Assignment 1

CS 432: Introduction to Web Science

Spring 2017

Jason Howse

Finished 26JAN2017

1. 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.

I used a YouTube video by “Kris Occhpinti” titled “BASH lesson tutorial Post Web Form with Curl” to learn how to post to a form using curl after setting curl up on my linux machine.[[1]](#endnote-1)

To demonstrate using curl to post to a form I created a form in google forms. After this I used the curl command “curl” followed by -d. -d sends specified data in a POST request to the HTTP server.[[2]](#endnote-2) I pulled the html code for the form I wanted to edit and extracted the action for the form “entry.1873578523=” and added some text. I added &submit=Submit so that the text would be submitted to the form and then followed by the URI for the form to be submitted. (see figure 1) The result of the execution of that line was a wall of html code that resulted in figure 2.

Figure 1 User Input

curl -d "entry.1873578523=some of the things&submit=Submit" "<https://docs.google.com/forms/d/e/1FAIpQLSe7ckfuSj5gc7tfEkYkSZnwMRML9IZ3iideULeu-B9I8E6oXg/formResponse>"

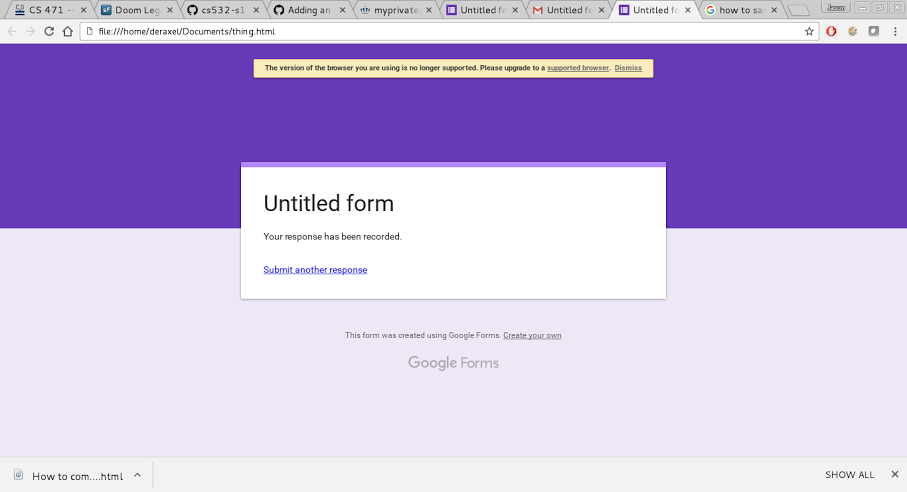


Figure 2 Curl output html result

1. Write a Python program that:

1. takes as a command line argument a web page

2. extracts all the links from the page

3. lists all the links that result in PDF files, and prints out the bytes for each of the links. (note: be sure to follow all the redirects until the link terminates with a "200 OK".)

4. show that the program works on 3 different URIs, one of which needs to be: <http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html>

see dotsave.py

To do this I created a definition in python called getlinks with a string input that is inserted in the parenthesis. Get links stores the user named website’s name as a string then opens that link and stores a string off all the html code that website has. The program then creates a new string array and string to get the domain of the original website and stores the domain as a string. This is used later in case any pdf links like that in the site <https://www.odu.edu/about/visitors/campus-map> contains any pdf links without the domain attached so the program can rebuild the link so it can access the file. I used regex three times to parse code. The first time searched the string to get either http or ftp followed by characters then a dot character then maybe a single character then two other characters and finally a forward slash. I used the websites to help me determine the characters to put in the parser.[[3]](#endnote-3)&[[4]](#endnote-4) All the pdf strings in the new pdf array are then trimmed by a loop so their contents can be analyzed and if necessary, the domain name is added to the front end of the string. After all the strings have been altered correctly they are then printed with the byte size of the file the URI indicates. An exception is made for pdfs that cannot retrieve a byte size due to damaged links and those links are skipped. Figure 3 is the output for the program in dotsave.py when three links are inputted into the command line. The three links I used were <https://www.odu.edu/about/visitors/campus-map>, <http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html>, and <https://www.odu.edu/academics/academic-records/transcripts>.

