

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.

Team Name: India

Group Member : Jalpa Patel, Amani Gaddamedi, Jasvinder Kaur

1 Implementing a simplified HTTP GET Request (15 Points)

The goal of this exercise is to review the hypertext 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 (`httpclient.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 separated 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

```
1: python httpclient.py http://west.uni-koblenz.de/index.php
2:
3: HTTP/1.1 200 OK
4: Date: Wed, 16 Nov 2016 13:19:19 GMT
5: Server: Apache/2.4.7 (Ubuntu)
6: X-Powered-By: PHP/5.5.9-1ubuntu4.20
7: X-Drupal-Cache: HIT
8: Etag: "1479302344-0"
9: Content-Language: de
```

```

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

```

1 import socket
2 from urllib.parse import urlparse
3 import sys
4 __author__ = "jasvinder"
5
6
7 # creates an INET, STREAMING socket
8 # input:
9 # url:String - takes the valid url to make tcp connection
10 # port: Integer - take the port number to connect
11 # output:
12 # TCPSocket - returns a tcp socket object for further communication.
13 def tcp_connection(url, port):
14     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     clientSocket.connect((url, port))
16     return clientSocket
17
18 # Parses the URL into six components.i.e scheme, network location, path, parameters, query and fragment
19 # input:
20 # url name:String - takes the valid url passed
21 # output:
22 # String - parsed Url into 6 components
23 def parse_url(url_name):
24     return urlparse(url_name)
25
26 # User request for sending user input URL as HTTP request ending with two empty lines
27 # input:
28 # url name:String - takes the valid url passed
29 # method_type:String - takes Http method type,i.e generally, POST or GET
30 # output:
31 # get_request:String - returns the valid Http Request according to method type

```

Figure 1: Code

2.

```

32 def encode_http_request(path, method_type):
33     protocol_type = 'HTTP/1.0'
34     get_request = ''.join([method_type, ' ', path, ' ', protocol_type, '\r', '\n', '\r', '\n'])
35     return get_request
36
37 # Sends the passed http_request request over passed tcp socket object.
38 # input:
39 # tcp_connection:TCP Socket object - takes the object of the TCP connection established
40 # http_request:String - takes the valid Http Request according to method type i.e GET or POST method
41 # output:
42 # Bytes: returns the Http request made in bytes to send it further for communication
43 def send_http_request(tcp_connection, http_request):
44     return tcp_connection.send(str.encode(http_request))
45
46 # Reads the Http header of the response from tcp_connection socket.
47 # input:
48 # tcp_connection:TCP Socket object - takes the object of the TCP connection established
49 # output:
50 # Http headers: String - gives the HTTP header of the response
51 def read_http_header(tcp_connection):
52     http_headers = ['HTTP/1.1 200 OK']# initialising headers too 22 as it has already been read to determine whether its 200 or not.
53     while(True):
54         response = tcp_connection.recv(1).decode('utf-8')
55         if(response == '\r'):
56             http_headers.append(response)
57         response = tcp_connection.recv(3).decode('utf-8')
58         if(response == '\n\r\n'):
59             http_headers.append(response)
60             break
61     else:
62         http_headers.append(response)

```

Figure 2: Code

3.

```

64     else:
65         http_headers.append(response)
66     return ''.join(http_headers)
67
68 # Checks whether the response is 200 or not from the tcp socket.
69 # input:
70 # tcp_connection: TCP Socket object - takes the object of the TCP connection established
71 # output:
72 # Boolean: True for 200 False for all other http status code.
73 def http_status_200(tcp_connection):
74     http_status = tcp_connection.recv(15).decode('utf-8')
75     print(http_status)
76     if http_status.startswith('HTTP/1.1 200 OK'):
77         return True
78     return False
79
80 # Reads the body of the Http response as binary
81 # input:
82 # tcp_connection: TCP Socket object - takes the object of the TCP connection established
83 # output:
84 # list - gives the body of the Http response, as binary data
85 def read_http_body_as_binary(tcp_connection):
86     http_body = []
87     response = tcp_connection.recv(1024)
88     while(response):
89         http_body.append(response)
90         response = tcp_connection.recv(1024)
91     return http_body
92
93 # Decodes http header
94 # input:
95 # http_header: Http header string.

```

Figure 3: Code

4.

```

95 # http_header: Http header string.
96 # output:
97 # list - list of http header strings.
98 def decode_http_header(http_header):
99     return http_header.strip().split('\r\n')
100
101 # Reads the decoded http header
102 # input:
103 # decoded_http_header: list of decoded http header
104 # output:
105 # String: tells whether type is text in binary
106 def find_content_type(decoded_http_header):
107     #print(decoded_http_header)
108     for value in decoded_http_header:
109         if value.startswith('Content-Type:'):
110             if value.startswith('Content-Type: text/html'):
111                 return 'text'
112             else:
113                 return 'binary'
114
115 # Reads the string and the file name and saves the string on the file.
116 # input:
117 # string_data: to be written: String - String of text to be written on the file
118 # file_name: String- file name.
119 # output:
120 # None
121 def save_text_data(string_data_to_be_written, file_name):
122     with open(file_name, 'w') as file:
123         file.write(string_data_to_be_written)
124
125 # Reads the string and the file name and saves the string on the file.
126 # input:

```

Figure 4: Code

5.

