

Assignment Three: My API

Assumptions

Below is a list of assumptions about the assignment gathered from the PDF:

- Calls to the web service will be from the host running “services.py”
- Responses from the Marvel API will be stored as a text document

Implementation

Access to the Canvas and Marvel API's and HTTP Authentication were implemented using Flask. Additionally, the time, hashlib, and json libraries were used to format both the requests and responses.

```
2 from flask import Flask, request
3 from flask_httpauth import HTTPBasicAuth
4 from werkzeug.security import generate_password_hash, check_password_hash
5 from ServicesKeys import *
6 import requests
7 import sys
8 import hashlib
9 import time
10 import json
11
```

Figure 1: Libraries used for the assignment.

HTTP Authentication was derived from the basic example given in the Flask documentation.

```
15 # authorized users to access flask
16 users = {
17     "admin": generate_password_hash("secret"),
18     "james": generate_password_hash("pi")
19 }
20
21 # authorized function from flask example
22 @auth.verify_password
23 def verify_password(username, password):
24     if username in users:
25         return check_password_hash(users.get(username), password)
26     print('>Could not verify your access level for that URL. You have to login with proper credentials')
27     return False
28
```

Figure 2: Basic HTTP Authentication implementation

Requests to the Canvas and Marvel APIs were formatted in much the same way.

```

30 # Canvas Route
31 @app.route('/Canvas')
32 @auth.login_required
33 def getCanvas():
34     # building url to send to canvas
35     filename = request.args.get('file')
36     reqstring = 'https://vt.instructure.com/api/v1/courses/%s/files/?search_term=%s&access_token=%s' % (104692, filename, canvastoken)
37     r = requests.get(reqstring)
38     # downloading file from canvas
39     canjson = json.loads(r.text[1:-1])
40     url = canjson['url']
41     canobj = requests.get(url, allow_redirects=True).content
42     f = open(filename, 'wb').write(canobj)
43     return (r.text, r.status_code, r.headers.items())
44
45 # Marvel Route
46 @app.route('/Marvel')
47 @auth.login_required
48 def getMarvel():
49     # building url to send to marvel
50     storynum = request.args.get('story')
51     ts = str(time.time())
52     hash = hashlib.md5((ts + marvelprivkey + marvelpubkey).encode()).hexdigest()
53     reqstring = 'http://gateway.marvel.com/v1/public/stories/{story}?apikey={apikey}&hash={hash}&ts={ts}'.format(story=storynum, apikey=marvelpubkey, hash=hash, ts=ts)
54     r = requests.get(reqstring)
55     # writing file to directory
56     f = open("Story" + str(storynum) + ".txt", 'w+')
57     f.write(r.text)
58     return (r.text, r.status_code, r.headers.items())

```

Figure 3: Requesting and Receiving from the APIs

However, receiving from the Canvas API was different as a second request was made to download the found file from Canvas. Responses from the Marvel API were stored as a text file.

Results

The project was completed in its entirety and works according to the project specifications and the above assumptions.

Evidence of the project working properly on the Raspberry Pi is displayed in **Figures 4-6**.

```

pi@raspberrypi:~/RPiWebService/HW3_Heiman $ python3 services.py -p 7808
* Serving Flask app "services" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:7808/ (Press CTRL+C to quit)

```

Figure 4: Initialization of services.py

```
pi@raspberrypi: ~/RPiWebService/HW3_Heiman
pi@raspberrypi:~ $ curl -u admin:secret "http://127.0.0.1:7808/Canvas?file=tcp_echo_client.py"
curl: (56) Illegal or missing hexadecimal sequence in chunked-encoding
pi@raspberrypi:~ $ cd RPiWebService
pi@raspberrypi:~/RPiWebService $ cd HW3_Assignment
-bash: cd: HW3_Assignment: No such file or directory
pi@raspberrypi:~/RPiWebService $ cd HW3_Heiman
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ ls
__pycache__  ServicesKeys.py  services.py  tcp_echo_client.py
pi@raspberrypi:~/RPiWebService/HW3_Heiman $
```

