Homework Network 1

Group:

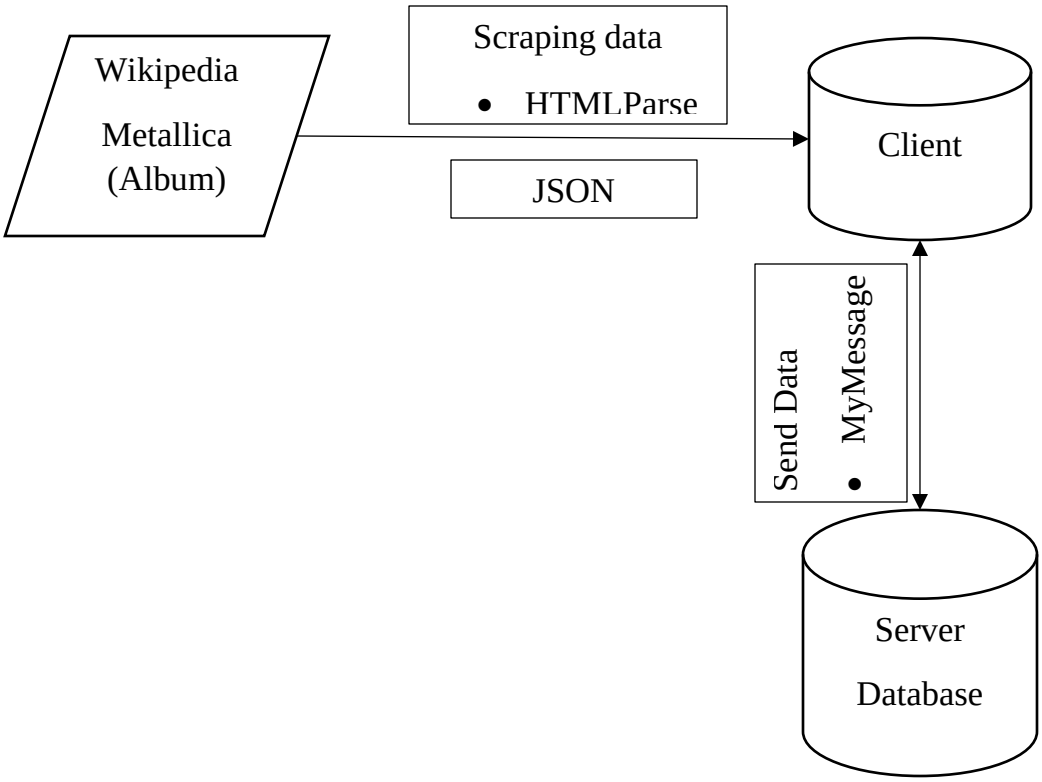
- LE Duong Quoc Thang Email: Duong-Quoc-Thang.Le@eurecom.fr
- VO Huynh Dan Email: Huynh-Dan.Vo@eurecom.fr

1. **Task 1**: Implement a client capable of scraping data from a webpage and to store these data into a small database
   - You will have to use sockets – no existing library will be allowed
   - The database must be implemented as a TCP server – you will have to define your own application protocol!
   - Suggestion: scrape data from tables in Wikipedia pages – anticipate further updates (e.g., date and location of soccer World Cup events …)
   - No group must use the same table as another group – make announcements over the mailing list together with group composition as soon as you have selected a table

1.1. Solution

Python version: Python 2.7

IDE: Pycharm 4.0 (for window PCs) and Python programming on Ubuntu



1.1.1. Design

1.1.2. Explanation

Wiki URL: http://en.wikipedia.org/wiki/Metallica_(album)

Track listing [edit]

All lyrics written by James Hetfield.

No.	Title	Music	Length
1.	"Enter Sandman"	James Hetfield, Lars Ulrich, Kirk Hammett	5:29
2.	"Sad but True"	Hetfield, Ulrich	5:24
3.	"Holier Than Thou"	Hetfield, Ulrich	3:47
4.	"The Unforgiven"	Hetfield, Ulrich, Hammett	6:26
5.	"Wherever I May Roam"	Hetfield, Ulrich	6:42
6.	"Don't Tread on Me"	Hetfield, Ulrich	3:59
7.	"Through the Never"	Hetfield, Ulrich, Hammett	4:01
8.	"Nothing Else Matters"	Hetfield, Ulrich	6:29
9.	"Of Wolf and Man"	Hetfield, Ulrich, Hammett	4:16
10.	"The God That Failed"	Hetfield, Ulrich	5:05
11.	"My Friend of Misery"	Hetfield, Ulrich, Jason Newsted	6:47
12.	"The Struggle Within"	Hetfield, Ulrich	3:51

