CSCI 4131 – Internet Programming

Homework Assignment 4 (Version 3)

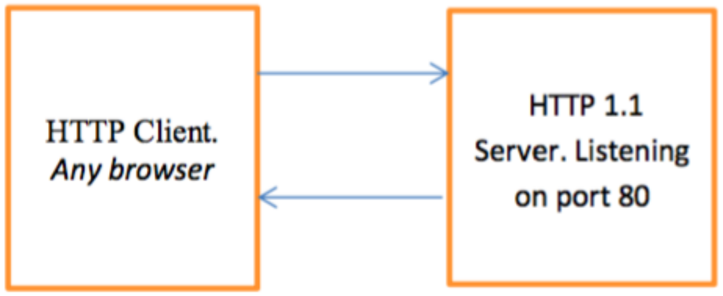
Due 3/02/2018 at 11:55 PM

Submissions after 3/2/2018 at 11:55PM will be accepted through 3/3/2018 at 11:55AM but will be assessed a late-submission penalty

1 Description

The objective of this assignment is to learn HTTP protocol (HTTP1.1) and build a tiny HTTP server. In this assignment you would learn how to program, using Python and TCP sockets, the functionalities of an HTTP client and server. You will need to go through RFC 2616 for HTTP 1.1 protocol details.

When a web client (such as Google Chrome) connects to a web server (such as ***www.google.com***) the interaction between them happens through Hyper Text Transfer Protocol (HTTP).



The working of the basic HTTP server is described below in brief:

1. An HTTP client connects to the HTTP-server and makes a HTTP request to request a web resource.
2. The HTTP Sever parses the request header fields. The request can be of type GET, POST, HEAD, etc. For a GET requests, the server identifies the requested resource (for example an HTML file) and checks if the resource exists.
3. The HTTP server then generates an appropriate HTTP response message. If the requested resource is found, the HTTP server includes successful (2xx) status code in the response headers along with other meta data such as the Content Type and Content Length and sends the resource data as the message body. If the requested resource is not found, then the HTTP Server sends response with status code 404.

2 Preparation and Provided Files

The following files are provided for this assignment:

• 403.html: this file should be sent to the client after forbidden header code.

• 404.html: this file should be sent to client when the requested file is not found.

• private.html: this file is the private file that triggers 403 forbidden code.

• calendar.html: replace this file with your own calendar.html file from Assignment 1.

To give above files proper permissions, please execute following commands after unzipping in your folder:

chmod 640 private.html

chmod 644 403.html

chmod 644 404.html

chmod 644 calendar.html

3 Functionality

When started, your server will establish a socket and bind to a port, listening for connections. You can send a request to the server with your web browser by pointing it at <host>:<port>/calendar.html. In most cases, <host> will be localhost and <port> should default to 9001.

Your code should not use the httplib module of the Python standard library. You should do your own socket programming on this assignment.

When invoked with no arguments, your server will bind to port 9001 and serve requests. It should also accept one optional command line argument which specifies the port to bind to. Two example invocations are:

• *python myServer.py*

• *python myServer.py 9002 Additionally, your server should log any incoming requests to STDIN.*

4 HTTP Protocol

HTTP is a protocol of non-trivial size. We will only be implementing a functional subset of HTTP: GET, HEAD, POST, PUT, DELETE, OPTIONS requests.

4.1 GET Request

GET requests are the most commonly encountered requests. When your web browser requests a webpage from a server, it is issuing a GET request. Here is an example GET request that will be received by your server:

GET /calendar.html HTTP/1.1
Host: localhost:9001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:33.0) Gecko/20100101 Firefox/33.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

For our simplified server, only the first line of the request is important. We need to extract the requested URI (calendar.html) and serve it to the client.

To test:

- Enter the following address in your browser – http://localhost:9001/calendar.html, the browser should display the calendar file that you developed for the first homework.
- Curl command: `curl -i -H "Accept: text/html" http://localhost:9001/calendar.html`

4.2 PUT Request

PUT Request creates a resource in the specified path or modify the existing resource at that path. It sends following responses:

- When a resource is created, it sends a 201 (Created) response and the path of the resource created as follows:
  HTTP/1.1 201 Created
  Content-Location: /new.html
- When a resource is modified, it sends either 200(Ok) and the path of the resource created as follows:
  HTTP/1.1 200 OK
  Content-Location: /new.html

It is almost similar to POST request. However, there is a slight difference between both. Refer https://sookocheff.com/post/api/when-to-use-http-put-and-http-post/
To test:

- Curl Command: `curl -i -X PUT localhost:9001/upload.html -d "htmlBody"`
- Enter the following address in your browser – http://localhost:9001/upload.html, the browser should display the html body that you mentioned.

