Test Cases

This document is intended to provided an overview of the test cases run on the http client and server. The tests in this repository use the python `mock` module, which allows us to mock return values from methods in a test environment (ie, we can run the client without gathering arguments from the user). The tests are written with Python's `unittest`.

Client tests

The first test makes a GET request to an http server running on localhost (the server provided in this repository), and checks that the response received is a 200 OK response. This is really an integration test between the two, which is great for proving how the two systems work together. However, to prove that the HTTP client really works and we haven't designed both systems incorrectly, we need another test. The server must be running on localhost:5678 to run this test.

The second test makes a GET request to CNN requesting `index.html` on destination port 80. Again, we check that the message that is returned is a 200 OK HTTP message. This test is a better indication that the client is working properly, since the request went out to a publicly available server. If this test passes, we have a very high level of confidence that our client has sent a properly formatted HTTP request and received the response.

Finally, we test a PUT request to the local server. This test looks for a 200 series code from the server (anything in 200, 201, 204 since all those are reasonable responses from the server). This is again more of an integration test and could still pass if both systems were built incorrectly in the same way. However, it's difficult to test a PUT request with an external server as most will either reject the method or require authentication. The server must be running on localhost:5678 to run this test.

Test output:

```
brianreid@Brians-MBP-2:~/grad_school/data_comm/http-sockets/(master).●●>
python3 tests/test_client.py
Ready to send the following HTTP request:
GET index.html HTTP/1.1
Host: www.cnn.com


Request sent!
Response received!
.Ready to send the following HTTP request:
GET index.html HTTP/1.1
Host: localhost


Request sent!
Response received!
.about to send test.txt...
Request to send:
PUT test.txt HTTP/1.1
```

```
Host: localhost

this is a test file
send a PUT request to the http sever with the name of this file to
create or overwrite this asset in myserver/static
sending...
Done sending
Response received!
.
----------------------------------------------------------------------
Ran 3 tests in 0.520s

OK
```

Server tests

Now that we are confident that the client is sending valid HTTP requests and working properly with TCP connections, we can use it to help us test the server by using its `main()` function to send requests to the server. There are three tests here as well. To run these tests, make sure the server is already running.

The first test mocks a request to the server asking to GET `index.html`. Since this file ships with the module in `myserver/static`, we check for a 200 status code in the response that the server gives.

The second test sends a request to the server for a file called `foo.hmtl`. Unless you have added it yourself, this file does not exists within `myserver/static`. Therefore we expect the server to respond with a 404.

The third test makes a PUT request to the server. Once this request is complete, we check a number of things:

- the status code is 201 (this file did not already exist and it was created)
- the filepath now exists in `myserver/static`
- the content of the newly written file is the same as the content in the file that exists "client-side" (in the `myclient/static` folder)

Ideally I would like to have written an automated test to ensure the correct behavior of the server when it gets a bad or unsupported request, but I did not have the time.

Test output:

```
1 brianreid@Brians-MBP-2:~/grad_school/data_comm/http-sockets/(master).>
python3 tests/test_server.py
Ready to send the following HTTP request:
GET foo.html HTTP/1.1
Host: localhost


Request sent!
Response received!
.Ready to send the following HTTP request:
GET index.html HTTP/1.1
```

```
Host: localhost


Request sent!
Response received!
.about to send test.txt...
Request to send:
PUT test.txt HTTP/1.1
Host: localhost

this is a test file
send a PUT request to the http sever with the name of this file to
create or overwrite this asset in myserver/static
sending...
Done sending
Response received!
.
----------------------------------------------------------------------
Ran 3 tests in 0.044s

OK
```