Procedure

here we will have a detailed view on the regular steps that are followed using a client while posting a certain content on a pre-selected web page of a particular web server.

3.1 POST method

As an initial step, the connection is to be made with the certain web page, where client needs to POST the contents. Then the materials to be posted are collected. This can be done in many ways. As mentioned earlier, these materials can be received from a different form elsewhere, or can be received as a search parameters by any other users, or can even be some simple character string taken as input form the keyboard. (In this lab experiment, we take in input form keyboard and post this in the webpage.) After that, client send information to the web page, that client wants the POST method for that connection application. Different several properties can be set by the client manually with this connection, if felt needed. Finally, a response is sent from the server. With this response message, the server sends a certain response code and a response message. For POST method, this code is 201. This information can be nicely utilized by the client for cross-checking. If the returned code matches 201, then client can decide that the sent contents are correctly posted in the web page. Later on, the client can read the entire web page again with the help of GET HTTP method, or read the partial posted content only, by a simple read operation.

It is to be mentioned that the implementation of the HTTP POST GET methods in this experiment is to be done using JAVA programming language. In JAVA, **HttpURLConnection** class from java.net package can be used to send Java HTTP Request programmatically for both POST and GET, which ultimately creates the main connection of the client program with the particular web page. Below are the steps we need to follow for sending Java HTTP POST request using HttpURLConnection class, as described above in this section.

1. Create **URL** object from the GET/POST URL String.
2. Call **openConnection()** method on URL object that returns instance of **HttpURLConnection**
3. Set the request method in **HttpURLConnection** instance, default value is GET. So we change this to POST.

4. For POST, we need to get the **OutputStream** from **HttpURLConnection** instance and write POST parameters into it. This is required to collect the contents to be posted in the web page.

5. *(optional)* Call **setRequestProperty()** method on **HttpURLConnection** instance to set request header values, such as "**User-Agent**" and "**Accept-Language**" etc.

6. We can call **getResponseCode()** to get the response HTTP code. This way we know if the request was processed successfully or there was any HTTP error message thrown.

7. After this we can simply use **Reader** and **InputStream** to read our posted contents from the web page or use GET HTTP method to create a new connection entirely to read the entire web page newly.

4 Algorithm

Algorithm 1: HTTP POST method implementation

Input: Given in the program by default according to the programmer's requirement
Output: Print of the posted content in the web page

Step 1: Create an instance of the **URL** class with the url of the web page in the constructor. Our webpage has the url: "*https://jsonplaceholder.typicode.com/posts/*"
Step 2: Create an instance of the **HTTPURLConnection** class, made as a return from the function **openConnection()** of the url instance.
Step 3: Enable the HTTP method, that the client wants to do, using **setRequestMethod()** function. Here the parameter will be "**POST**".
Step 4: Enable the abolity of the client program to post anything on the given web page using the function **setDoOutput** with changing default parameter to **"True"**.
Step 5: Create a string containing the content that needs to be posted. Save it using s String instance.
Step 6: Create an instance of the **OutputStream** class to have a byte stream attachment with the connection instance with the web page.
Step 7: Finally write the saved string on to this instance of **OutputStream** class, using function **write()**. Thus the string to be posted is sent to the server to post in the given web page.
`/* Verify if the content is correctly posted using response from the server. */`
Step 8: **if** *responseCode == HTTP_CREATED* **then**
    print out the response code and response messages from the web server
else
    print out an error message
end
`/* Here HTTP_CREATED is a defined value that equals to 201 */`
Step 9: Now read the posted content from the web page, using instances of **InputStreamReader** and **BufferedReader** classes.
Step 10: An empty string is initialized to store the read contents from the connection instance created at Step 2.
Step 11: **while** *reading up to the end of the posted content* **do**
    append each sentence of the posted content with the initailized string.
end
Step 12: Print the created string, being read from the posted content in the web page.
Step 13: Exit

In this algorithm, using Step 8, it is verified by the client whether the posted content has reached the server correctly. This is done using checking the response code from server. After that, using Steps 9-12, the posted contents are read by the client itself to visualize the content correctly. Steps 1-7 are the actual steps to post any content on the given web page. It needs to be mentioned, that Step 4 is necessary, since posting on any web page is strictly prohibited by the web server. Since we are trustworthy clients and as we could be given this permission to post anything, JAVA connection class function needs to remove the default value "**False**" for the function **setDoOutput**. Otherwise, nothing could be posted in any web page ever.

If, not only the posted content, entire web page is required to be read, then a separate connection instance is to be created. After that, the GET method needs to be implemented similarly to Algorithm 1.

