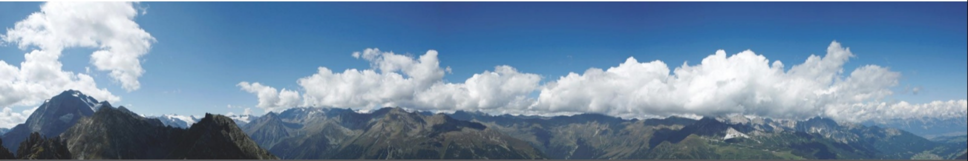
703128 PS/2 Web Services

HTTP

Tuesday, 2015-10-20



Schedule

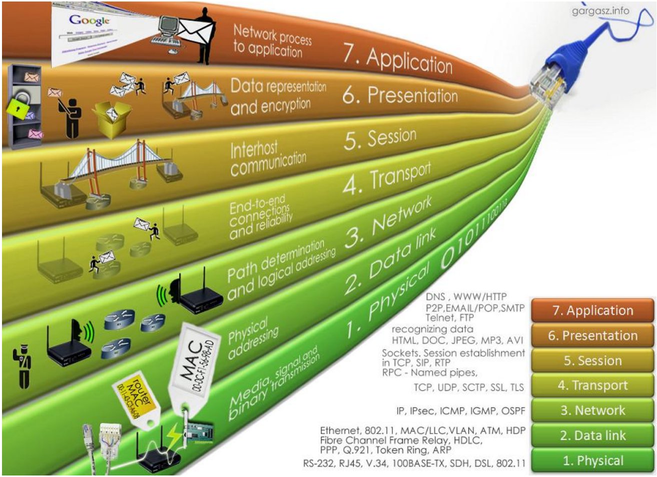
No.	Date	Bi-Weekly Exercise	Project	
			Presentation	Report
1	Tue, 2015-10-13			
2	**Tue, 2015-10-20**	✓		
3	Tue, 2015-10-27		✓	
4	Tue, 2015-11-03	✓		
5	Tue, 2015-11-10		✓	
6	Tue, 2015-11-17	✓		
7	Tue, 2015-11-24		✓	
8	Tue, 2015-12-01	✓		
9	Tue, 2015-12-15		✓	✓
10	Tue, 2016-01-12	✓		
11	Tue, 2016-01-19		✓	
12	Tue, 2016-01-26	✓		
13	Tue, 2016-02-02		✓	✓

Outline

- HTTP
  - Overview
  - Bi-Weekly Exercise
- Project
  - Team
  - Project Preparation

HTTP Overview

OSI Reference Model



DNS , WWW/HTTP
P2P,EMAIL/POP,SMTP
Telnet, FTP

recognizing data
HTML, DOC, JPEG, MP3, AVI

Sockets. Session establishment
in TCP, SIP, RTP
RPC - Named pipes,

TCP, UDP, SCTP, SSL, TLS

IP, IPsec, ICMP, IGMP, OSPF

Ethernet, 802.11, MAC/LLC,VLAN, ATM, HDP
Fibre Channel Frame Relay, HDLC,
PPP, Q.921, Token Ring, ARP

RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11

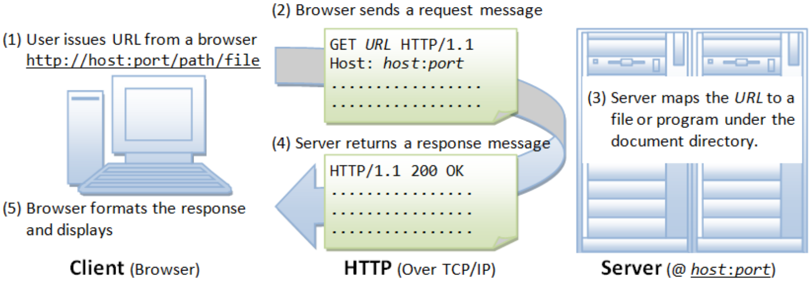
(*) http://www.gargasz.info/osi-model-how-internet-works/

More at https://en.wikipedia.org/wiki/OSI_model

The Basics: HTTP Protocol

- The Hypertext Transfer Protocol (HTTP) is an application-level TCP/IP based protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems (internet)
- Defines how to interact with Web servers:
  - Request-reply interactions (stateless)
  - Content types
  - One resource per request
  - Simple access control

HTTP Request/Response



(2) Browser sends a request message

(1) User issues URL from a browser
http://host:port/path/file

```
GET URL HTTP/1.1
Host: host:port
.................
.................
```

(3) Server maps the *URL* to a file or program under the document directory.

