Building RESTful API

Marek Konieczny

marekko@agh.edu.pl,

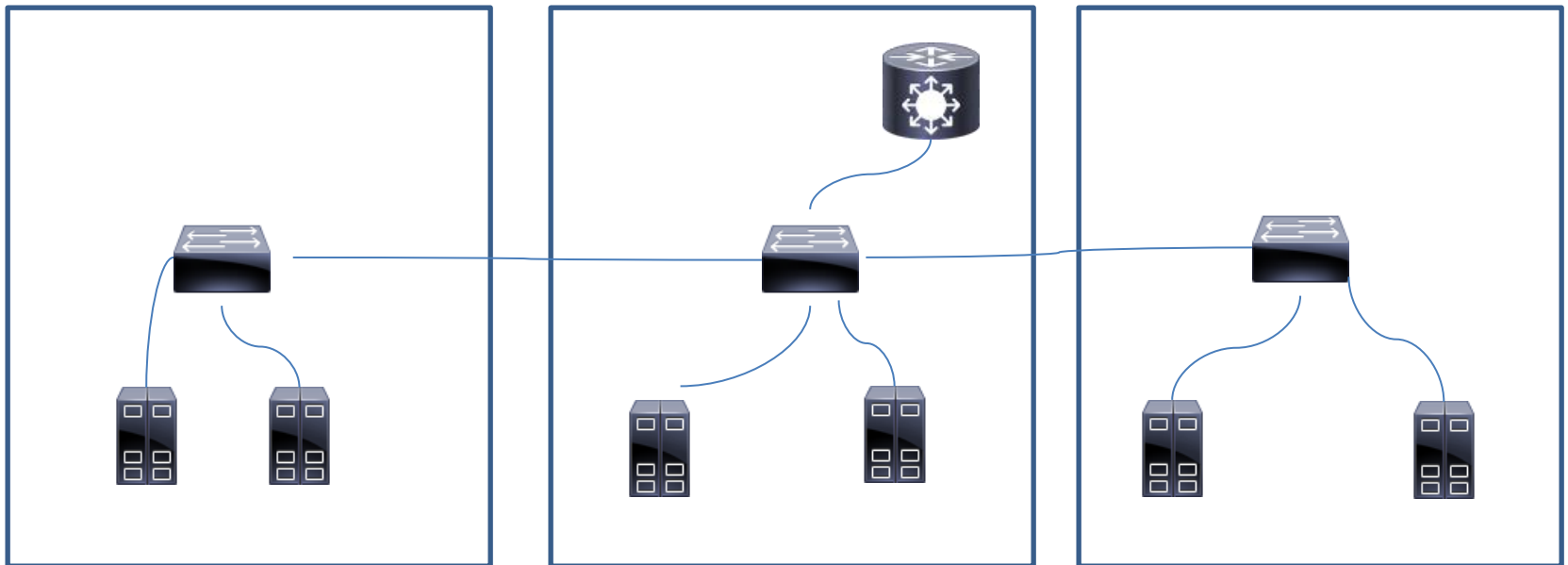Room 4.43, Spring 2023

AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Class Logistics

- We will meet on:
  - 7-10.3 laboratory sessions
  - 21-24.3 homework
- All materials on UPEL: including exam part
- Grading:
  - Showing up => +1
  - Hard work => +3
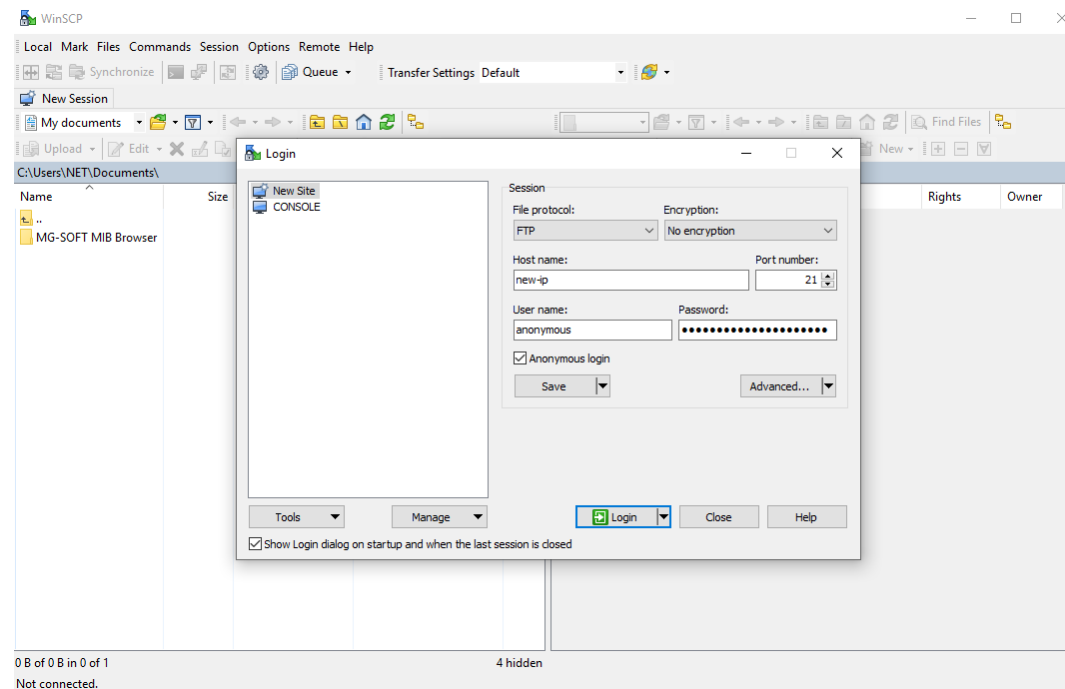 - Extra activity => +1