4.3 DELETE Request

DELETE request deleted a resource from the specified path. It sends the following response:

HTTP/1.1 200 OK
Date: 2018-02-09 03:22:30.463323

To test:

- Enter the following address in your browser – http://localhost:9001/upload.html, the browser should display the html body that you mentioned.
- Curl Command: `curl -X "DELETE" localhost:9001/upload.html -i`
- Enter the following address in your browser – http://localhost:9001/upload.html, the browser should display 404.html.

4.4 OPTIONS Request

OPTIONS request is used to list the communication methods available for the specified resource. The client can specify the target resource in following ways:

- To list down communications methods for a particular resource index.html
  OPTIONS /index.html HTTP/1.1
- To list down communication methods for the whole server
  OPTIONS / HTTP/1.1

The response for the OPTIONS request looks like

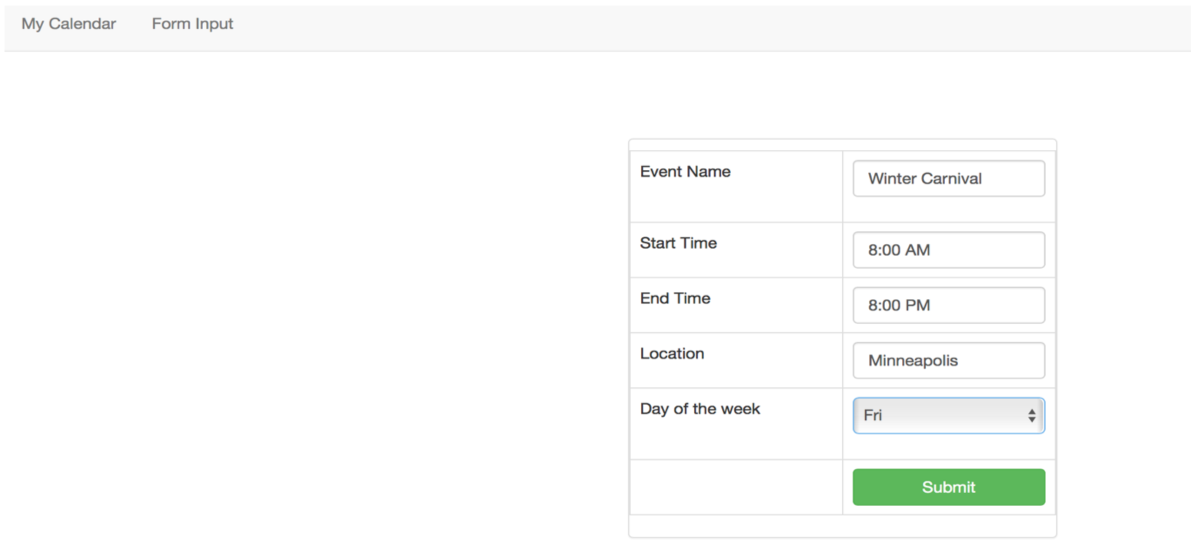
HTTP/1.1 200 OK
Allow: OPTIONS, GET, HEAD, PUT, POST
Cache-Control: max-age=604800
Date: 2018-02-09 03:58:57.654801
Content-Length: 0

To test:

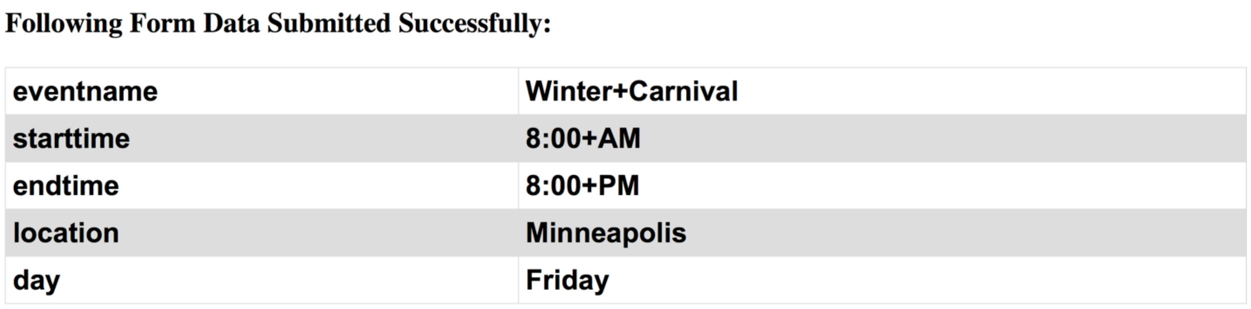
- Curl Command: `curl -X OPTIONS localhost:9002 -I`
- Curl Command: `curl -X OPTIONS localhost:9002/calendar.html -I`