Figure 5: Using curl to connect to Canvas API and result

```
pi@raspberrypi: ~/RPiWebService/HW3_Heiman
pi@raspberrypi:~ $ curl -u admin:secret "http://127.0.0.1:7808/Canvas?file=tcp_echo_client.py"
curl: (56) Illegal or missing hexadecimal sequence in chunked-encoding
pi@raspberrypi:~ $ cd RPiWebService
pi@raspberrypi:~/RPiWebService $ cd HW3_Assignment
-bash: cd: HW3_Assignment: No such file or directory
pi@raspberrypi:~/RPiWebService $ cd HW3_Heiman
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ ls
__pycache__  ServicesKeys.py  services.py  tcp_echo_client.py
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ curl -u admin:secret "http://127.0.0.1:7808/Marvel?story=8605"
curl: (56) Illegal or missing hexadecimal sequence in chunked-encoding
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ ls
__pycache__  ServicesKeys.py  services.py  Story8605.txt  tcp_echo_client.py
pi@raspberrypi:~/RPiWebService/HW3_Heiman $
```

Figure 6: Using curl to connect to Marvel API and result

The file stored from the Marvel API can be seen below in **Figure 7**.

```
pi@raspberrypi: ~/RPiWebService/HW3_Heiman
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ ls
__pycache__ ServicesKeys.py services.py tcp_echo_client.py
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ curl -u admin:secret "http://127.0.0.1:7808/Marvel?story=8605"
curl: (56) Illegal or missing hexadecimal sequence in chunked-encoding
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ ls
__pycache__ ServicesKeys.py services.py Story8605.txt tcp_echo_client.py
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ cat Story8605.txt
{"code":200,"status":"Ok","copyright":"© 2020 MARVEL","attributionText":"Data provided by Marvel. © 2020 MARVEL","attributionHTML":"<a href=\"http://marvel.com/>Data provided by Marvel. © 2020 MARVEL</a>","etag":"82b815848a2895ca165ad5f7a4894f240583a462","data":{"offset":0,"limit":20,"total":1,"count":1,"results":[{"id":8605,"title":"3 of 5 - Two Plus Two; THE INITIATIVE BANNER","description":"","resourceURI":"http://gateway.marvel.com/v1/public/stories/8605","type":"cover","modified":"2016-05-26T09:57:16-0400","thumbnail":null,"creators":{"available":1,"collectionURI":"http://gateway.marvel.com/v1/public/stories/8605/creators","items":[{"resourceURI":"http://gateway.marvel.com/v1/public/creators/1057","name":"Arthur Suydam","role":"penciller (cover)"},"returned":1],"characters":{"available":4,"collectionURI":"http://gateway.marvel.com/v1/public/stories/8605/characters","items":[{"resourceURI":"http://gateway.marvel.com/v1/public/characters/1009187","name":"Black Panther"}, {"resourceURI":"http://gateway.marvel.com/v1/public/characters/1009356","name":"Human Torch"}, {"resourceURI":"http://gateway.marvel.com/v1/public/characters/1009629","name":"Storm"}, {"resourceURI":"http://gateway.marvel.com/v1/public/characters/1009662","name":"Thing"},"returned":4],"series":{"available":1,"collectionURI":"http://gateway.marvel.com/v1/public/stories/8605/series","items":[{"resourceURI":"http://gateway.marvel.com/v1/public/series/784","name":"Black Panther (2005 - 2008)"},"returned":1],"comics":{"available":1,"collectionURI":"http://gateway.marvel.com/v1/public/stories/8605/comics","items":[{"resourceURI":"http://gateway.marvel.com/v1/public/comics/13438","name":"Black Panther (2005) #28"},"returned":1],"events":{"available":1,"collectionURI":"http://gateway.marvel.com/v1/public/stories/8605/events","items":[{"resourceURI":"http://gateway.marvel.com/v1/public/events/255","name":"Initiative"},"returned":1],"originalIssue":{"resourceURI":"http://gateway.marvel.com/v1/public/comics/13438","name":"Black Panther (2005) #28"}}}}}}}}
pi@raspberrypi:~/RPiWebService/HW3_Heiman $
```

Figure 7: File downloaded from Marvel API

The file downloaded from canvas can be seen below in **Figure 8**.

```
pi@raspberrypi: ~/RPiWebService/HW3_Heiman
pi@raspberrypi:~/RPiWebService/HW3_Heiman $ cat tcp_echo_client.py
#!/usr/bin/env python3

"""
A simple echo client
"""

import socket

host = '192.168.1.108'
port = 50000
size = 1024
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host,port))
s.send(b'Hello, world')
data = s.recv(size)
s.close()
print ('Received:', data)
pi@raspberrypi:~/RPiWebService/HW3_Heiman $
```

Figure 8: File downloaded from Canvas API