INFO-H-511: Web Services
TP 1 - HTTP Programming

Lecturer: Stijn Vansummeren
Teaching Assistant: Francois Picalausa
http://cs.ulb.ac.be/public/teaching/infoh511

2011–2012

Part I: Manually Querying the Web

In this exercise, we will query the World Wide Web directly through the HTTP protocol. For this purpose, we will use the netcat utility. The general format of an HTTP 1.1 GET request is as follows[1]:

```
GET /index.html HTTP/1.1<CR><LF>
Host: www.example.com<CR><LF>
[Other headers]<CR><LF>
<CR><LF>
```

Exercise 1.1

In the following dialog, identify

- the lines sent by the client and those sent by the server,
- the HTTP method used,
- the status code with which the server answered,
- the content-type(s) accepted by the client/provided by the server,
- the actual data submitted in the request/answered by the server

```
POST /data/shorten HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept: application/json, text/javascript, */*; q=0.01
Referer: http://bitly.com/
Host: bitly.com
Content-Length: 141
Cookie: [...]

url=www.ulb.ac.be&basic_style=1&classic_mode=&rapid_shorten_mode=[...]

HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json; charset=UTF-8
Content-Length: 176
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{"status_code": 200, "data": {"url": "http://bit.ly/8UEV5N", "hash": "8UEV5N", [...]
```

[1] Complete reference available on the web

Exercise 1.2

Construct and execute the following HTTP requests:

- Retrieve the home page of this course (`http://cs.ulb.ac.be/public/teaching/infoh511`). Repeat with the HEAD method instead of GET. Explain the difference.
- Retrieve the footer image of that web page ((`http://cs.ulb.ac.be/public/lib/tpl/ulb/images/footer.gif`). What is the Content-Type being served? Display this image to ensure that you received it correctly.
- GET Google in German. Hint: use the 'Accept-Language' header.
- Request `http://dbpedia.org/resource/Berlin` in both the `application/rdf+xml` and `text/html` media types. Explain the result.

Part II: Interrogating REST-style Web Services

In the first part of this lab, we have explored ways of manually interrogating the web. We turn to programmatic ways to query the Web, over full-fledged APIs. For the following exercises, you are free to use your favorite programming language. However, skeleton code as well as solution to the exercises are currently only available in Java.

Exercise 1.3

Yahoo! and Bing both provide REST APIs to deal with map and address data. In this exercise, we will use Yahoo! PlaceFinder to convert a street name to a location, and Bing to display the corresponding map. For this purpose, we will use Apache HTTP Components[2], which provides a low-level library for making HTTP requests.

- Download the code skeleton available on the Labs webpage (`http://cs.ulb.ac.be/public/teaching/infoh511`).
- Fill-in the `getLocations` method to retrieve the corresponding locations, with an HTTP request to the Yahoo! PlaceFinder[3] service.
- Add the necessary code to parse the response in `parseYahooServiceResponse`.
- Finally, add the necessary code to download a map in `getMapImagery`, using Bing Map Imagery API[4]. A key to access this service will be provided for the duration of the lab.

Exercise 1.4

Google provides an extensive RESTful API to create and manipulate Google Documents. For this exercise, a small client application has been written that creates an Employees spreadsheet and populates it with some data. You are tasked to write the library that sends application calls to Google API, and translate the results back. For this exercise, we will use the Jersey Client API which provides a more high-level client interface to the HTTP protocol.

- Use the Google Documents List API[5], to implement `createSpreadsheet`, a method of the class `GoogleSpreadsheetClient` . In this method, you will need to POST a document.
- Continue implementing other methods, using GET, POST, PUT, and/or DELETE, as documented in Google Spreadsheets API[6]

[2] `http://hc.apache.org/httpcomponents-client-ga/`
[3] `http://developer.yahoo.com/geo/placefinder/guide/index.html`
[4] `http://msdn.microsoft.com/en-us/library/ff701724.aspx`
[5] `http://code.google.com/apis/documents/`
[6] `http://code.google.com/apis/spreadsheets/`