This is the content of table which we want to scrape from Wikipedia. In order to do that, we use HTMLParser library to scrape all of tables in that webpage and save as datatype = list. (*For more information, you can read the file HTML_table_parser.py*)

```
C:\Python27\python.exe C:/Users/[redacted]/PycharmProjects/MyWorkspcace/client.py
[['Metallica'], [''], ['Studio album by Metallica'], ['Released', 'August 12, 1991 ( 1991-08-12 ) [ 1 ]'], ['Recorded', 'October 6, 19
[['', ['', ['Singles from Metallica'], ['" Enter Sandman " Released: July 29, 1991 ( 1991-07-29 ) [ 2 ]  " The Unforgiven " Released
[['', '"Enter Sandman"   Sorry, your browser either has JavaScript disabled or does not have any supported player. You can download th
[['Professional ratings'], ['Review scores'], ['Source', 'Rating'], ['AllMusic', '[ 49 ]'], ['Chicago Tribune', '[ 50 ]'], ['Robert Ch
[['No.', 'Title', 'Music', 'Length', ''], ['1.', '" Enter Sandman "', 'James Hetfield , Lars Ulrich , Kirk Hammett', '5:29'], ['2.', '
[['Japanese bonus track', ''], ['No.', 'Title', 'Writer(s)', 'Length', ''], ['13.', '" So What? " (originally performed by Anti-Nowher
[['James Hetfield  lead vocals , rhythm guitar , acoustic guitar , 12-string guitar , sitar ; lead guitar on " Nothing Else Matters "
[['Weekly charts [ edit ]', ''], ['Australian Albums ( ARIA ) [ 96 ]', '1'], ['Austrian Albums ( 3 Austria ) [ 97 ]', '5'], ['Belgian
[['', 'Year-end charts [ edit ]', ''], ['US Billboard 200 [ 113 ]', '7']]
[['Chart (1990-1999)', 'Position'], ['US Billboard 200 [ 114 ]', '8']]
[['']]
[['Region', 'Certification', 'Sales/shipments'], ['Argentina ( CAPIF ) [ 115 ]', '7 Platinum', '420,000 x'], ['Australia ( ARIA ) [ 11
[['Preceded by  Unforgettable... with Love by Natalie Cole', 'Billboard 200  number-one album August 31 September 27, 1991', "Succeede
[['], ['], ['James Hetfield  Lars Ulrich  Kirk Hammett  Robert Trujillo    Dave Mustaine  Ron McGovney  Cliff Burton  Jason Newsted'
[['']]

Process finished with exit code 0
```

However, we just need the 5-th table (yellow highlight) which contain the tracks list. Next, we standardize our data in order to remove unwanted information and we use the built-in json library parse the JSON and read through the data.

```
C:\Python27\python.exe C:/Users/[redacted]/PycharmProjects/MyWorkspcace/client.py
[
   {
      "No.": "1.",
      "Music": "James Hetfield , Lars Ulrich , Kirk Hammett",
      "Length": "5:29",
      "Title": " Enter Sandman "
   },
   {
      "No.": "2.",
      "Music": "Hetfield, Ulrich",
      "Length": "5:24",
      "Title": " Sad but True "
   },
   {
      "No.": "3.",
      "Music": "Hetfield, Ulrich",
      "Length": "3:47",
      "Title": "Holier Than Thou"
   },
   {
      "No.": "4.",
      "Music": "Hetfield, Ulrich, Hammett",
      "Length": "6:26",
      "Title": " The Unforgiven "
   },
   {
      "No.": "5.",
```

We use this library because of its following advantages:

- Our data is saved as string which is easily used for sending data to server from client and vice versa.
- It maps directly to some combination of dictionaries and lists. Hence, we can query quickly information we want.

Next step, we build our server which defining our TCP protocol for sending data. We know that TCP/IP is a stream-based protocol, not a message-based protocol. It's not sure that every *send()* call by one peer results in a single *recv()* call by the other peer receiving the exact data sent. It might receive the data splitting across multiple *recv()* calls due to packet fragmentation. So, we define our own message-based protocol on top of TCP in order to differentiate message boundaries. Therefore, in order to read a message, we continue to call *recv()* until you've read an entire message or an error occurs. One simple way of sending a message is to prefix each message with its

length. Then, to read a message, firstly, read the length, after that, read that many bytes. (*For more information, you can read the file MyMessage.py*)

Next step, after receiving data from the client, server will store it into file text "*Track_List.txt*"

Finally, to check whether our data loses information during sending from client to server or not, the server will send the data it received to the client. In the client, it will be compare with the original data scraping from wiki.

