Introduction to Web Science

Assignment 4

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: November 23, 2016, 10:00 a.m.
Tutorial on: November 25, 2016, 12:00 p.m.

In this assignment we cover two topics: 1) **HTTP** & 2) **Web Content**

For all the assignment questions that require you to write code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

Group Name : Yankee
Candidates :

1. Sabin Bhattarai - 216203590
   sbhattarai@uni-koblenz.de
2. Biplov K.C. - 216203865
   biplov@uni-koblenz.de
3. Syed Salman Ali. - 216203923
   salmanali@uni-koblenz.de

1 Implementing a simplified HTTP GET Request (15 Points)

The goal of this exercise is to review the hyptertext transfer protocol and gain a better understanding of how it works.

Your task is to use the python programming language to create an HTTP client (http-client.py) that takes a URL as a command line argument and is able to download an arbitrary file from the World Wide Web and store it on your hard drive (in the same directory as your python code is running). The program should also print out the complete HTTP header of the response and store the header in a seperated file.

Your programm should only use the socket library so that you can open a TCP socket and and sys library to do command line parsing. You can either use urlparse lib or your code from assignment 3 in order to process the url which should be retrieved.

Your programm should be able to sucessfully download at least the following files:

1. http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-science
2. http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbeitet_03.jpg

Use of libraries like `httplib`, `urllib`, etc are not allowed in this task.

1.1 Hints:

There will be quite some challenges in order to finnish the task

- Your program only has to be able to process HTTP-responses with status 200 OK.
- Make sure you receive the full response from your TCP socket. (create a function handling this task)
- Sperated the HTTP header from the body (again create a function to do this)
- If a binary file is requested make sure it is not stored in a corrupted way

1.2 Example

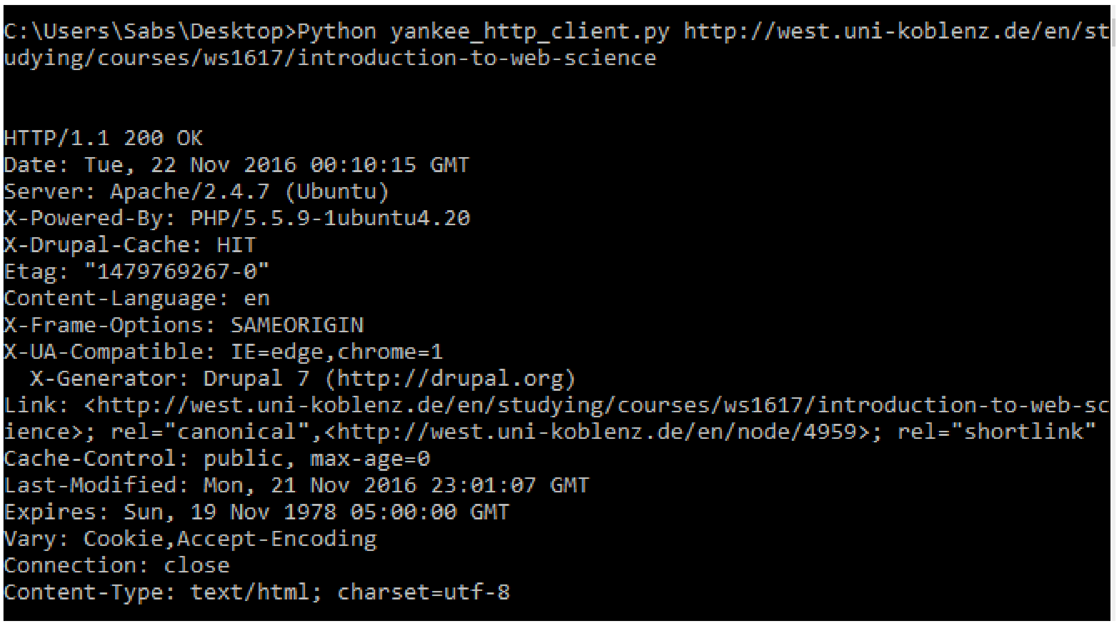
```
python httpclient.py http://west.uni-koblenz.de/index.php

HTTP/1.1 200 OK
Date: Wed, 16 Nov 2016 13:19:19 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
X-Drupal-Cache: HIT
Etag: "1479302344-0"
Content-Language: de
```

```
X-Frame-Options: SAMEORIGIN
X-UA-Compatible: IE=edge,chrome=1
X-Generator: Drupal 7 (http://drupal.org)
Link: <http://west.uni-koblenz.de/de>; rel="canonical",<http://west.uni-koblenz.de
Cache-Control: public, max-age=0
Last-Modified: Wed, 16 Nov 2016 13:19:04 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
Connection: close
Content-Type: text/html; charset=utf-8
```