Environment preparation

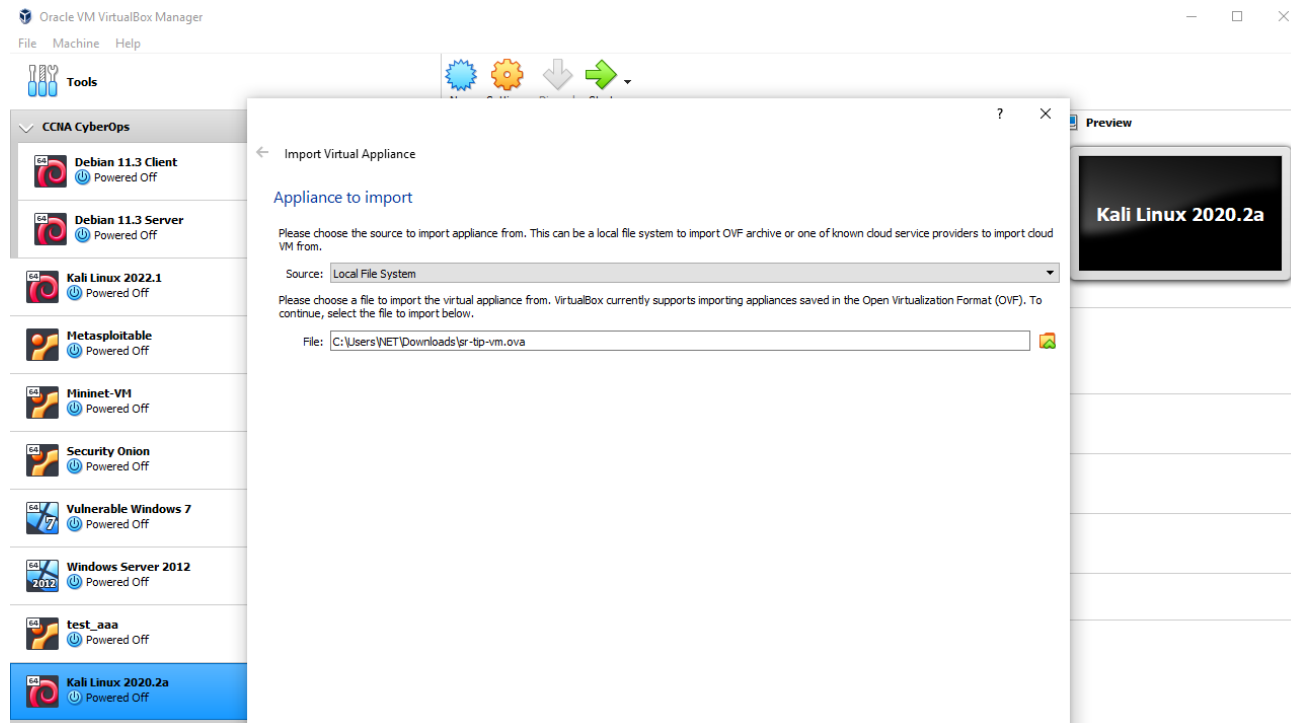- Wire-up all environment to have internet connection