1.1.3. Experiment and Result

Start server before running client

From client

```
Run: server client
C:\Python27\python.exe C:/Users/[redacted]/PycharmProjects/MyWorkspcace/client.py
...Client connected to server...

...Client sent the content of table from Server...
...Client received data from Server...
...Comparing data sent to server with received from server...
     YES, they are the same
...Connection to server is closed...

Process finished with exit code 0
```

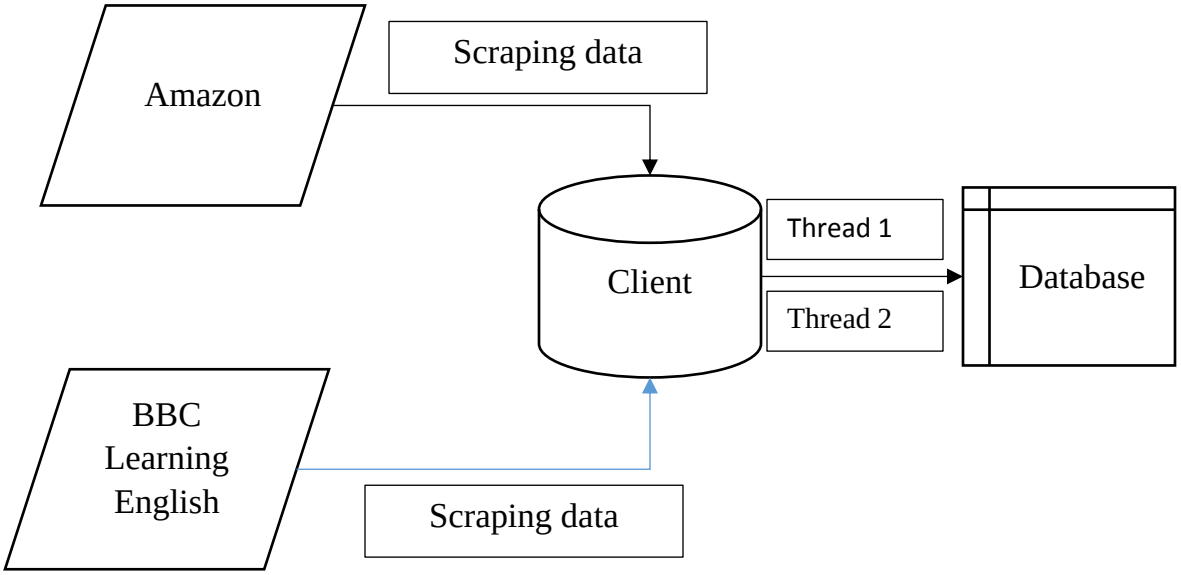
From server

```
Run: server client
C:\Python27\python.exe C:/Users/[redacted]/PycharmProjects/MyWorkspcace/server.py
...Socket Server bind complete...
...Chat server started on port 8888...
...Server connected to Client... ('127.0.0.1', 51091)
...Server received the content of table from Client...
...Server send data, which already is received from client, back to the client...

Stored data
...Connection to client is closed...

Process finished with exit code 0
```

2. **Task 2:** Use a library like dryscrape for scraping more complex webpages.
   - https://dryscrape.readthedocs.org/en/latest/
   - You should select two different sites
   - Run two instances of your scraping engine using threads, while scraping two different pages (e.g. scrape prices from merchant websites)
   - Store your results in a database now shared between threads (watch out for potential synchronization issues!)

2.1. Solution:

2.1.1. Design



2.1.2. Explanation

To achieve the objective of 2nd task. We programmed on Ubuntu using python built-in and dryscrape library for collecting data from sites

- There are two websites chosen that are http://www.amazon.com/s/ref=nb_sb_noss_2?url=search-alias%3Daps&field-keywords=iphone (Search for iphone on Amazon)

  http://www.bbc.co.uk/learningenglish/english/features/6-minute-english (BBC Leaning English - 6 minutes program)

- We used multithreads, two primary functions crawling_BBC() and crawling_Amazon().
  - crawling_BBC() uses dryscrape to collect data, then it parses data and write down to the database (database.txt file)
  - crawling_Amazon() similarly does the same thing as crawling_BBC()
- We using firefox browser to open those websites and + f12 to see html file for choosing good tables and divs in order to have better extract.
- To prevent overlapping data, we decided to use pauses between two threads. In particular, by using command **time.sleep(2)**. Furthermore, after we run every function, we collect data in a string and write the whole string to the database.