The header will be printed and stored in index.php.header. The retrieved html document will be stored in index.php

Answer 1



```
C:\Users\Sabs\Desktop>Python yankee_http_client.py http://west.uni-koblenz.de/en/st
udying/courses/ws1617/introduction-to-web-science


HTTP/1.1 200 OK
Date: Tue, 22 Nov 2016 00:10:15 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
X-Drupal-Cache: HIT
Etag: "1479769267-0"
Content-Language: en
X-Frame-Options: SAMEORIGIN
X-UA-Compatible: IE=edge,chrome=1
  X-Generator: Drupal 7 (http://drupal.org)
Link: <http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-sc
ience>; rel="canonical",<http://west.uni-koblenz.de/en/node/4959>; rel="shortlink"
Cache-Control: public, max-age=0
Last-Modified: Mon, 21 Nov 2016 23:01:07 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
Connection: close
Content-Type: text/html; charset=utf-8
```

Figure 1: Obtained output showing the printed header file

```
import socket
import sys
from urllib.parse import urlparse

ENCODING = 'utf-8'

#takes command line arguments and performs downloading files
def mainFunction(commandArguments , totalArg):
    if (totalArg > 1):
        downloadArbitaryFileWEB(commandArguments[1])
    else:
        print ("\n Invaild URL -- Default URL shown below will be used : " +
        "\n http://west.uni-koblenz.de/en/studying/courses/ws1617/" +
        "introduction-to-web-science \n\n")
        downloadArbitaryFileWEB("http://west.uni-koblenz.de/en/studying/" +
        "courses/ws1617/introduction-to-web-science")


#Connects to the server and receives response. writes to a file
def downloadArbitaryFileWEB(url , content_typeLength=14):
    #Connecting to Server and obtaining responses in Byte format
    responseInBytes = _clientConnectAndResponse(url)

    headerBytes = responseInBytes[:responseInBytes.find(b'\r\n\r\n')]
    remainingBytes = responseInBytes[responseInBytes.find(b'\r\n\r\n'):]\
                                    .strip(b'\r\n\r\n')

    #Here we check if HTTP responses with 200 OK
    if (headerBytes.find(b'\r\n')) > 0 :
        if headerBytes[:headerBytes.find(b'\r\n')].decode(ENCODING).lower()\
                        != "http/1.1 200 ok":
            print ("Cannot determine if HTTP 200 OK. Further Processing halts")
            return

    #Display Header
    print('\n')
    print (headerBytes.decode(ENCODING))

    #checking if the URL is for an image
    isImage = _checkIfImage(headerBytes , content_typeLength)
    headerFile = open('index.php.header' , 'w', newline='\n')
    headerFile.write(headerBytes.decode(ENCODING))

    if isImage:
        imageToFile = open(urlparse(url).path.split('/')[-1], 'wb')
        imageToFile.write(remainingBytes)
        imageToFile.close()
    else:
        htmlToFile = open('index.php', 'wb')
        htmlToFile.write(remainingBytes)
        htmlToFile.close()


#Helper function that handles socket connection and also gives response in bytes
def _clientConnectAndResponse(url, timeout=10, receive_buffer=8096):
    parsed = urlparse(url)
    try:
        host, port = parsed.netloc.split(':')
    except ValueError:
        host, port = parsed.netloc, 80

    clientSocket = socket.create_connection((host, port), timeout)

    method  = 'GET %s HTTP/1.0\r\n\r\n' % parsed.path
    clientSocket.sendall(bytes(method, ENCODING))

    response = [clientSocket.recv(receive_buffer)]
    while response[-1]:
        response.append(clientSocket.recv(receive_buffer))

    responseInBytes = b''.join(r for r in response)
    return responseInBytes

#Takes headerBytes and checks if response from server was an image file or not
def _checkIfImage (headerBytes, content_typeLength):
    if (headerBytes.find(b'Content-Type:') >= 0):
        headerBytesAfterContentType = headerBytes[headerBytes.find\
                                                  (b'Content-Type:'):]
    else:
        print ("DOESNOT HAVE A CONTENT TYPE")
        raise ValueError('No Content Type in header File')
    if (headerBytesAfterContentType.find(b';') >= 0):
        fileFormat = headerBytesAfterContentType[:headerBytesAfterContentType\
                                                 .find(b';')]
    else:
        fileFormat = headerBytesAfterContentType
    #print (fileFormat.decode(ENCODING))
    fileFormatStrip = fileFormat.decode(ENCODING)[content_typeLength:]

    if fileFormatStrip.lower().find('image/') >= 0:
        return True
    return False

if __name__ == "__main__":
     #Main Function call alongside command line Argument Handling
     totalArg = len(sys.argv)
     commandArguments = sys.argv # Get the arguments list
     mainFunction(commandArguments , totalArg)
```

