

# CSCI 4131 – Internet Programming

## Homework Assignment 4

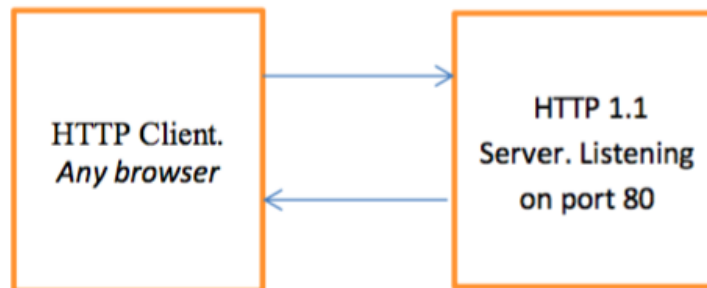
**Due 3/02/2018 at 11:55 PM**

*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, using Python and TCP sockets, you will program the functionality of an HTTP server. You will need to go through RFC 2616

(<https://www.w3.org/Protocols/rfc2616/rfc2616.html>) for specific details on the HTTP protocol. When a web client (such as Google Chrome) connects to a web server (such as [www.google.com](http://www.google.com)) the interaction between them is governed by the Hyper Text Transfer Protocol (HTTP).



The basic behavior of an HTTP server is described briefly below:

1. A client sends an HTTP request to an HTTP-server requesting a resource.
2. The HTTP Server parses the request header fields. The first piece of information the HTTP Server looks for is the request Method. The request can be of type GET, POST, etc. For a GET request, the server extracts the name and location of the requested resource in the request message (for example an HTML file) and checks if the resource exists.
3. The HTTP server then generates an appropriate HTTP response message as follows. If the requested resource exists and is accessible (permissions enable access) 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, appends the resource data to the message body, and sends the HTTP response message to the client. If the requested resource is not found, then the HTTP Server sends an HTTP response message with status code 404. If the resource is found, but permissions don't permit access, the HTTP server sends an HTTP response message with status code 403.

## 2 Files Provided, and Preparations Required

The following files are provided for this assignment:

- 403.html: this file should be sent to the client if permissions don't permit access.
- 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.

