

Assignment 1

Fall 2014

CS595 Web Science

Dr. Michael Nelson

Mathew Chaney

September 11, 2014

Contents

1	Question 1	3
1.1	Question	3
1.2	Resources	3
1.3	Answer	3
2	Question 2	4
2.1	Question	4
2.2	Resources	4
2.3	Source	4
2.4	Answer	5
3	Question 3	6
3.1	Question	6
3.2	Resources	6
3.3	Answer	6

List of Figures

1	Screenshot of curl POST result	3
---	------------------------------------------	---

1 Question 1

1.1 Question

Demonstrate that you know how to use "curl" well enough to correctly POST data to a form. Show that the HTML response that is returned is "correct". That is, the server should take the arguments you POSTed and build a response accordingly. Save the HTML response to a file and then view that file in a browser and take a screen shot.

1.2 Resources

- L^AT_EX: http://www.electronics.oulu.fi/latex/examples/example_1
- L^AT_EX: <http://scott.sherrillmix.com/blog/programmer/displaying-code-in-latex/>
- curl: <http://curl.haxx.se/docs/httpscripting.html#POST>

1.3 Answer

Using a simple wiki server built from the Go language net/http package tutorial, found here: <https://golang.org/doc/articles/wiki/>

```
1 [mchaney@mchaney-d gowiki]$ curl -v -d "body=something something" localhost:8080/save/  
  TestPage  
2 * About to connect() to localhost port 8080 (#0)  
3 *   Trying ::1...  
4 * connected  
5 * Connected to localhost (::1) port 8080 (#0)  
6 > POST /save/TestPage HTTP/1.1  
7 > User-Agent: curl/7.27.0  
8 > Host: localhost:8080  
9 > Accept: */*  
10 > Content-Length: 24  
11 > Content-Type: application/x-www-form-urlencoded  
12 >  
13 * upload completely sent off: 24 out of 24 bytes  
14 < HTTP/1.1 302 Found  
15 < Location: /view/TestPage  
16 < Date: Fri, 29 Aug 2014 13:32:01 GMT  
17 < Content-Length: 0  
18 < Content-Type: text/plain; charset=utf-8  
19 <  
20 * Connection #0 to host localhost left intact  
21 * Closing connection #0
```

The wiki uses the `/save/` url to save the contents of a text box as the body of the target page.

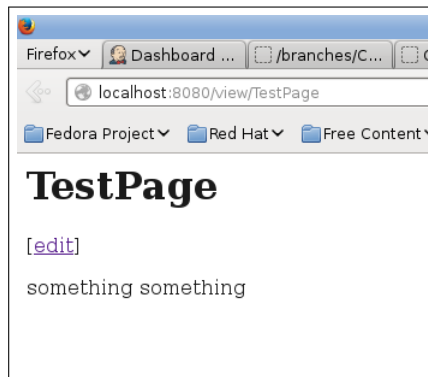


Figure 1: Screenshot of curl POST result

2 Question 2

2.1 Question

Write a Python program that:

1. takes one argument, like "Old Dominion" or "Virginia Tech"
2. takes another argument specified in seconds (e.g., "60" for one minute).
3. takes a URI as a third argument:
`http://sports.yahoo.com/college-football/scoreboard/`
or
`http://sports.yahoo.com/college-football/scoreboard/?week=2&conf=all`
or
`http://sports.yahoo.com/college-football/scoreboard/?week=1&conf=72`
etc.
4. dereferences the URI, finds the game corresponding to the team argument, prints out the current score (e.g., "Old Dominion 27, East Carolina 17), sleeps for the specified seconds, and then repeats (until control-C is hit).

2.2 Resources

I used BeautifulSoup and Requests to write a bot that plays the game <http://www.kingdomofloathing.com/>. Code for which can be found @ <http://www.github.com/mattchaney/meatmachine/>. Both of these modules were useful for this assignment.

- Requests: <http://docs.python-requests.org/en/latest/>
- BeautifulSoup: <http://www.crummy.com/software/BeautifulSoup/bs4/doc/>

2.3 Source

```
1 #!/usr/bin/env python
2
3 import sys
4 import requests
5 import time
6 import re
7 from bs4 import BeautifulSoup
8
9 if __name__ == "__main__":
10     if len(sys.argv) != 4:
11         print "Usage:\n\tpython getscore.py [school] [seconds] [uri]\n"
12         sys.exit()
13     school = sys.argv[1]
14     try:
15         period = float(sys.argv[2])
16     except ValueError:
17         print "period must be a valid float value"
18         sys.exit()
19     uri = sys.argv[3]
20     while True:
21         try:
22             response = requests.get(uri)
23         except Exception as e:
24             print e
25             sys.exit()
26     soup = BeautifulSoup(response.content)
27     gametag = soup.find('em', text=re.compile('^'+school+'$'))
28     if not gametag:
29         print "Game not found"
30         sys.exit()
31     game = gametag.parent.parent.parent
```

```

32     try:
33         teams = [game.find_all('span', {'class': 'team'})[0].em.text, game.find_all('span',
34             {'class': 'team'})[1].em.text]
35         scores = [game.find('span', {'class': 'away'}).text, game.find('span', {'class': 'home'})
36             .text]
37     except:
38         print "Game not found"
39         sys.exit()
40     print teams[0] + ' ' + scores[0] + ', ' + teams[1] + ' ' + scores[1]
41     time.sleep(period)

```

2.4 Answer

```

1 [mchaney@mchaney-d q2]$ python getscore.py "Texas A&M" 10 "http://sports.yahoo.com/college-
   football/scoreboard/"
2 Texas A&M 52, South Carolina 28
3 Texas A&M 52, South Carolina 28
4 ^CTraceback (most recent call last):
5   File "getscore.py", line 26, in <module>
6     time.sleep(period)
7 KeyboardInterrupt
8 [mchaney@mchaney-d q2]$ python getscore.py "alskds" 5 "http://sports.yahoo.com/college-
   football/scoreboard/"
9 Game not found
10 [mchaney@mchaney-d q2]$ python getscore.py "Texas A&M" "http://sports.yahoo.com/college-
   football/scoreboard/"
11 Usage:
12     python getscore.py [school] [period] [uri]
13
14 [mchaney@mchaney-d q2]$

```

3 Question 3

3.1 Question

Consider the "bow-tie" graph in the Broder et al. paper (fig 9): <http://www9.org/w9cdrom/160/160.html>

Now consider the following graph:

```
A --> B
B --> C
C --> D
C --> A
C --> G
E --> F
G --> C
G --> H
I --> H
I --> J
I --> K
J --> D
L --> D
M --> A
M --> N
N --> D
```

3.2 Resources

- Graph Structure in the web: <http://www9.org/w9cdrom/160/160.html>
- Stanford, The web graph: <http://nlp.stanford.edu/IR-book/html/htmledition/the-web-graph-1.html>
- Notes from the class:
 - SCC: Strongly Connected Component - all contained nodes are interconnected
 - IN: Connects into SCC, but not out from SCC
 - OUT: Connects out from SCC, but not in to SCC
 - Tendrils: In or out excluding all SCC
 - Tube: IN->OUT or OUT->IN connection
 - Disconnected: Not connected to other sites

3.3 Answer

For the above graph, give the values for:

IN: A, B, C, G
SCC: M
OUT: D, H
Tendrils: L, K, I, J
Tubes: N
Disconnected: E, F