Environment preparation

- Start the Windows NET images
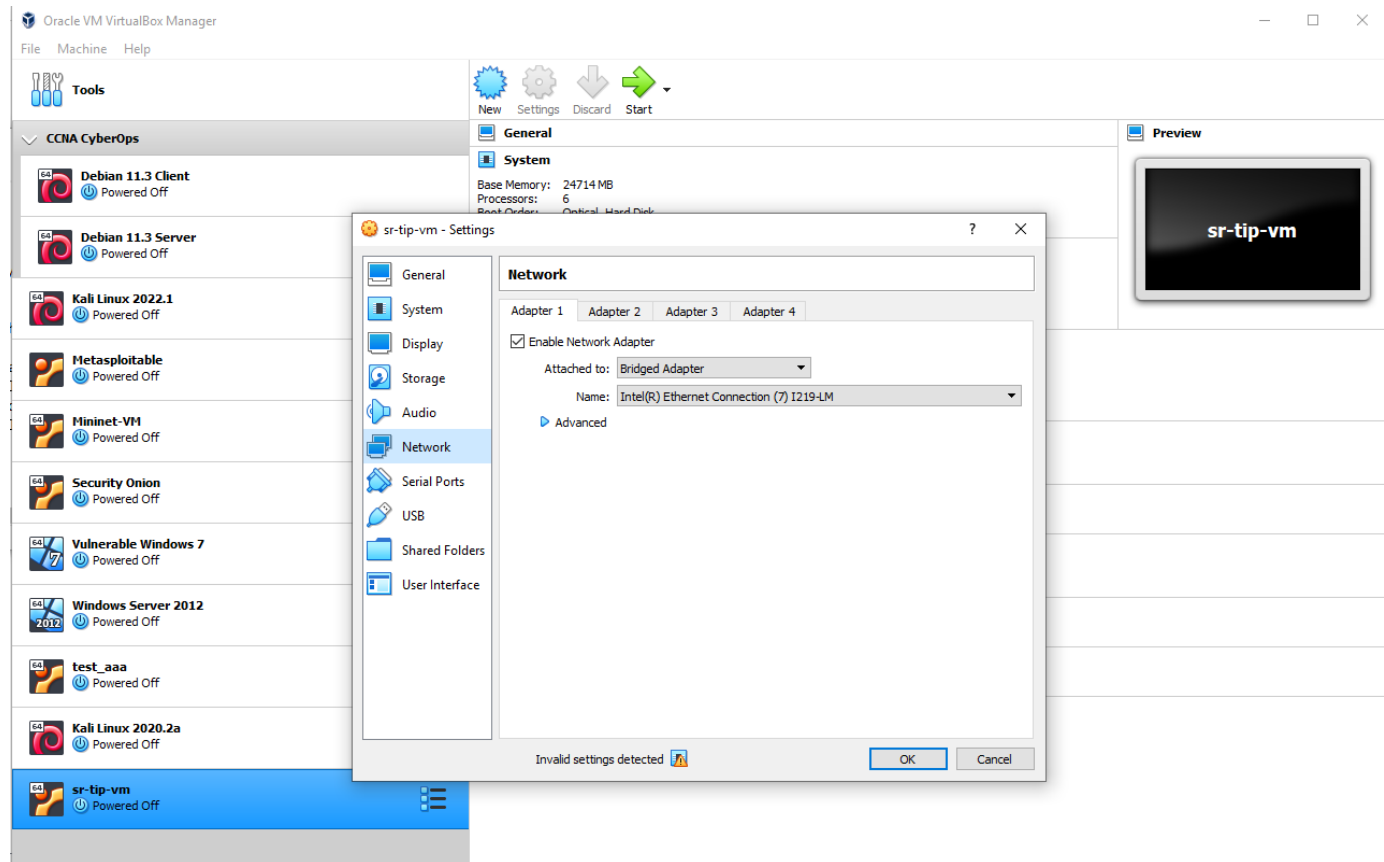- Open WinSCP:
  - *172.17.144.136*
- download image

Environment preparation

- Open VirtualBox

- Import appliance

Environment preparation

- Make sure you are using Bridged Adapter in network settings

Environment preparation

- Start Virtual Machine
  - You have access to Ubuntu with python and pycharm
  - Credentials: lab/lab

Origin

- **Re**presentational **S**tate **T**ransfer
- Architectural style
  - not dependent on any specific protocol
- Describes a set of principles derived from analysis of World Wide Web Architecture
  - To make any distributed system scalable

UNIVERSITY OF CALIFORNIA,
IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

Dissertation Committee:
Professor Richard N. Taylor, Chair
Professor Mark S. Ackerman
Professor David S. Rosenblum

Basics

- Resource
  - fundamental building block of web-based systems
- Web is often named „resource-oriented"
- Resource is anything with which consumer interacts while achieving some goal
- Resource e.g.: document, video, business process, device, spreadsheet, printer
- Exposition of resource to Web:
  - Abstracting out resource information aspects
  - Presenting these aspects to digital world by means of some representation