2.1.3. Experiment and Result

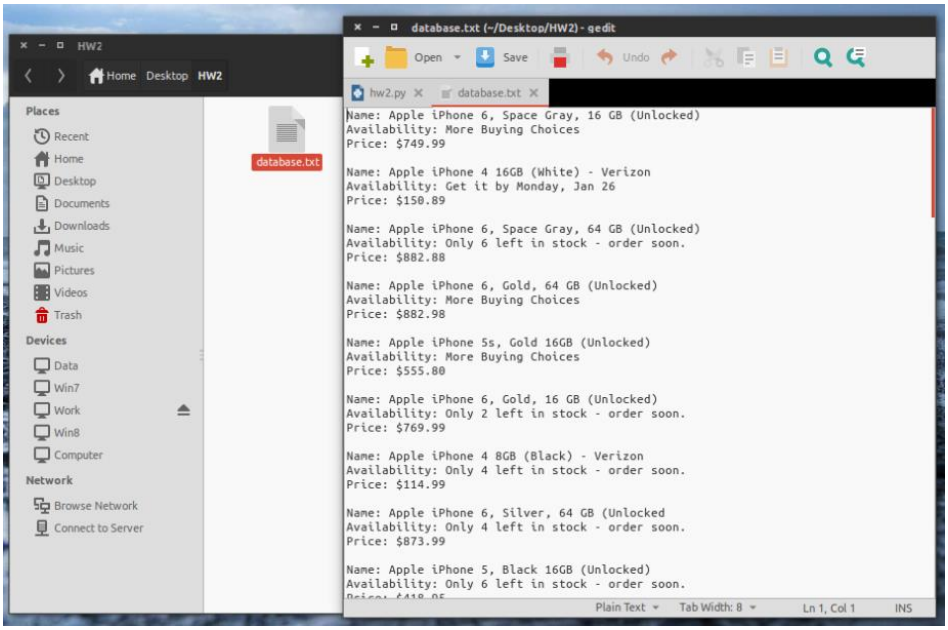
To run the code for task2 we could follow this instruction:

- Install dryscrape library first here
- Then use Ctr + Alt + T to open terminal
- Jump to HW2 directory by command **cd HW2**
- Type **python hw2.py**

```
danvo@danvo: ~/Desktop/HW2
danvo@danvo:~$ cd HW2
bash: cd: HW2: No such file or directory
danvo@danvo:~$ cd Desktop/HW2
danvo@danvo:~/Desktop/HW2$ python hw2.py
```

After running we will see the result from the terminal. The data will be stored in the text file **database.txt**. every record is distinct from others by newline

```
danvo@danvo: ~/Desktop/HW2
danvo@danvo:~/Desktop/HW2$ python hw2.py
Logging in to Amazon
Logging in to BBC Learning English
Collecting data from Amazon ...
Collecting data from BBC ...
Writing data down to database ...
Name: Apple iPhone 6, Space Gray, 16 GB (Unlocked)
Availability: More Buying Choices
Price: $749.99

Name: Apple iPhone 4 16GB (White) - Verizon
Availability: Get it by Monday, Jan 26
Price: $150.89

Name: Apple iPhone 6, Space Gray, 64 GB (Unlocked)
Availability: Only 6 left in stock - order soon.
Price: $882.88

Name: Apple iPhone 6, Gold, 64 GB (Unlocked)
Availability: More Buying Choices
Price: $882.98

Name: Apple iPhone 5s, Gold 16GB (Unlocked)
Availability: More Buying Choices
```



We can also uncomment two lines above to make sure we connect to two correct websites (There will be two files Amazon.png and BBCEnglish.png)

```
        print "Logging in to Amazon"
        sess.visit('/s/ref=nb_sb_noss_2?url=search-alias%3Daps&field-
keywords=iphone')
        #sess.render('Amazon.png')
        product = sess.xpath('//div[@class="a-fixed-left-grid-inner"]')


        if product == None:
                print "Cannot connect to Amazon, check your internet connection"
        if product != None:
                data1=""
                print "Collecting data from Amazon ..."
                for i in product:
                        name=i.at_xpath('div/div/a/h2[@class]')
                        #print "Name:", name.text()
                        data1 = data1 + "Name: "+ name.text() + "\n"
                        av=i.at_xpath('div/div/div/div/div[@class]')
                        #print "Availablity:", av.text()
                        data1 = data1 + "Availability: " + av.text() + "\n"
                        price=i.at_xpath('div/div/div/div/a[@class]')
                        #print "Price:", price.text()
                        data1 = data1 + "Price: " + price.text() + "\n\n"
                print "Writing data down to database ... "
                print data1
                with open ('database.txt', 'a') as f1:
                        f1.write(data1)


def crawling_BBC():
        # set up a web scraping session
        sess = dryscrape.Session(base_url = 'http://www.bbc.co.uk')
        # we don't need images
        sess.set_attribute('auto_load_images', False)
        print "Logging in to BBC Learning English"
        sess.visit('/learningenglish/english/features/6-minute-english')
        #sess.render('BBCEnglish.png')
        course = sess.xpath('//li[@class="course-content-item active"]')
        if course == None:
```

--END--