2 Download Everything (15 Points)

If you have successfully managed to solve the previous exercise you are able to download a web page from any url. Unfortionately in order to successfully render that very webpage the browser might need to download all the included images

In this exercise you should create a python file (downloadEverything.py) which takes two arguments. The first argument should be a name of a locally stored html file. The second argument is the url from which this file was downloaded.

Your program should

1. be able to find a list of urls the images that need to be downloaded for successful rendering the html file.
2. print the list of URLs to the console.
3. call the program from task 1 (or if you couldn't complete task 1 you can call wget or use any python lib to fulfill the http request) to download all the necessary images and store them on your hard drive.

To finnish the task you are allowed to use the 're' library for regualar expressions and everything that you have been allowed to use in task 1.

2.1 Hints

1. If you couldn't finnish the last task you can simulate the relevant behavior by using the program wget which is available in almost any UNIX shell.
2. Some files mentioned in the html file might use relative or absolut paths and not fully qualified urls. Those should be fixed to the correct full urls.
3. In case you run problems with constructing urls from relative or absult file paths you can always check with your web browser how the url is dereferenced.

Answer 2

```
import re
import sys
from urllib.parse import urlparse
from urllib.parse import urljoin
from yankee_http_client import downloadArbitaryFileWEB

ENCODING = 'utf-8'

def mainDownloadEverythingFunction(commandArguments , totalArg):
    if (totalArg > 1):
        parseHTML(commandArguments[1] , commandArguments[2])
    else:
        print ("Please give valid arguments")
        return

def parseHTML(filename , url, content_typeLength=14):
    with open(filename , 'r') as html:
        content = html.read()
        pat = re.compile (r'<img [^>]*src="([^"]+)')
        # Here we also check\ from <link href and images inside them
        pat2 = re.compile (r'<link [^>]*href="([^"]+)')

        moreimages = pat2.findall(content)
        moreimagesFiltered = list(filter(lambda x:\
                                        x.find('.ico') >= 0  or\
                                        x.find('.png') >= 0  or\
                                        x.find('.jpeg') >= 0  , moreimages))

        img = pat.findall(content)

    allImagesURL = moreimagesFiltered + img
    toAppendRelativeImageURL = _getPartToAppendToRelativeImageURL(url)

    for imageurl in allImagesURL:
        if(imageurl.find('://') > 0):
            downloadArbitaryFileWEB(imageurl)
            #perform download for that url
        else:
            link = urljoin(toAppendRelativeImageURL, imageurl)
            downloadArbitaryFileWEB(link) #perform download for that url


# Helper function that returns the front part of the url that is to be appened
#to relative urls for valid image construction
def _getPartToAppendToRelativeImageURL(url):
    parsed = urlparse(url)
    try:
            host, port = parsed.netloc.split(':')
        except ValueError:
            host = parsed.netloc
        return parsed.scheme +'://'+ host


#Main Function call alongside command line Argument Handling
totalArg = len(sys.argv)
commandArguments = sys.argv # Get the arguments list
mainDownloadEverythingFunction(commandArguments , totalArg)
```