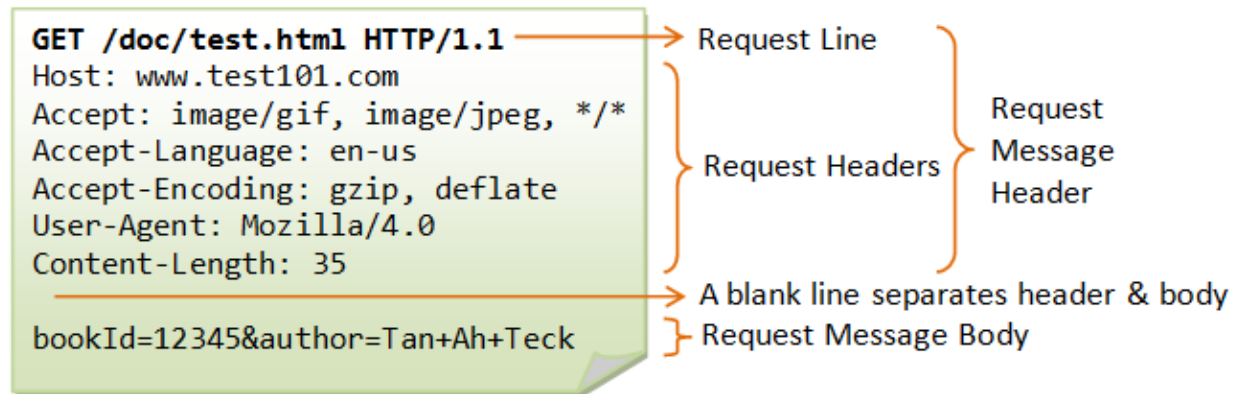# Uniform Resource Identification (URI)

- URI or URL?
  - URI = URL || URN
  - URN is not so popular (urn:oasis:names:specification:docbook:dtd:xml:4.1.2)
  - Usually URI = Uniform Resource Locator (URL)
- Different types : File URL, FTP URL, HTTP URL
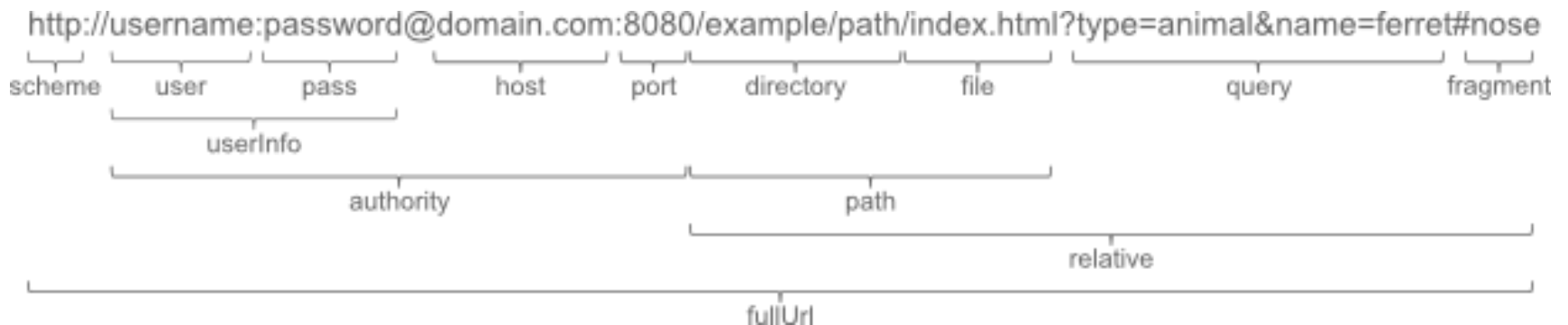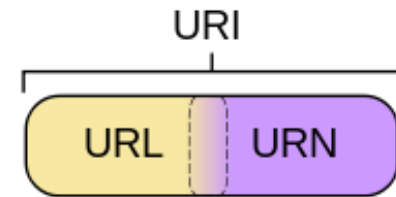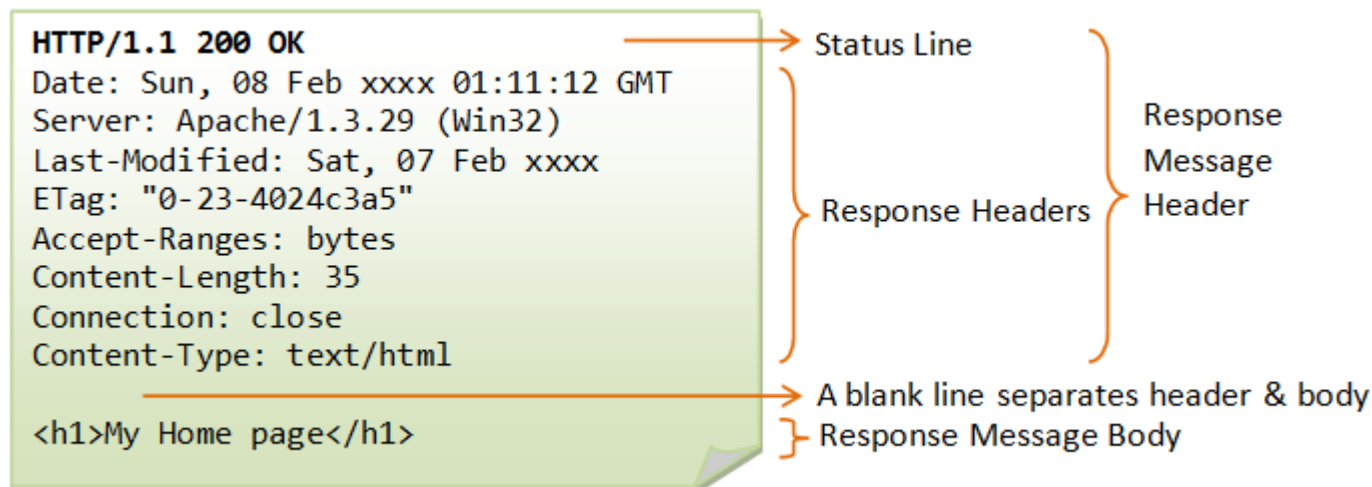
HyperText Transfer Protocol (HTTP)

