CSCD 330 - Computer Networks

Lab 3, Flask Server

Overview

Write an API server using the flask library. The server must have 3 unique API calls. One for converting from a domain name into a business address. Another for converting from a domain name into the weather(**NOT hourly**) forecast. Finally, a call for returning the network range of IP addresses associated with the domain.

Instructions

Complete the above program following the steps in the next section. You must use the same APIs as used in lab 2 (excluding those needed to graph). Call this file lab3.py and include your name at the top of the file in a comment – `#author: YOUR NAME`. Your output should match the provided example output. You must also create a README explaining how to run your program and answering the questions below. Furthermore, you must create a test script in `bash`. Use `curl` for testing. Call this file test.sh. **Assume that we will start your server before running your bash script.**

Flask can be installed via: `sudo apt-get install python3-flask`

You may only use the following imports:

```python
from flask import Flask
from json import loads
from requests import get
from socket import gethostbyname
from subprocess import getstatusoutput
```

Steps:

1. Create the API server:

The flask server must support 3 API calls to the following specifications:

1. Accepts a URL in the format `/address/\<domain_name\>` and returns the physical address from the `whois` entry.

2. Accepts a URL in the format `/weather/\<domain_name\>` and returns the weather using the same API from lab 2 but the forecast and not the hourly forecast.
3. Accepts a URL in the format `/range/\<domain_name\>` and returns the IP range of the domain. `whois` can help.

Note: whenever you modify the server, you must restart the server to see the affects.

2. Cache requests:

To save on network traffic, cache requests such that duplicate calls are not re-run. Instead return the original response prefaced with "Cached:" so that we know the cache is working. Python dictionaries work well for this.

Questions:

1. Identify the following in the URL: http://localhost:5000/weather/google.com
   - Domain:
   - Path:
   - Port:
   - Protocol:
2. Identify the following in the URL: https://translate.google.com/
   - Domain:
   - Subdomain:
   - TLD:
   - Path:
   - Protocol:
   - Port:
3. What is a Python decorator?
4. Is there any problem with your cache implementation? Would your cache implemenation work in production?

Turn in:

Submit a tarball with the following:

- Your source code (in python) called `lab3.py`
- Your test script (in bash) called test.sh – test at least 3 inputs.
- Your README answering the above questions and explaining how to run your program.

In case you have forgotten: `tar -czvf lab3_YOURNAME.tar.gz *.py *.sh README`

Example output:

```
curl localhost:5000/address/google.com
    1600 Amphitheatre Parkway, Mountain View, CA, 94043
curl localhost:5000/range/google.com
    Network range for google.com is 142.250.0.0 - 142.251.255.255
curl localhost:5000/address/google.com
    Cached: 1600 Amphitheatre Parkway, Mountain View, CA, 94043
curl localhost:5000/weather/google.com
    Sunny, with a high near 74. North wind 2 to 10 mph.
curl localhost:5000/weather/google.com
    Cached: Sunny, with a high near 74. North wind 2 to 10 mph.
```