(4) Server returns a response message

```
HTTP/1.1 200 OK
................
................
................
```

(5) Browser formats the response and displays

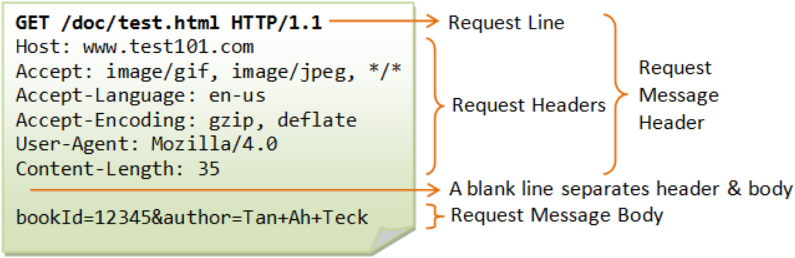
Client (Browser)

HTTP (Over TCP/IP)

Server (@ *host:port*)

(*) http://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

HTTP Request Anatomy

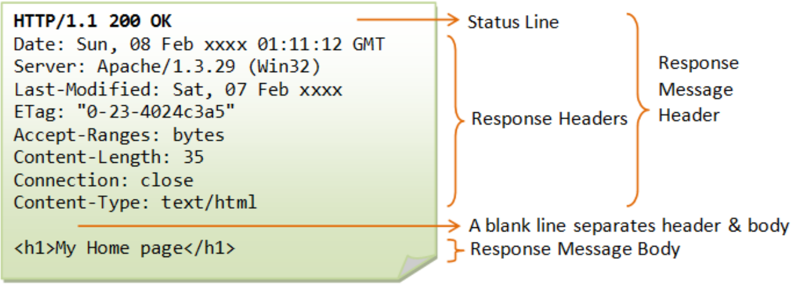


```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

(*) http://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

HTTP Response Anatomy



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

Status Line

Response Headers

Response Message Header

A blank line separates header & body

Response Message Body

(*) `http://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html`

HTTP Request Methods

- GET → retrieve a web resource from the server
- HEAD → same as GET but returns no data, only information about it
- POST → send data to a resource on the server
- PUT → store the data passed in the request with the URL as identifier
- DELETE → delete the resource identified by the URL
- OPTIONS → request the list of methods the server allows to be applied
- TRACE → sends back the request (diagnostic)

HTTP Response Codes

- 1xx: Information
- 2xx: Successful
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

Bi-Weekly Exercise

Send Request to Server

Use telnet [1]

```
$ telnet wikihow.com 80
GET /Main-Page HTTP/1.1
Host: www.wikihow.com
```

Use curl [2]

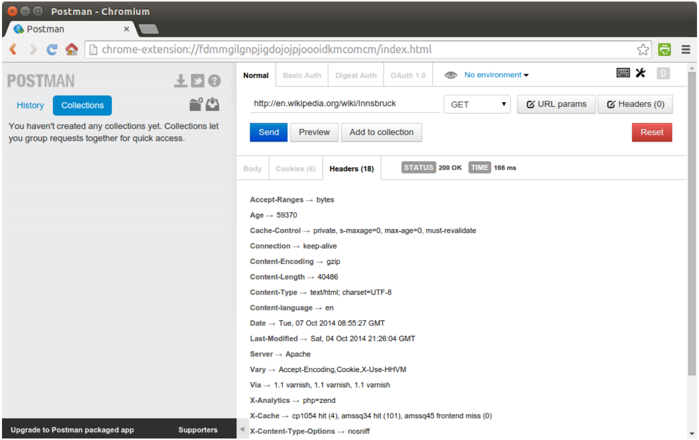
```
$ curl https://en.wikipedia.org/wiki/Innsbruck
```

[1] https://en.wikipedia.org/wiki/Telnet

[2] http://curl.haxx.se/

Send Request to Server

- Chrome Extension: Postman - REST Client



You haven't created any collections yet. Collections let you group requests together for quick access.

http://en.wikipedia.org/wiki/Innsbruck

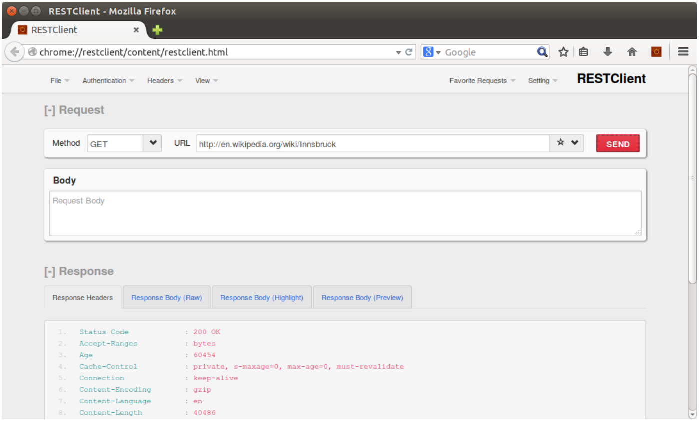
STATUS 200 OK TIME 166 ms