- Protocol for distributed systems for sharing media
- Hypermedia: logic extension of hypertext which includes graphics, audio, video which creates new media
- Based on the request-response schema
  - Client send requests to server
- Format
  - Method line
  - Headers
  - Empty line
  - Optional body

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

                                  → A blank line separates header & body
bookId=12345&author=Tan+Ah+Teck    → Request Message Body
```
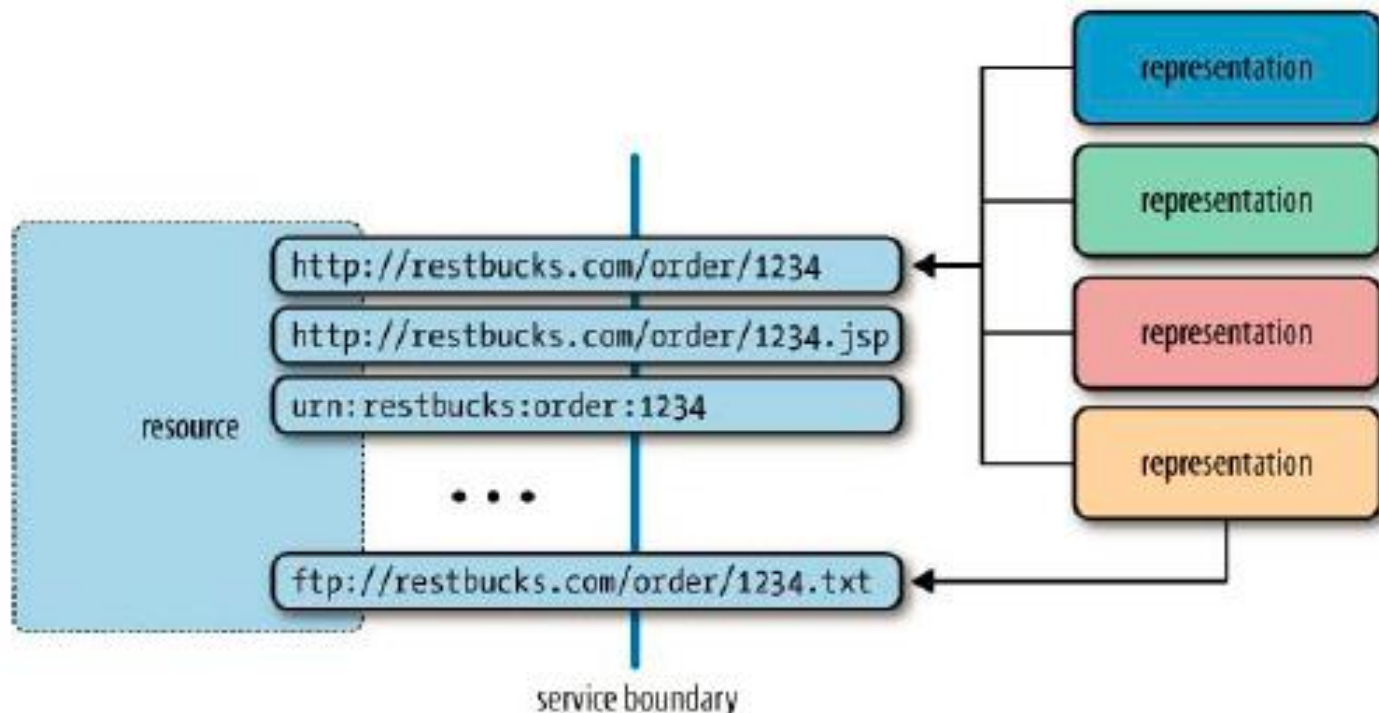
HyperText Transfer Protocol (HTTP)

- First line contains code of the response
  - Succes: 2xx
  - Redirections: 3xx
  - Error of client: 4xx
 - Server error: 5xx

- Later headers, and body

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

                                → A blank line separates header & body
<h1>My Home page</h1>           → Response Message Body
```