Figure 3 Outputs for dotsave.getinks(\*)

=============================== RESTART: Shell ===============================

>>> import dotsave

>>> dotsave.getlinks("https://www.odu.edu/about/visitors/campus-map")

https://www.odu.edu/content/dam/odu/images/maps/campus/campus-map.pdf is 38809925 bytes

>>> dotsave.getlinks("http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html")

http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf is 2184076 bytes

http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf is 622981 bytes

http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf is 4308768 bytes

http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf is 1274604 bytes

http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf is 639001 bytes

http://www.cs.odu.edu/~mln/pubs/jcdl-2014/jcdl-2014-brunelle-damage.pdf is 2205546 bytes

http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf is 1254605 bytes

http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf is 709420 bytes

http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf is 2350603 bytes

>>> dotsave.getlinks("https://www.odu.edu/academics/academic-records/transcripts")

https://www.odu.edu/content/dam/odu/offices/university-registrar1/docs/transcript-request-form.pdf is 97729 bytes

https://www.odu.edu/content/dam/odu/offices/university-registrar1/docs/transcript-request-form.pdf is 97729 bytes

https://www.odu.edu/content/dam/odu/offices/university-registrar1/docs/transcript-request-form.pdf is 97729 bytes

1. 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 --> K

L --> D

M --> A

M --> N

N --> D

O --> A

P --> G

For the above graph, give the values for:

IN:

SCC:

OUT:

Tendrils:

Tubes:

Disconnected:

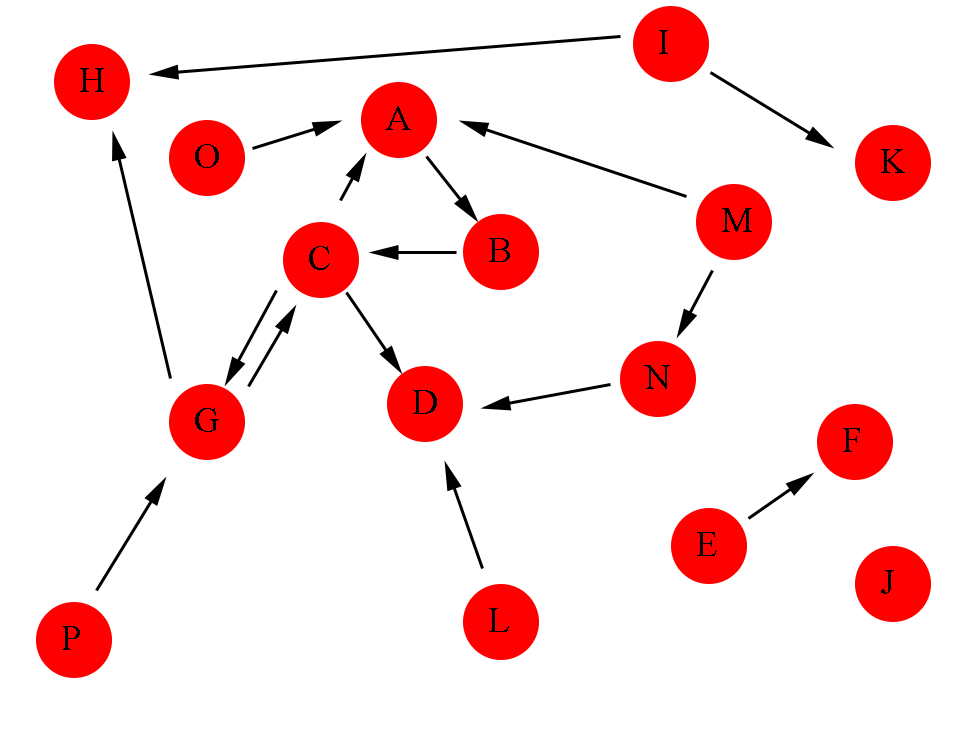
For this I created figure 4 in order to visualize all the ins and outs of the above graph. Once I did this it is easy to see that pages A, B, and C are part of the SSC. Because G links back to C it is included in the SSC. The ins are identified by any pages that links to any of the SSC which include M, O, and P. Any page that is linked to by the SSC pages are OUT which include D and H. OUT side tendrils and easy to Identify as they only link to D or H or are linked through a tendril link. These Tendril links are I, K, and L. The TUBE is any link that connects a page on the IN side to a page on the OUT side and the only candidate for that is page N. The last Pages were the Disconnected pages and these pages are defined by having no links that connect them to the SCC, OUT, IN, or any Tendrils. These pages include E, F, and J. 

Figure 4 Visualization of all the pages and their connections

IN: MOP

SCC: ABCG

OUT: DH

Tendrils: IKL

Tubes: N

Disconnected: EFJ

References

1. M. (2009, September 15). Retrieved January 26, 2017, from https://www.youtube.com/watch?v=Im72HDtqkFs [↑](#endnote-ref-1)
2. Curl.1 the man page. (n.d.). Retrieved January 26, 2017, from https://curl.haxx.se/docs/manpage.html [↑](#endnote-ref-2)
3. 7.2. re — Regular expression operations. (n.d.). Retrieved January 26, 2017, from https://docs.python.org/3.1/library/re.html [↑](#endnote-ref-3)
4. Python Regular Expressions  |  Google for Education  |  Google Developers. (n.d.). Retrieved January 26, 2017, from https://developers.google.com/edu/python/regular-expressions [↑](#endnote-ref-4)