```

139 # Reads the string and the file name and saves the string on the file.
140 # input:
141 # url_name: String - Url of the resource to be downloaded.
142 # saveheader: Boolean- default is True. This is a flag for stating whether the header or not.
143 # output:
144 # None
145 def call_http_server(url_name, saveheader = True):
146     parsed_url = parse.urlparse(url_name)
147     tcp_socket = tcp_connection(parsed_url.netloc, 80)
148     http_request = encode_http_request(parsed_url.path, 'GET')
149     send_http_request(tcp_socket, http_request)
150     print(url_name)
151     if http_status_200(tcp_socket):
152         http_header = read_http_header(tcp_socket)
153         http_data = read_http_body_as_binary(tcp_socket)
154         decoded_message = decode_http_header(http_header)
155         file_name = parsed_url.path.split('/')[1]
156         if saveheader:
157             header_file_name = file_name + '.header'
158             save_text_data(http_header, header_file_name)
159             if (find_content_type(decoded_message) == 'text'):
160                 text_file_name = file_name + '.html'
161                 save_binary_data(http_data, text_file_name)
162             elif (find_content_type(decoded_message) == 'binary'):
163                 save_binary_data(http_data, file_name)
164         else:
165             print('http status not 200')
166     tcp_socket.close()
167
168 if __name__ == '__main__':
169     call_http_server(sys.argv[1])
170

```

Figure 5: Code

6.

```
Applications Places
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
File Edit View Search Terminal Tabs Help
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ python http_client.py "http://west.uni-koblenz.de/en/studying/course/s/ws1617/introduction-to-web-science"
http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-science
HTTP/1.1 200 OK
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ ls
downloadEverything.py India_assignment4_Q2client.py introduction-to-web-science.html untitled.pdf
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ cat introduction-to-web-science.header
HTTP/1.1 200 OK
Date: Tue, 22 Nov 2016 23:14:11 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
X-Drupal-Cache: HIT
Etag: "1479855724-0"
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-science>; rel="canonical",<http://west.uni-koblenz.de/en/node/4959>; rel=shortlink
Cache-Control: public, max-age=0
Last-Modified: Tue, 22 Nov 2016 23:02:04 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
Connection: close
Content-Type: text/html; charset=utf-8
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$
```

Figure 6: Output

7.

```
Applications Places
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
File Edit View Search Terminal Tabs Help
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ python http_client.py "http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbettel_03.jpg"
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbettel_03.jpg
HTTP/1.1 200 OK
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ ls
downloadEverything.py _IMG0076-Bearbettel_03.jpg India_assignment4_Q2client.py introduction-to-web-science.html untitled.pdf
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ cat _IMG0076-Bearbettel_03.jpg.header
HTTP/1.1 200 OK
Date: Tue, 22 Nov 2016 23:27:42 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Fri, 07 Aug 2015 10:33:19 GMT
Etag: "1f7a516eb11f3c0"
Accept-Ranges: bytes
Content-Length: 4056
Connection: close
Content-Type: image/jpeg
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$
```

Figure 7: Output

8.

```
Applications Places
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
File Edit View Search Terminal Tabs Help
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ python http_client.py "http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbettel_tetzgfffffffffffffffffffs.jpg"
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbettel_tetzgfffffffffffffffffffs.jpg
HTTP/1.1 403 Fo
http status not 200
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$
```

Figure 8: Output

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. Unfortunately 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 finish the task you are allowed to use the 're' library for regular expressions and everything that you have been allowed to use in task 1.

2.1 Hints

1. If you couldn't finish 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 absolute 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 absolute file paths you can always check with your web browser how the url is dereferenced.

Answer:

Code:

1.

```

1 import re
2 import sys
3 import http_client
4
5 # Reads the file saved at the file name and returns the data.
6 # input:
7 #   file_name:String - name of the file to be read.
8 # output:
9 #   String: Data from the file.
10 def read_data(file_name):
11     with open(file_name, 'r') as f:
12         data = f.read()
13     return data
14
15 # finds the src from the img tag in full html string..
16 # input:
17 #   html:String - HTML data as strings
18 # output:
19 #   List: List of urls from image tags src.
20 def match_img_src(html):
21     img_pattern = '<img.*?src=([^\s]+)'
22     return re.findall(img_pattern, html)
23
24
25 # Converts the relative urls from url list to complete http urls.
26 # input:
27 #   url_list:List - URL list as list of string.
28 #   url:String - url of main page from where html was downloaded.
29 # output:
30 #   List - List of complete http url.
31

```

Figure 9: Code

2.