Services

- Entire state of system is exposed as resources

Services
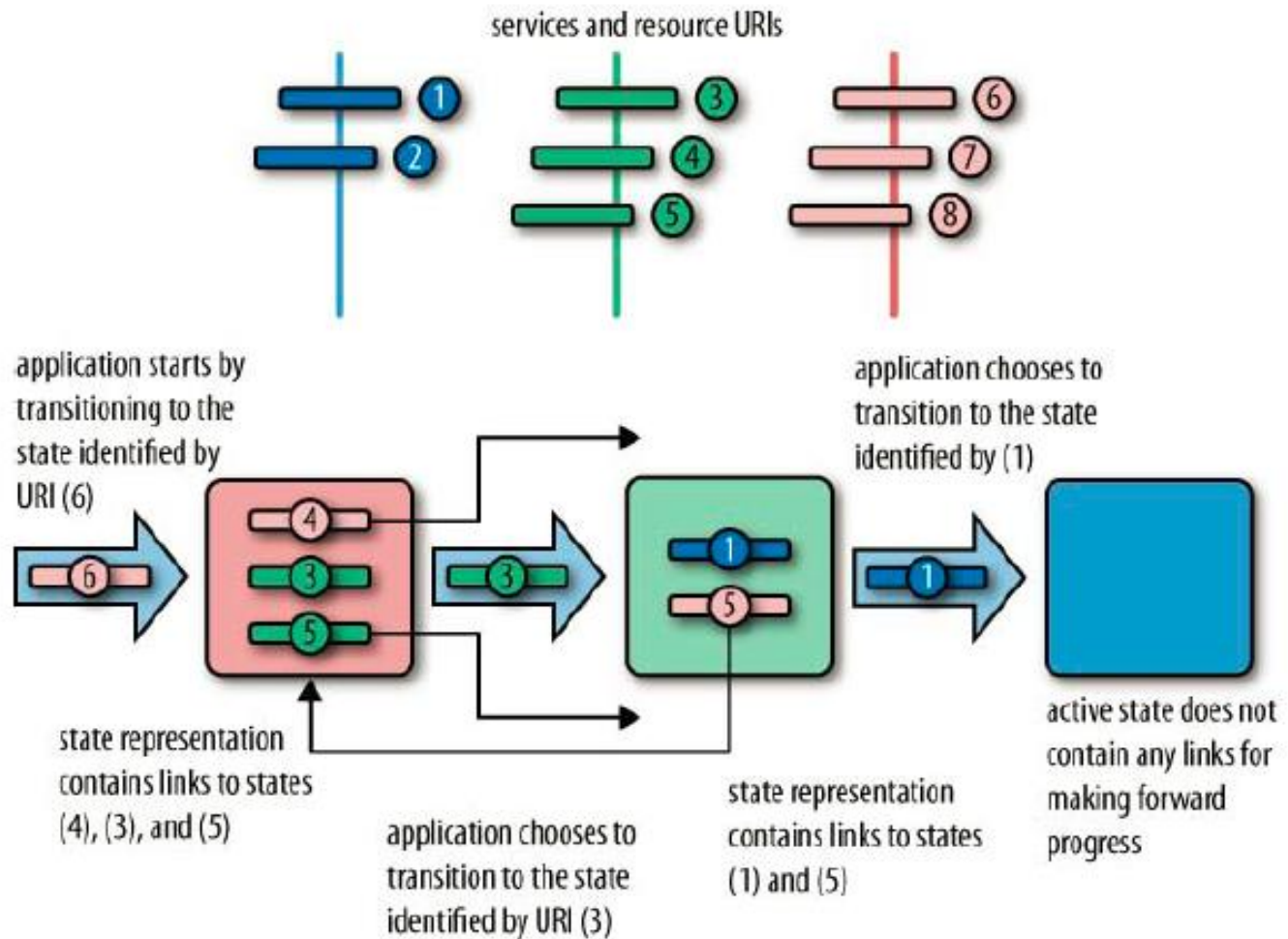
RESTful Level 2: Verbs Summary

HTTP Verb	Common Meaning	Safe	Idempotent
GET	Retrieve the current state of the resource	YES	YES
POST	Create a sub resource	NO	NO
PUT	Initialize or update the state of a resource at given URI	NO	YES
DELETE	Clear a resource	NO	YES