Please execute following commands after unzipping in your folder to set the permissions correctly:

```
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 from your web browser by entering the URL **<host>:<port>/calendar.html** in the browser's address bar. For this assignment, in most cases, **<host>** will be localhost and **<port>** should default to 9001.

**Your code should not use the http lib module of the Python standard library. You should do your own socket programming on this assignment (examples will be presented in class)**

When called with no arguments, your server will bind to port 9001 and serve requests. It should also accept one optional command line argument which specifies a port to bind to. The ways to start your server will be:

- 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 request methods, specifically: the GET, POST, PUT, DELETE, and OPTIONS methods.

## 4.1 GET Request

GET requests are the most commonly used HTTP request method. 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 your simplified server, only the first line of the request is important. Your program needs 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.
- Use the following Curl command:  
`curl -i -H "Accept: text/html" http://localhost:9001/calendar.html`

## 4.2 PUT Request

A PUT Request creates a resource in the specified path or modifies the existing resource specified by the path. It sends following responses:

- When a resource is created, it sends a 201 (resource 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*

A PUT request is very similar to POST request, but there is a slight difference between them. Refer <https://sookocheff.com/post/api/when-to-use-http-put-and-http-post/> for details.

To test:

- i. Use the Curl Command:  
`curl -i -X PUT localhost:9001upload.html -d "htmlBody"`
- ii. Enter the following address in your browser – <http://localhost:9001/upload.html>, and 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 (assuming you have the file upload.html in your server's top-level directory):

- i. Enter the following address in your browser – <http://localhost:9001/upload.html>, the browser should display the html from the file upload.html.
- ii. Issue the Curl Command:  
`curl -X "DELETE" localhost:9001/database/upload.html -i`
- iii. Enter the following address in your browser – <http://localhost:9001/upload.html>, the browser should display 404.html.

## 4.4 OPTIONS Request

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

- To list the HTTP methods available for a particular resource named `index.html`, the following HTTP request is required:  
*OPTIONS /index.html HTTP/1.1*
- To list the HTTPs methods available 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*

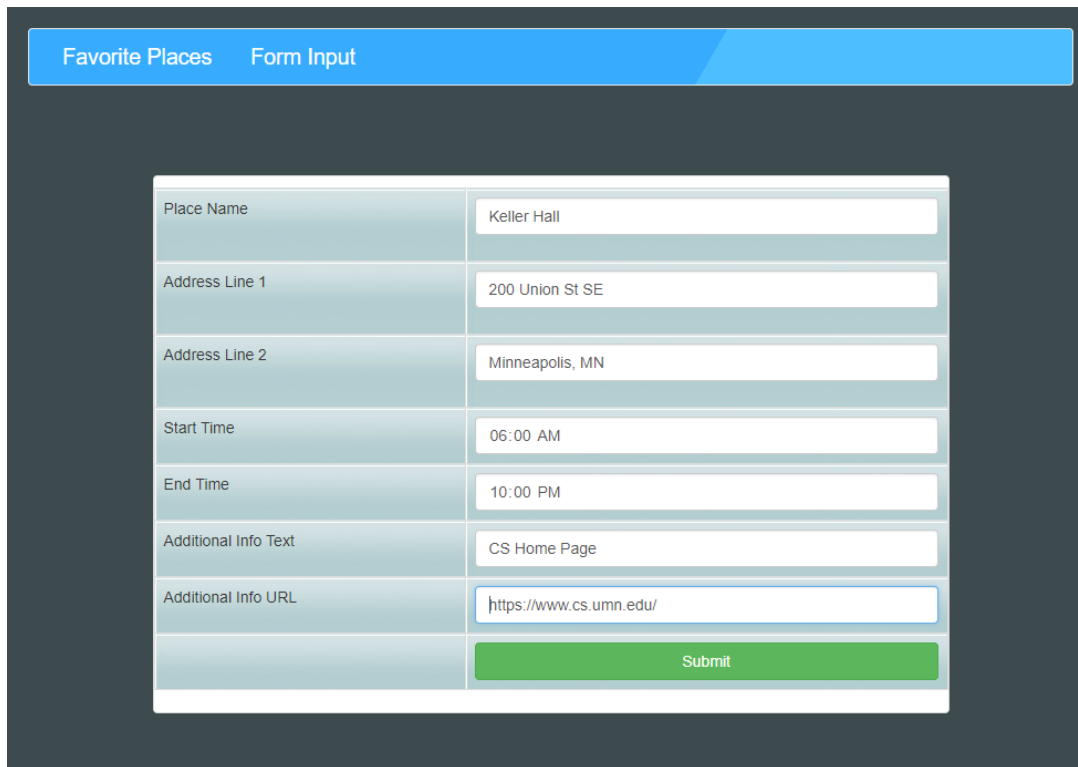
To test, issue the following Curl Command:

`curl -X OPTIONS localhost:9001 -i`

## 4.5 POST request

- For this homework, you will instead submit the HTMLL form you created in your previous assignments to your python HTTP server. You can achieve this by changing the method of the form to POST and the action on your form to: <http://localhost:9001>
- Upon successful submission your, server should return an HTML page with all the submitted information. The following screen shots depict the request from browser, and the information from your server's response message rendered in the browser

## Form Submitted by Browser



The screenshot shows a web application with a dark grey background. At the top, there is a blue navigation bar with two tabs: 'Favorite Places' and 'Form Input'. Below the navigation bar, a form is displayed with a light blue border. The form contains several input fields, each with a label on the left and a text input on the right. The fields are: 'Place Name' with the value 'Keller Hall', 'Address Line 1' with '200 Union St SE', 'Address Line 2' with 'Minneapolis, MN', 'Start Time' with '06:00 AM', 'End Time' with '10:00 PM', 'Additional Info Text' with 'CS Home Page', and 'Additional Info URL' with 'https://www.cs.umn.edu/'. At the bottom of the form is a green 'Submit' button.

Label	Value
Place Name	Keller Hall
Address Line 1	200 Union St SE
Address Line 2	Minneapolis, MN
Start Time	06:00 AM
End Time	10:00 PM
Additional Info Text	CS Home Page
Additional Info URL	https://www.cs.umn.edu/

## Browser 's Display of Information sent in response to POST request

Following Form Data Submitted Successfully

Place Name: Keller Hall

Address Line 1: 200 Union St SE

Address Line 2: Minneapolis, MN

Open Time: 06:00 AM

Close Time: 10:00 PM

Additional Info: CS Home Page

URL: https://www.cs.umn.edu /

## 4.6 Redirection Response

- Your server will send redirection responses in response to requests involving 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 a request includes the URL

<http://comp1.cs.umn.edu:5555/csumn>

the request should be redirected to <https://www.cs.umn.edu>

as result, your browser should be automatically redirected to the new URL. To accomplish this, your server should send an appropriate response message with required headers. You will have to include appropriate location headers in the response. The redirection response should 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 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 HTTP Server, you should pick a port number above 5000. You can test the HTTP server using a HTTP client, a browser and the curl command.

## 6 Submission Instructions

- Submit your server program that is renamed to <UMN x500> myServer.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 Criteria

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 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**
  - /MyCalendar.html - Gives OPTIONS, GET, HEAD
  - /myform.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.**