```
Accept-Ranges → bytes
Age → 59370
Cache-Control → private, s-maxage=0, max-age=0, must-revalidate
Connection → keep-alive
Content-Encoding → gzip
Content-Length → 40486
Content-Type → text/html; charset=UTF-8
Content-language → en
Date → Tue, 07 Oct 2014 08:55:27 GMT
Last-Modified → Sat, 04 Oct 2014 21:26:04 GMT
Server → Apache
Vary → Accept-Encoding,Cookie,X-Use-HHVM
Via → 1.1 varnish, 1.1 varnish, 1.1 varnish
X-Analytics → php=zend
X-Cache → cp1054 hit (4), amssq34 hit (101), amssq45 frontend miss (0)
X-Content-Type-Options → nosniff
```

(*) `https://github.com/a85/POSTMan-Chrome-Extension/wiki`

Send Request to Server

- Firefox Extension: RESTClient



Method: GET

URL: http://en.wikipedia.org/wiki/Innsbruck

```
Status Code             : 200 OK
Accept-Ranges           : bytes
Age                     : 60454
Cache-Control           : private, s-maxage=0, max-age=0, must-revalidate
Connection              : keep-alive
Content-Encoding        : gzip
Content-Language        : en
Content-Length          : 40486
```

(*) http://restclient.net/

Examples

Try and see the response headers:

```
$ curl -I https://en.wikipedia.org/wiki/Innsbruck
HTTP/1.1 200 OK
Content-language: en
...

$curl -I https://de.wikipedia.org/wiki/Innsbruck
HTTP/1.1 200 OK
Content-language: de
...

$ curl -I https://en.wikipedia.org/Innsbruck
HTTP/1.1 404 Not Found
...
```

Examples

```
$ curl -I https://en.wikipedia.org/wiki/innsbruck
HTTP/1.1 301 Moved Permanently
Location: https://en.wikipedia.org/wiki/Innsbruck
...

$ curl -I -A "Mobile" https://en.wikipedia.org/wiki/Innsbruck
HTTP/1.1 302 Found
Location: https://en.m.wikipedia.org/wiki/Innsbruck
...