RESTful Maturity Model

Glory of REST

Level 3: Hypermedia Controls

Level 2: HTTP Verbs

Level 1: Resources

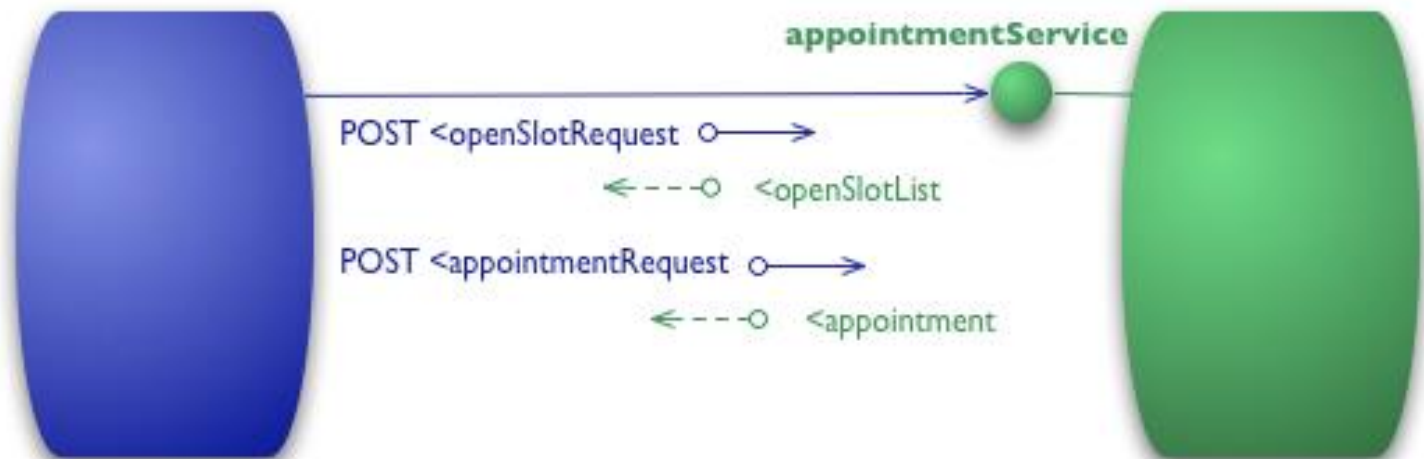Level 0: The Swamp of POX
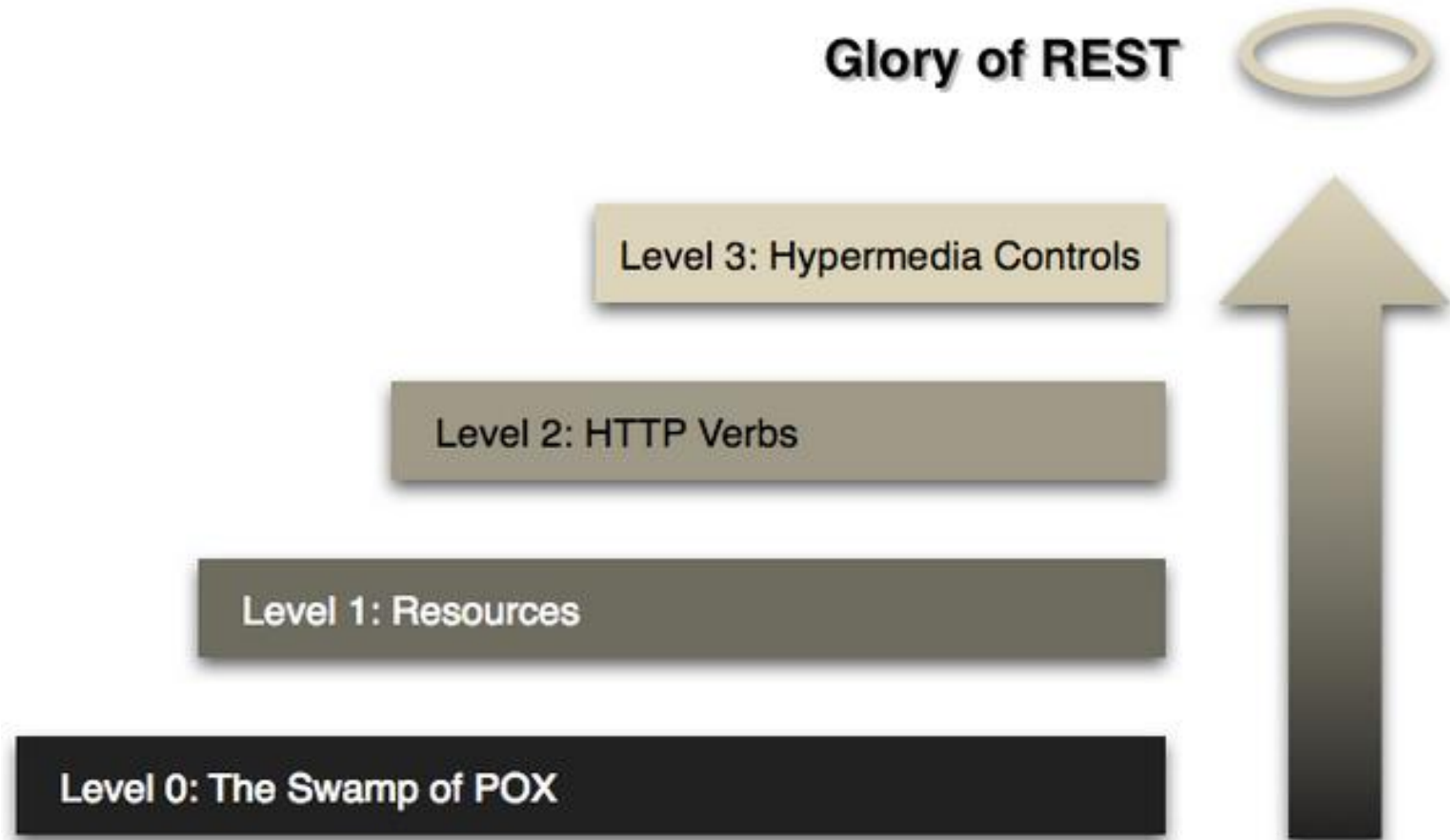
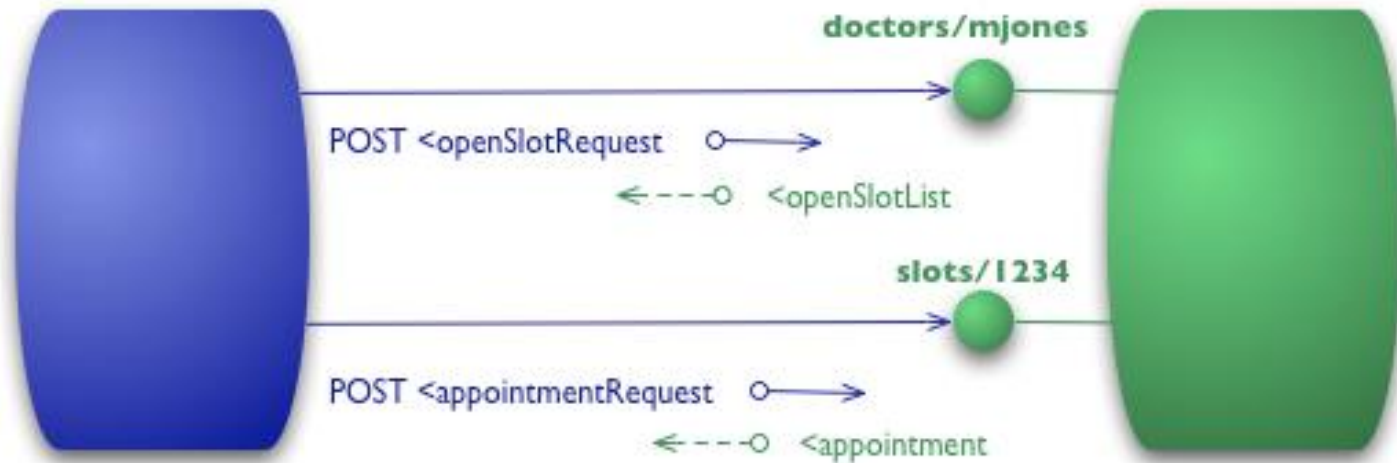# Level 0

```
POST /appointmentService HTTP/1.1
[various other headers]

<openSlotRequest date = "2010-01-04" doctor = "mjones"/>
```