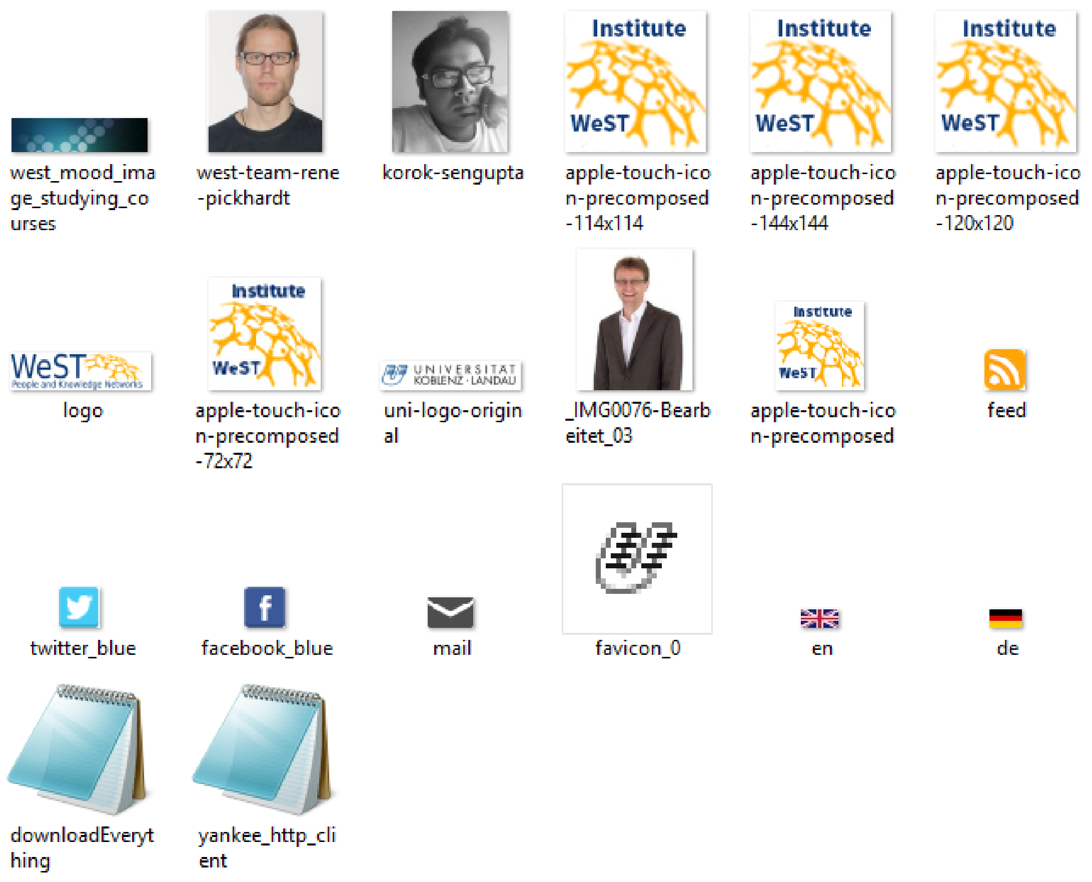

west_mood_image_studying_courses
west-team-rene-pickhardt
korok-sengupta
apple-touch-icon-precomposed-114x114
apple-touch-icon-precomposed-144x144
apple-touch-icon-precomposed-120x120
logo
apple-touch-icon-precomposed-72x72
uni-logo-original
_IMG0076-Bearbeitet_03
apple-touch-icon-precomposed
feed
twitter_blue
facebook_blue
mail
favicon_0
en
de
downloadEverything
yankee_http_client

Figure 2: Screen-shot of folder containing all the downloaded images

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment4/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of ”Web Information Retrieval”.

LaTeX

Currently the code can only be build using LuaLaTeX, so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the LaTeXengine to `LuaLaTeX`.