$ curl -I -A "Mobile" http://www.wikihow.com/Main-Page
HTTP/1.1 302 Found
Location: http://m.wikihow.com/Main-Page
...
```

Task

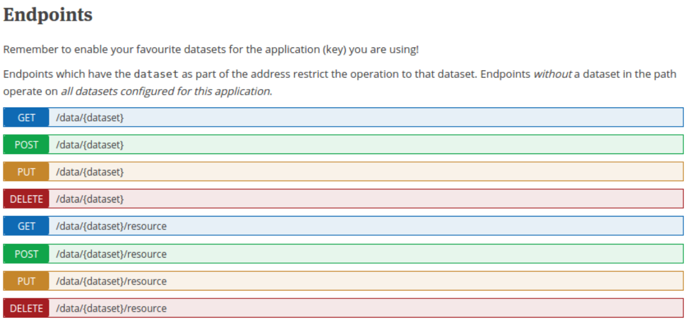
Individual Assignment:

1. Download the worksheet from the course webpage
2. Complete the exercises in the worksheet
3. Send it to the tutors (`ws1516@sti2.at`) before the next two sessions (Tuesday, 2015-11-03)

Team & Project Preparation

Access a Service (1)

A web API is a programmatic interface to a defined request-response message system (e.g. JSON or XML), which is exposed via the web (typically defined as a set of HTTP request-response messages)



Endpoints

Remember to enable your favourite datasets for the application (key) you are using!

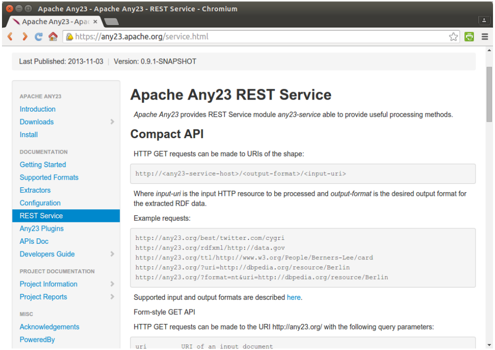
Endpoints which have the `dataset` as part of the address restrict the operation to that dataset. Endpoints *without* a dataset in the path operate on *all datasets configured for this application.*

GET	/data/{dataset}
POST	/data/{dataset}
PUT	/data/{dataset}
DELETE	/data/{dataset}
GET	/data/{dataset}/resource
POST	/data/{dataset}/resource
PUT	/data/{dataset}/resource
DELETE	/data/{dataset}/resource

(*) https://en.wikipedia.org/wiki/Application_programming_interface

Access a Service (2)

Structured data extractor. For example, Anything To Triples (Any23), a library, a web service and a command line tool to extracts structured data in RDF format from a variety of Web documents.



Last Published: 2013-11-03 | Version: 0.9.1-SNAPSHOT

Apache Any23 REST Service

Apache Any23 provides REST Service module *any23-service* able to provide useful processing methods.

Compact API

HTTP GET requests can be made to URIs of the shape:

```
http://<any23-service-host>/<output-format>/<input-uri>
```

Where *input-uri* is the input HTTP resource to be processed and *output-format* is the desired output format for the extracted RDF data.

Example requests:

```
http://any23.org/best/twitter.com/cygri
http://any23.org/rdfxml/http://data.gov
http://any23.org/ttl/http://www.w3.org/People/Berners-Lee/card
http://any23.org/?uri=http://dbpedia.org/resource/Berlin
http://any23.org/?format=nt&uri=http://dbpedia.org/resource/Berlin
```

Supported input and output formats are described here.

Form-style GET API

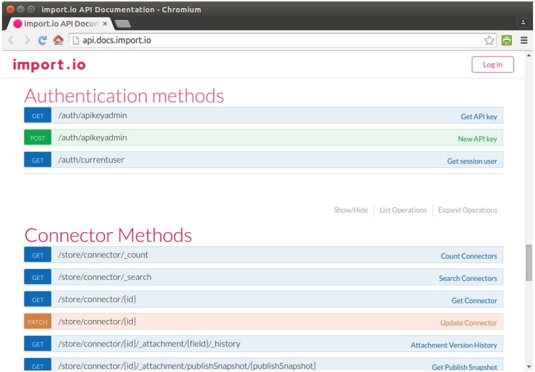
HTTP GET requests can be made to the URI http://any23.org/ with the following query parameters:

```
uri         URI of an input document
```

(*) `https://any23.apache.org/`, `http://any23.org/`

Access a Service (3)

Web (semi-structured) data extractor. For example, Import.io, a quick and powerful tool to extract data from webpages.



import.io API Documentation - Chromium

api.docs.import.io

import.io

Log in

Authentication methods

GET	/auth/apikeyadmin	Get API key
POST	/auth/apikeyadmin	New API key
GET	/auth/currentuser	Get session user

Show/Hide | List Operations | Expand Operations

Connector Methods

GET	/store/connector/_count	Count Connectors
GET	/store/connector/_search	Search Connectors
GET	/store/connector/{id}	Get Connector
PATCH	/store/connector/{id}	Update Connector
GET	/store/connector/{id}/_attachment/{field}/_history	Attachment Version History
GET	/store/connector/{id}/_attachment/publishSnapshot/{publishSnapshot}	Get Publish Snapshot

(*) `https://import.io/`

Task

Group Assignment:

1. Prepare slides about your **project idea** to be presented in the next session (Tuesday, 2015-10-27)
2. Your slides should be covering the following items:
   - Team – introduce your team
   - Project Description – describe your project idea, i.e. motivations, goals, etc.
   - Identified Services – enumerate all **identified services** to be included in your project and **how to access each service** (e.g. through API calls, web crawling & scraping)
   - Technologies – explain what technologies will be utilized
   - Expected Results – explain the expected results
3. Remember: Your project idea should be focused more into the services composition

Thank You