Lab 2: Using flask to build a HTTP server

50.012 Networks

Hand-out: September 20
Hand-in: September 27

1 Objectives

- Learn more on HTTP request/response handling
- Familiarize yourself with flask, an HTTP API framework
- Implement a simple HTTP server in Python

2 Notes

- You can collaborate with another student, please hand in code individually with both authors noted in the header

3 Set up of your machine

- Connect to `SUTD_student` over wireless, test that you have Internet access with `ping` or similar
- Install flask with `sudo pip install Flask`

4 Introduction

- In this assignment, you will implement a simple HTTP server using flask/Python
- A nice tutorial can be found here:

  `https://www.raspberrypi.org/learning/python-web-server-with-flask/worksheet/`

  - Ignore the parts about raspberry and the editor application.
- As editor, a simple one is `mousepad`, or `nano` (in the terminal). `vim` and `emacs` are the traditional advanced editors in Linux. For some reason, SUTD students like `sublime` (with annoying prompt to pay). `atom` looks interesting.
- If you arrive at the "Browsing on other devices" part, you should be able to connect to your desktop's private IP with your mobile phone, if it is connected to SUTD$_{\text{student}}$

5 What to implement

- Follow the steps in the tutorial to the end to understand the general setup
- Decide on a (fictional/real) website you want to implement. It should have 3 individual resources that can be queried.
- At least one of your resources should have dynamic content - maybe your server uses some kind of database, or you process user input? There are many options!
  - Maybe you have a `news/<topic>` resource, and some datastructure to store news
  - You could have individual pages for different users under a `user/<name>` resource
  - You could even load remote content and embed it into your website (more on that next time).
- Each resource should use a different function on the server to generate the response (don't just use one template+function for all).
- To interact with your HTTP server, we suggest a browser, `curl` or Python with `requests` library
  - For a nice `requests` tutorial, see here: `http://www.python-requests.org/en/latest/user/quickstart/`
- For debugging the server/communication with the server, try using `wireshark`
- For fun, you can also use `telnet <YOURIP> 5000` to manually request content from your webserver
- Depending on how fluent you are in HTML, your website could use more advanced styles. For example, `https://html5up.net/` has very nice site templates, that can be converted to jinja templates with little effort. Most of the magic comes from the supplied CSS files.
  - Note: flask will serve local files in the `static` folder. It is easiest to move all images and css in that folder/subfolders of that folder. You will then also need to update references to those resources in the hmtl5up templates.

6 What to Hand in

6.1 eDimension submission:

- You will submit in the complete server code via eDimension. You can collaborate with a friend on the code, in that case please state both your names in the comments at the start of the file. Both students will then submit the file individually.
- If you use third party templates (e.g., from html5up) please cite the sources

6.2 Checkoff:

- Demo your Python code to the TA and explain it. In particular, show how you retrieve webpages by using your browser.