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

Clone this lab:
`git clone http://wit-projects.ulb.ac.be/rhodecode/INFO-H-511/Labs/Lab1-exercises`

Part I: Manually Querying the Web

In this exercise, we will query the World Wide Web directly through the HTTP protocol. Recall that 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
```

[1] Complete reference available on the web

```
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", [...]
```

Exercise 1.2

For the following exercise, you can use `curl` utility from a terminal. To run curl with a given HTTP method, and a custom header, call it as follows:

```
curl -v -X <method> -H "<header>: <value>" <url>
```

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.
- **Supplemental** 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 services over HTTP.

Exercise 1.3

Yahoo! and Bing both provide HTTP 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. A code skeleton is provided in the `MapClient` directory.

- Update the `getLocations` functions to query the Yahoo! PlaceFinder[2] service service.
- Parse the service response in the `parseYahooServiceResponse`, and `parseLocation` functions.
- Download a map in the `getMapImagery` function, with Bing Map [3]. 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, we will call the Google Spreadsheet API to create a new Spreadsheet with a few data programmatically.

- Locate the `createSpreadsheet` function of the `GoogleSpreadsheetClient` class in the code. Using Google Documents List API[4], describe what the function does, and how it proceeds to do so.
- Continue implementing other methods, using GET, POST, PUT, and/or DELETE, as documented in Google Spreadsheets API[5], so that the `main` function works.

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