```

32 # List - List of complete http url.
33 def make_absolute_url(url_list, url):
34     downloaded_domain = http_client.parse_url(url).netloc
35     complete_url = []
36     for url in url_list:
37         parsed_url = http_client.parse_url(url)
38         if(parsed_url.netloc):
39             complete_url.append(url)
40         else:
41             complete_url.append('http://' + downloaded_domain + url )
42     return complete_url
43
44 # Reads the html files extracts all html links to image and download it and saves it to the disc and prints all the URLs at the console.
45 # input:
46 #   file_name - Name of the file in which html page is saved-
47 #   url:String - Url of the resource from where main file was downloaded.
48 # output:
49 #   None
50 def download_urls(file_name, url):
51     html_data = read_data(file_name)
52     url_list = match_img_src(html_data)
53     full_url_list = make_absolute_url(url_list, url)
54     print('List of URLs extracted from downloaded html')
55     print(full_url_list)
56     for url in full_url_list:
57         print('Downloading url')
58         http_client.call_http_server(url, False)
59
60 if __name__ == '__main__':
61     download_urls(sys.argv[1], sys.argv[2])
62

```

Figure 10: Code

Output:

1.

```

Applications - Jass
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4
jass@jass-Compaq-CQ58-Notebook-PC: ~/Documents/jass/WebScience/India/assignment4$ python downloadEverything.py "/home/jass/WebScience/India/assignment4/Introduction-to-web-science.html" "http://west.uni-koblenz.de/en/studying/courses/ws1617/Introduction-to-web-science"
List of URLs extracted from downloaded html
[ 'http://west.uni-koblenz.de/sites/all/themes/westonega/images/uni-logo-original.png', 'http://west.uni-koblenz.de/sites/all/themes/westonega/images/nail.png', 'http://west.uni-koblenz.de/sites/all/themes/westonega/images/facebook_blue.png', 'http://west.uni-koblenz.de/sites/default/files/cns/feed.png', 'http://west.uni-koblenz.de/sites/all/themes/westonega/images/twitter_blue.png', 'http://west.uni-koblenz.de/sites/all/themes/westonega/logo.png', 'http://west.uni-koblenz.de/sites/default/files/styles/medium/public/cns/header/west_image_studying_courses.jpg?itok=ezou7K', 'http://west.uni-koblenz.de/sites/all/modules/languageicons/flags/en.png', 'http://west.uni-koblenz.de/sites/all/modules/languageicons/flags/de.png', 'http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IM0076-Bearbeitet_03.jpg?itok=nvWQdG19', 'http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/aboutus/team/persons/west-team-rene-pickardt.png?itok=ly9pC1t7', 'http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/aboutus/team/persons/korok-sengupta.jpg?itok=040800yU' ]
downloading url
http://west.uni-koblenz.de/sites/all/themes/westonega/images/uni-logo-original.png
HTTP/1.1 200 OK
downloading url
http://west.uni-koblenz.de/sites/all/themes/westonega/images/nail.png
HTTP/1.1 200 OK
downloading url
http://west.uni-koblenz.de/sites/all/themes/westonega/images/facebook_blue.png
HTTP/1.1 200 OK
downloading url
http://west.uni-koblenz.de/sites/default/files/cns/feed.png
HTTP/1.1 200 OK
downloading url
http://west.uni-koblenz.de/sites/all/themes/westonega/images/twitter_blue.png
HTTP/1.1 200 OK
downloading url
http://west.uni-koblenz.de/sites/all/themes/westonega/logo.png
HTTP/1.1 200 OK
downloading url

```

Figure 11: Output

The screenshot displays a web browser window titled "jess@jss-Compaq-CQ58-Notebook-PC - WebScience/India/assignment4". The browser's address bar shows the URL "http://www.west.uni-koblenz.de/sites/all/themes/westomega/images/twitter_blue.png". The main content area of the browser displays a series of HTTP requests and responses, formatted as follows:

```
download url  
http://west.uni-koblenz.de/sites/all/themes/westomega/logo.png  
HTTP/1.1 200 OK
```

```
download url  
http://west.uni-koblenz.de/sites/default/files/styles/headerblid/public/cns/header/west_mood_image_studying_courses.jpg?tok=ezBuY_7k  
HTTP/1.1 200 OK
```

```
download url  
http://west.uni-koblenz.de/sites/all/modules/languagetools/flags/en.png  
HTTP/1.1 200 OK
```

```
download url  
http://west.uni-koblenz.de/sites/all/modules/languagetools/flags/de.png  
HTTP/1.1 200 OK
```

```
download url  
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbeitet_03.jpg?tok=nYwQC19  
HTTP/1.1 200 OK
```

```
download url  
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/aboutus/team/persons/west-team-rene-pickhardt.png?tok=ly9PG17  
HTTP/1.1 200 OK
```

```
download url  
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/aboutus/team/persons/korok.png?tok=I04080yU  
HTTP/1.1 200 OK
```

The taskbar at the bottom of the screen shows several open applications, including "Applications", "Places", "jess@jss-Compaq-CQ58-Notebook-PC", "Telegram (1)", "Pictures", and "untitled.pdf". A small notification bubble is visible in the bottom right corner, indicating a message from "WebScience" with the text "jaka ok!".

9

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 \LaTeX engine to LuaLaTeX.