4.5 POST request

- You will use the form that you developed for first homework. In the first homework, the form was submitted to http://www-users.cselabs.umn.edu/~tulaj001/form_handler.php. For this homework, you will instead submit your form to your python HTTP server. You can achieve this by changing the method of the form to “post” and action as http://localhost:9001
- Upon successful submission your, server should return an HTML page with all the submitted information. Following are sample screen-shots:



Event Name	Winter Carnival
Start Time	8:00 AM
End Time	8:00 PM
Location	Minneapolis
Day of the week	Fri



Following Form Data Submitted Successfully:

eventname	Winter+Carnival
starttime	8:00+AM
endtime	8:00+PM
location	Minneapolis
day	Friday

4.6 Redirection Response

- Additionally, you will also send redirection responses for certain URLs. If the request is for a resource named “csumn”, then the client should be redirected to the following location: https://www.cs.umn.edu/. For example if the request is for URL is http://comp1.cs.umn.edu:5555/csumn it should be redirected to the above URL. The browser should be automatically redirected to the new URL. For this, your server should send an appropriate response message with required headers. You would have to include appropriate location headers in the response. The redirection response would include “Permanently moved” status code. Please refer to section 10.3 of RFC 2616 for details.

4.7 Response payload for error conditions

You will need to handle error conditions, as specified below, and also include an appropriate error message (in plain text) in the response body, specific to the error condition.

Your HTTP server should handle following error conditions: 403, 404, 405, and 406. It should send appropriate error responses as specified below.

1. If the requested resource does not have appropriate permissions (i.e. it is not world-readable), 403 error response should be sent

2. If the requested resource is not found, 404 error response should be sent

3. If the request is anything other than GET, HEAD, POST, PUT, DELETE and OPTIONS the server should send 405 – method not allowed response.

4. If the request contains accept headers, and the content characteristics of the requested resource are not acceptable according to the accept headers, for example accept headers specified only html files and the requested entity is an image file, then 406 error response should be sent.

5.Testing Guidelines

To run your HTTPServer, you should pick a port number above 5000. You can test the HTTP server using a HTTP client, a browser and curl command.

6 Submission Instructions

• Submit your server program that is renamed to <UMN x500>_server.py

User *john1234* should submit *john1234_server.py*

• You don't need to provide any extra files. We will copy our own 403.html, 404.html, private.html, calendar.html, form.html when we test your code.

PLEASE ENSURE TO TEST YOUR CODE ON CS LAB MACHINES.

7 Evaluation (Will be decided once we finalize on which HTTP Protocols to keep)

Your submission will be graded out of 100 points on the following items:

- Server establishes a socket and binds to 9001 by default. (**5 points)**
- Server accepts a parameter to change port. (**5 points)**
- Server accepts connections from clients and reads incoming messages. (**10 points)**
- Server correctly identifies GET requests. (**10 points)**
- Server correctly processes GET requests for binary files like images. **(Bonus 5 points)**
- Server correctly processes POST requests. (**10 points)**
- Server correctly processes PUT requests. **(Bonus 10 points)**
- Server correctly processes DELETE requests. **(Bonus 10 points)**

- Server correctly processes OPTION requests. **10 points**
    - /calendar.html - Gives OPTIONS, GET, HEAD
    - /form.html - Gives OPTIONS, GET, HEAD, POST
    - / - Gives OPTIONS, GET, HEAD, POST
    - / (If you attempt Bonus) - Gives OPTIONS, GET, HEAD, POST, PUT, DELETE
- Server correctly responds with 200 and serves requested page. (**5 points)**
- Server correctly responds with 406. (**5 points)**
- Server correctly responds with 405. (**5 points)**
- Server correctly responds with 403. (**5 points**)
- Server correctly responds with 404. (**5 points)**
- Server redirection 301. (**10 points)**
- Server logs requests to STDIN. (**5 points)**
- Source code is documented and readable. (**5 points)**
- Submission instructions are followed. (**5 points)**

Reminder: All assignments will be graded on the cselabs machines - so make sure to test your server on a cselabs machine to ensure it functions correctly as specified in the evaluation criterial above.