Computer Communication and Networking (ITCS-6166/8166)

HTTP Client and Server

HTTP:

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

Client Process:

Client process is the one which keeps looking and asking for information. The client process in our project is used to pass information to server side using the command line and execute the commands in a proper way.

Server Process:

This is the process which takes a request from the clients. After getting a request from the client, it will perform the process asked for, and send data back to the client. After its execution, it becomes ready to serve another client. Server processes are always alert and ready to serve incoming requests.

1. The server creates a socket with the specified port number which it uses to listen to incoming client requests.
2. It accepts multiple client requests trying to connect to the same port, unless it receives a 'shutdown' termination signal.
3. After a client connection is accepted, it reads the HTTP request and splits it into command and filename.

GET:

The HTTP GET method is used to access a representation of a resource. In the "happy" (or non-error) path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400
(BAD REQUEST).

PUT:

PUT is most-often utilized for adding capabilities, PUT-ing to a known resource URL with the request body containing the newly-updated representation of the original resource. However, PUT can also be used to create a resource in the case where the client chooses the resource ID

instead of by the server. In other words, if the PUT is to a URL that contains the value of a non-existent resource ID. Again, the request body contains a resource representation.
On successful update, return 200 (or 204 if not returning any content in the body) from a PUT.
If using PUT for create, return HTTP status 201 on successful creation. A body in the response is optional—providing one consumes more bandwidth. It is not necessary to return a link via a Location header in the creation case since the client already set the resource ID.

Project Execution Instructions (On Linux Ubuntu 16.04):

This project consists of two .java files:
1. Server.java
2. Client.java

To execute the programs, following commands are used:

```
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ javac Client.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$
```

```
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ javac Server.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$
```

After the execution, we get Server.class and Client.class files in the respective folders.

```
ck@ck-Inspiron-5559:~/workspace/HTTPClientServer/src$ javac Final/Server.java
ck@ck-Inspiron-5559:~/workspace/HTTPClientServer/src$ java Final.Server 8000
Server waiting for a connection on address...: localhost/127.0.0.1:8000
```

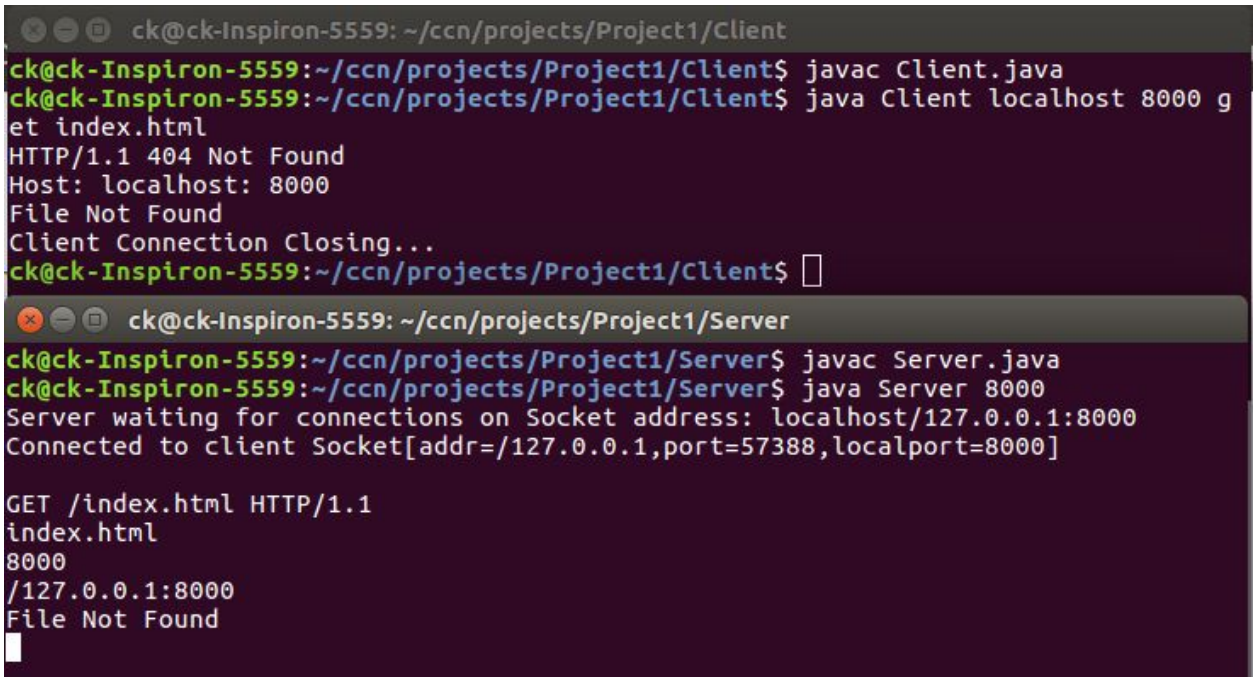
Now, we open the port of the server so that it should continuously listen for the request from the client. Here, we used port 8000.
Now client will request for a file. In our case, client will ask for content of file trial.html which is in Server directory. If file is not present in Server directory then it will show 404 file not found. At server terminal, we get the content of the requested file on client's terminal.

```
ck@ck-Inspiron-5559:~/workspace/HTTPClientServer/src$ java Final.Client localhost 8000 GET index.html
Client: HTTP/1.1 GET request sent to server localhost
HTTP/1.1 200 OK
Host: localhost:8000
```

