INFO-H-511: Web Services

Stijn Vansummeren

The Labs

The Labs

Teaching Assistant:

Stefan Eppe (`stefan.eppe@ulb.ac.be`)

Goals

- Gaining **practical knowledge** of the Web Services concepts
- Bootstrapping your **project**

4 labs

1. HTTP Programming (today)
2. Developing REST Services (13 March)
3. Using & Developing SOAP Services (20 March)
4. Presenting Web Services (3 April)

A practical note

Programming language

- You are *free* to choose your tools to solve the exercises.
- However, skeleton code and solutions are currently only provided in *Java*.

Lab 1: HTTP Programming

Schedule for this lab

- 16:10 – 16:45: Manual HTTP queries → Exercises 1.1 and 1.2.
- 16:55 – 18:00: Interrogating a RPC-style Web Service → Exercise 1.3

Part I: Manually Querying the Web

Dissecting an HTTP request

```
GET rfc5023#section-10 HTTP/1.1
Host: tools.ietf.org
User-Agent: Mozilla/5.0 (Windows ...
Accept: text/html,...;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
If-Modified-Since: Mon, 09 Jan 2012 22:47:46 GMT
If-None-Match: "1827ba7-21df5-4b6202feb4880;4b6ff84c9e6cf"
Cache-Control: max-age=0
```

Part I: Manually Querying the Web

Dissecting an HTTP response

```
HTTP/1.1 200 OK
Date: Fri, 27 Jan 2012 00:46:12 GMT
Server: Apache/2.2.21 (Debian)
Content-Location: rfc5023.html
Last-Modified: Mon, 09 Jan 2012 22:47:46 GMT
Etag: "1827ba7-21df5-4b6202feb4880;4b777cde1187e"
Accept-Ranges: bytes
Content-Encoding: gzip
Content-Length: 29741
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=latin-1

<!DOCTYPE html PUBLIC "-//W3C//DTD
...
```

Part I: Manually Querying the Web

Sending requests

Many tools can be used to send HTTP requests: `firefox`, `curl`, `wget`, ...

In this first part, we will use curl:

```
curl -X <Method> -H "<header>: <value>" http://...
```

Part II: Interrogating a RPC-style Web Service

MapClient

We will implement a tool to fetch maps of given places:

```
java MapClient "Atomium" /tmp/map.png
```

Part II: Interrogating a RPC-style Web Service

Google Maps' Geocode and Staticmap are **RPC-style** services.

The **API**'s provide (cf. documentation):

1. the URI where the method is available,
2. the corresponding HTTP method and headers,
3. the format of the response

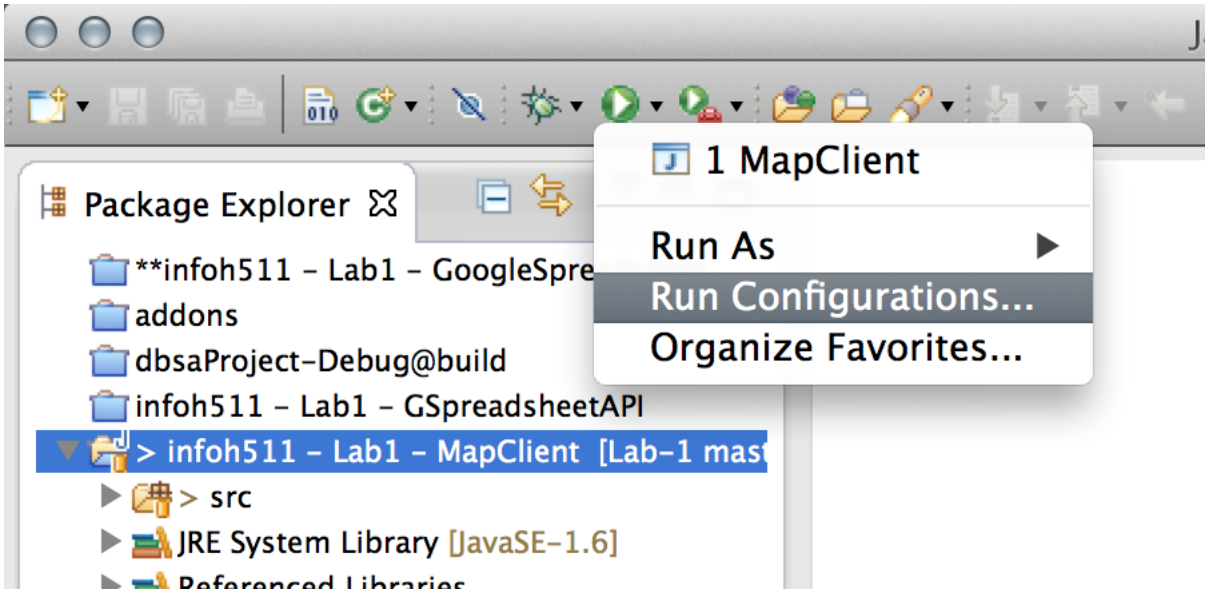
Part II: Interrogating a RPC-style Web Service

Requests with Jersey

```
String url = "http://www.example.org/map?place=Atomium";

Client client = ClientBuilder.newClient(new ClientConfig());
WebTarget target = client.target(url);

Response response = target.request().get();
InputStream data = response.readEntity(InputStream.class);

...

data.close();
```

Part II: Interrogating a RPC-style Web Service

Command line arguments in Eclipse



Part II: Interrogating a RPC-style Web Service

Command line arguments in Eclipse