Level 1

Hypermedia as the Engine of Application State (HATEOAS)

```
HTTP/1.1 201 Created
Location: http://royalhope.nhs.uk/slots/1234/appointment
[various headers]

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <link rel = "/linkrels/appointment/cancel"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/addTest"
        uri = "/slots/1234/appointment/tests"/>
  <link rel = "self"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/changeTime"
        uri = "/doctors/mjones/slots?date=20100104@status=open"/>
  <link rel = "/linkrels/appointment/updateContactInfo"
        uri = "/patients/jsmith/contactInfo"/>
  <link rel = "/linkrels/help"
        uri = "/help/appointment"/>
</appointment>
```

REST Design Methodology

1. Identify resources to be exposed as services
2. Model relationships between resources with hyperlinks
3. Define URIs to address the resources
4. Understand what it means to do a GET, POST, PUT, DELETE for each resource
5. Design and document resource representation
6. Implement and deploy on Web server
7. Test with a Web browser

	GET	PUT	POST	DELETE
/loan	✓	✓	✓	✓
/balance	✓	✗	✗	✗
/client	✓	✓	✓	✗
/book	✓	✓	✓	✓
/order	✓	?	✓	✗
/soap	✗	✗	✓	✗

Task 0 – environment setup

- Open attached distributed.py file
- You will need following packages:
    - pip3 install fastapi
    - pip3 install uvicorn
- Start web server inside the file location
    - uvicorn distributed:app --reload

Task 0 – environment setup

- Navigate to Swagger UI:
  - http://localhost:8000/docs
- Open API specification
  - http://localhost:8000/openapi.json

- Based on tutorials:
  - https://fastapi.tiangolo.com/tutorial/

Task 0 – environment setup

Task 1 – Doodle API example

- Create simple Doodle API
- Small API for voting
  - User can create poll (see what is insider poll)
  - User can cast a vote inside this polls
  - User can add, update and delete all information he provides
  - User can see the results of votes
- Construct API and build the system
  - Test it with the Swagger UI

Homeworks

– Simple projects detailed description on UPEL