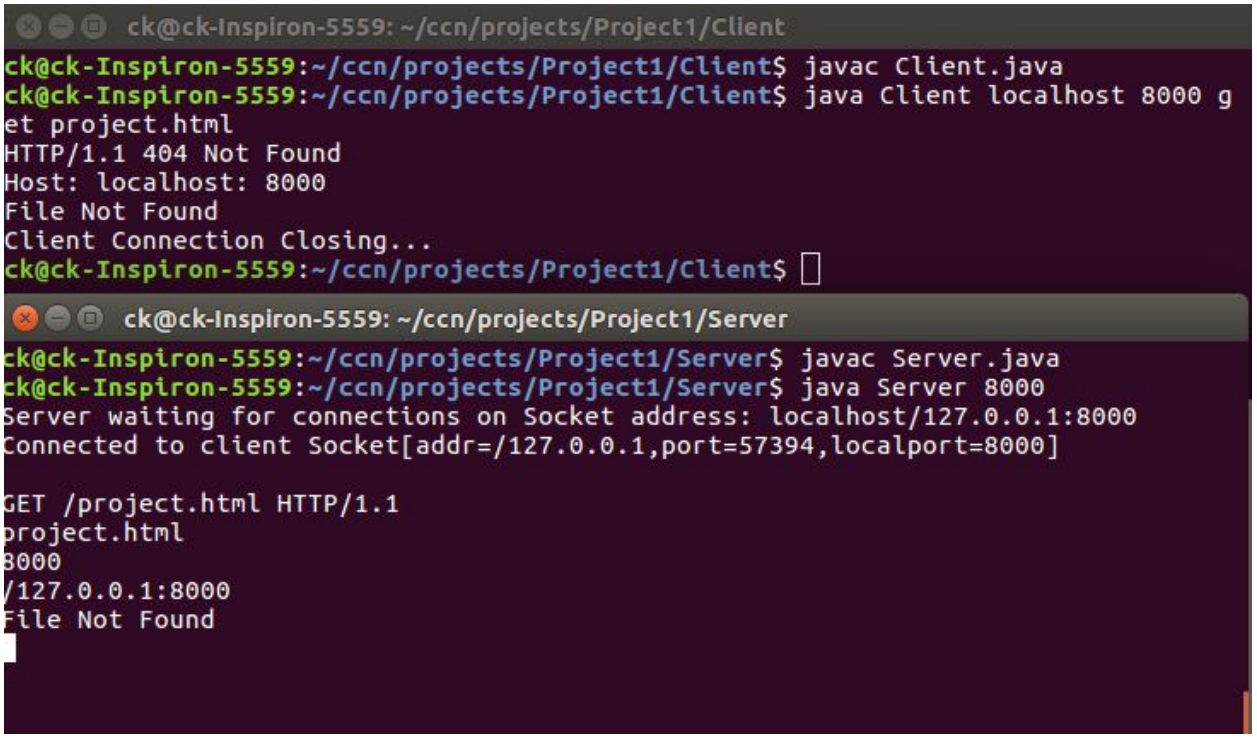
Now, we will use PUT to copy the file at client to the server. In our case, file index.html is client file. Hence it is executed as follows:

After using the PUT command, we can see that the file index.html is stored in Server directory. From this we can say that, file at client is sent to server and saved there.

GET index.html (before PUT)



GET project.html (before PUT)



PUT index.html

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Client
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ javac Client.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ java Client localhost 8000 p
uT index.html
<html>
<h1>
<p>

This is my First HTML File

</p1>
</h1>
</html>

HTTP/1.1 200 OK
Host: localhost: 8000
File saved successfully to Server

Client Connection Closing...
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$
```

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Server
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ javac Server.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ java Server 8000
Server waiting for connections on Socket address: localhost/127.0.0.1:8000
Connected to client Socket[addr=/127.0.0.1,port=57368,localport=8000]

PUT index.html HTTP/1.1
File saved successfully to Server
```

PUT project.html

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Client
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ javac Client.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ java Client localhost 8000 p
uT project.html
<html>
<h1>
<p>

This is Project-1 of CCN

</p1>
</h1>
</html>

HTTP/1.1 200 OK
Host: localhost: 8000
File saved successfully to Server

Client Connection Closing...
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$
```

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Server
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ javac Server.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ java Server 8000
Server waiting for connections on Socket address: localhost/127.0.0.1:8000
Connected to client Socket[addr=/127.0.0.1,port=57370,localport=8000]

PUT project.html HTTP/1.1
File saved successfully to Server
```

GET index.html (after PUT)

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Client
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ javac Client.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ java Client localhost 8000 g
et project.html
HTTP/1.1 200 OK
Host: localhost: 8000
<html>
<h1>
<p>This is Project-1 of CCN
</p>
</h1>
</html>

Client Connection Closing...
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$
```

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Server
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ javac Server.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ java Server 8000
Server waiting for connections on Socket address: localhost/127.0.0.1:8000
Connected to client Socket[addr=/127.0.0.1,port=57444,localport=8000]

GET /project.html HTTP/1.1
project.html
<html>
<h1>
<p>This is Project-1 of CCN
</p>
</h1>
</html>
```

GET index.html (after PUT)

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Client
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ javac Client.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$ java Client localhost 8000 g
et index.html
HTTP/1.1 200 OK
Host: localhost: 8000
<html>
<h1>
<p>This is my First HTML File
</p>
</h1>
</html>

Client Connection Closing...
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Client$
```

```
ck@ck-Inspiron-5559: ~/ccn/projects/Project1/Server
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ javac Server.java
ck@ck-Inspiron-5559:~/ccn/projects/Project1/Server$ java Server 8000
Server waiting for connections on Socket address: localhost/127.0.0.1:8000
Connected to client Socket[addr=/127.0.0.1,port=57448,localport=8000]

GET /index.html HTTP/1.1
index.html
<html>
<h1>
<p>This is my First HTML File
</p>
</h1>
</html>

<html>
```

References:

1. www.google.com
2. www.stackoverflow.com
3. www.oracle.com/technetwork/socket-140484.html
4. www.tutorialspoint.com/java/java_multithreading.htm