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: uniform
Group members: Pradip Giri, Jalak Arvind Kumar Pansuriya, Madhu Rakhal Magar

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

```
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
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:

Listing 1 helper.py

```
1: # pylint: disable-msg=C0103
2: """
3: Helper utility librabry for httpclient.py
4:
5: """
6:
7:
8: def save_file(data, file_name):
9:     """ creates file and write data into it"""
10:     with open(file_name, 'w+') as f:
11:         f.write(data)
12:     print('{} successfully written'.format(file_name))
13:
14:
15: def is_scheme_present(url):
16:     """
17:     return true if and only if :// is present in url
18:     """
19:     return url.find("://") > 0
20:
21:
22: def get_port_number(domain):
23:     """
24:     returns the port number if available in given domain info
25:     otherwise returns default port 80
26:     """
27:     port = None
28:     if domain.find(":"):
29:         try:
30:             port = int(domain.split(":")[1])
```

```
31:         except (ValueError, IndexError):
32:             port = 80
33:     return port
34:
35:
36: def url_with_protocal(url):
37:     """ adds http:// if not present
38:     """
39:     if not is_scheme_present(url):
40:         return 'http://' + url
41:     return url
42:
43:
44: def url_without_port(protocal):
45:     """ returns protocal without port number"""
46:     return protocal.split(":")[0]
47:
48:
49: def is_empty_string(string):
50:     """ returns True iff string is empty"""
51:     return string == ''
52:
53:
54: def is_status_sucess(status_text):
55:     """ returns whether 200 OK is present or not"""
56:     infos = status_text.split("\n")[0]
57:     return '200' in infos
```

Listing 2 httpclient.py

```
 1: #!/urs/bin/python
 2: # pylint: disable-msg=C0103
 3: """ Simple http client which downloads resources from url"""
 4: import sys
 5: import socket
 6: from urllib.parse import urlparse
 7: from helper import get_port_number
 8: from helper import url_with_protocal
 9: from helper import is_empty_string
10: from helper import save_file
11:
12:
13: def save_downloaded_file(data, file_name, isHeader=False):
14:     """
15:     creates required file name to save
16:     if file name is empty string then index.html is used as default file name
17:     """
18:     new_file_name = None
19:     if not file_name:
20:         file_name = "index.html"
```

```
21:     if isHeader:
22:         new_file_name = file_name + '.header'
23:         print("Header Information \n")
24:         print(data)
25:         print('\n')
26:     else:
27:         new_file_name = file_name
28:     save_file(data, new_file_name)
29:
30:
31: def handle_file_request(s, data, file_name):
32:     """ handle downloading file from the server"""
33:     with open(file_name, 'wb') as f:
34:         f.write(data)
35:         while True:
36:             data = s.recv(1024)
37:             if not data:
38:                 break
39:             f.write(data)
40:         print('{} successfully downloaded'.format(file_name))
41:
42:
43: def handle_http_request(s, data, file_name):
44:     """ download html files"""
45:     full_response = data.decode()
46:     while True:
47:         data = s.recv(1024)
48:         result = data.decode()
49:         if not data:
50:             break
51:         full_response += result
52:     save_downloaded_file(full_response, file_name.split('/')[-1])
53:
54:
55: def handle_receving_data(s, file_name):
56:     """ handles the receiving of the data from socket server
57:     only processed futher if the server sends back 200 OK other wise programe
58:     will terminate.
59:     """
60:     file_name = file_name.split('/')[-1]
61:
62:     data = s.recv(1024)
63:     header, result = data.split(b'\r\n\r\n')
64:     is_status_200_ok = b'200 OK' in header
65:     is_file_request = b'Content-Type: image/'
66:
67:     if not is_status_200_ok:
68:         print("Got response other than 200 OK")
69:         return False
```

```
70:
71:     save_downloaded_file(header.decode(), file_name, True)
72:
73:     if is_file_request:
74:         handle_file_request(s, result, file_name)
75:     else:
76:         handle_http_request(s, result, file_name)
77:
78:
79: def create_socket_server(domain, port):
80:     """ creates socket server and request the given url"""
81:     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
82:     s.connect((domain, port))
83:     return s
84:
85:
86: def start_server(clean_url):
87:     """ entry point of the application """
88:     url = url_with_protocal(clean_url)
89:     HOST, DOMAIN, PATH, _, PARAMETERS, FRAGMENTS = urlparse(url)
90:     PORT = get_port_number(DOMAIN)
91:     print("Protocal used: {} ".format(HOST))
92:     org_path = PATH
93:     if PATH == '':
94:         PATH = '/'
95:
96:     if not is_empty_string(PARAMETERS):
97:         PATH = PATH + '?' + PARAMETERS
98:
99:     if not is_empty_string(FRAGMENTS):
100:         PATH = PATH + "#" + FRAGMENTS
101:     # request = "GET / HTTP/1.1Host: " + PATH + " \r\n"
102:     # here we are using http version 1.0
103:     request = "GET " + PATH + " HTTP/1.0\r\n\r\n"
104:
105:     # lets create an INET, STREAMurling socket
106:     s = create_socket_server(DOMAIN, PORT)
107:     # lets send request to the server
108:     s.send(request.encode())
109:     handle_receving_data(s, org_path)
110:     s.close()
111:
112: if __name__ == '__main__':
113:     input_url = sys.argv[1]
114:     start_server(input_url)
```

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

Listing 3 downloadEverything.py

```
1: # pylint: disable-msg=C0103
2: """ download every image from server"""
3: import re
4: import sys
5: from httpclient import start_server
6:
7:
8: list_of_images = []
9: input_file = sys.argv[1]
10: url = sys.argv[2]
```

```
11: images_with_full_path = []
12:
13:
14: def img_full_path(img_path, input_url):
15:     """ build full path of the image
16:     """
17:     if img_path.find("://") == -1:
18:         return input_url.strip() + img_path
19:     return img_path
20:
21:
22: with open(input_file, 'r') as f:
23:     santences = f.read()
24:     matches = re.findall('src="([^"]+)"', str(santences))
25:     list_of_images = list(filter(lambda url: url.find('.js') < 0, matches))
26:
27:
28: for i, img_url in enumerate(list_of_images):
29:     images_with_full_path.append(img_full_path(img_url, url))
30:
31:
32: # printing all the url to the console
33: for i, img_url in enumerate(images_with_full_path):
34:     print(img_url)
35:
36: # download every file
37: for i, img_url in enumerate(images_with_full_path):
38:     start_server(img_url)
```

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`.