5 Implementation in JAVA

```
1  /* HTTP POST method implementation */
2  package post;
3  import java.io.*;
4  import java.net.HttpURLConnection;
5  import java.net.MalformedURLException;
6  import java.net.URL;
7
8  public class post{
9      public static void main(String[] args) throws MalformedURLException,
           IOException {
10
11         // opening the connection to a URL webpage where we can post
12         URL myUrl = new URL("https://jsonplaceholder.typicode.com/posts/");
13         HttpURLConnection conn = (HttpURLConnection) myUrl.openConnection();
14
15         // setting the method for this connection
16         conn.setRequestMethod("POST");
17
18         // now posting anything on the given URL webpage
19         conn.setDoOutput(true);
20         // this is to be done so that we can surely post anything using this
               connection
21         OutputStream out = conn.getOutputStream();
22         // this is to take anything into out for posting using the connection
23         String postedString = "Hi!!! We have posted something!!! Yay!!!";
24         // this is the string we wanted to post in the URL
25         out.write(postedString.getBytes());
26         // we write/post all the bytes in the string postedString
27
28         // now we test if our posting the postedString string is correct or not
29         // first we get the responsecode from the connection, then we test by
               printing out the code and the responseMessage from URL server
30         int responseCode = conn.getResponseCode();
31         System.out.print("Value of http created is: "+conn.HTTP_CREATED+"\n");
32         if (responseCode == conn.HTTP_CREATED)
33         {
34             System.out.print("This is the response Code: "+responseCode+"\n");
35             System.out.print("This is the response Message from server: "+conn.
                   getResponseMessage()+"\n");
36         }
37         else System.out.print("GO HOME EVERYBODY :( ");
38
39         // Now since we have completed posting, we need to see our posted
               content
40         // this is to be done using our GET method to the connection
41         // we DO NOT NEED to use GET method, we just read it using inputstream
               and strings
42
43         // we need string to get printed. so we need a string-type object, where
                the new URL content will be saved and later on printed
44         InputStreamReader in = new InputStreamReader(conn.getInputStream());
45         // This is to so that our connection can receive the new posting from
               the URL
46         BufferedReader buffer = new BufferedReader(in);
47         // Now buffer can read from the URL server webpage; used for reading
48         StringBuffer fromServer = new StringBuffer();
```

```
49          // This is the string where each line of the new postings from the URL
                will be saved
50          String eachLine=null;
51          // While loop continues until we reach end of the posted content
52          while((eachLine=buffer.readLine()) != null)
53          {
54              fromServer.append(eachLine);
55              fromServer.append(System.lineSeparator());
56          }
57          // After while loop everything from the webpage is now saved into the
                stringBuffer fromServer, which can be printed
58          buffer.close();
59          // We close the stream for any further doing
60
61          // now all the webpage content is saved in the fromServer Stringbuffer
62          // now we just print this fromServer
63          System.out.print("Here is our posted content :\n"+fromServer);
64      }
65  }
```

6 Input/Output

The web page used for this experiment of POST method is "*https://jsonplaceholder.typicode.com/posts/*". Output of the program is given below.

```
Value of http created is: 201
This is the response Code: 201
This is the response Message from server: Created
Here is our posted content :
{
"Hi!!! We have posted something!!! Yay!!!": "",
"id": 101
}
```

7 Discussion & Conclusion

Based on the focused objective(s) to understand about HTTP methods, the additional lab exercise will make the student more confident towards the fulfilment of the objectives(s). Different operations can be done by the client after creating a connection with an web page. Each operation corresponds a certain HTTP method, that is used as input for that operation. HTTP POST method is used if the client wants to post/write any content to any web page. On the other hand, GET method is used, just for reading the entire web page.
It is to be mentioned here, in real world all HTTP servers are extremely conservative. For this reason, not a single web page from the server allows any clients or anyone to write/post something on it. For this purpose, certain web pages exist for students to practice and learn about HTTP POST method. Our given url of the web page for this experiment is one such web page. In addition to this security, JAVA libraries also does not allow this type of alteration of contents for any web page. For this reason, if one wants to actually post something, the **Step 4** of Algorithm 1 needs to be added in the programming to release this security.

8 Lab Task (Please implement/think yourself and show the output to the instructor)

Implement the GET HTTP method using JAVA programming on the web page "*http://webcode.me/*".

8.1 GET method analysis

The GET method is one of the other HTTP methods used by clients upon after getting connected to any web page. This method is mainly used if client simply wants to read the entire contents of any certain web page. The HTTP GET method requests a representation of the specified resource, in this case the specified web page. Requests using GET should only retrieve data for reading.

The S steps and theoretical logic are exactly similar, even less complex, to the mentioned POST method in this experiment. Here no need to enforce any content getting ready since, nothing is to done for writing. Only a simple reading request is pushed towards the server. If server accepts which is verified using the response code 200 by the client, then the entire web page is returned as a string instance, that is finally simply read by the client. The following section simplifies the steps to implement the GET HTTP method for the web page "*http://webcode.me/*".

8.2 GET method Algorithm

Algorithm 2: HTTP GET method implementation

Input: Given in the program by default according to the programmer's requirement
Output: Print of the read content from the web page

Step 1: Create an instance of the **URL** class with the url of the web page in the constructor. Our webpage has the url: "*http://webcode.me/*"
Step 2: Create an instance of the **HTTPURLConnection** class, made as a return from the function **openConnection()** of the url instance.
Step 3: Enable the HTTP method, that the client wants to do, using **setRequestMethod()** function. Here the parameter will be "**GET**".
`/* Here HTTP_OK is a defined value that equals to 200 */`
Step 8: **if** *responseCode == HTTP_OK* **then**
    print out the response code and response messages from the web server
    An empty string is initialized to store the read contents from the connection instance created at Step 2.
    while *reading up to the end of the posted content* **do**
        append each sentence of the posted content with the initailized string.
    end
    Print the created string, that contains the entire content of the web page.
else
    print out an error message and exit from the program.
end
Step 9: Exit

8.3 GET method expected output

Here is the given output for the GET method on the given web page, "*http://webcode.me/*".

```
The response message and code from website is: OK 200
Now we get our website:
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My html page</title>
</head>
<body>
<p>
Today is a beautiful day. We go swimming and fishing.
</p>
<p>
Hello there. How are you?
</p>
</body>
</html>
```

9 Lab Exercise (Submit as a Lab Report)

- Implement the GET HTTP method usign JAVA for the webpage "*http://webcode.me/*".
- Implement the GET HTTP method using JAVA for a simple famous web server's homepage, for example "*www.youtube.com*".
- 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.
- 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 step (as can be seen in Algorithm 2) in implementing GET HTTP method for any web page?

10 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.