Building RESTful API

Marek Konieczny

marekko@agh.edu.pl,

Room 4.43, Spring 2024

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Class Logistics

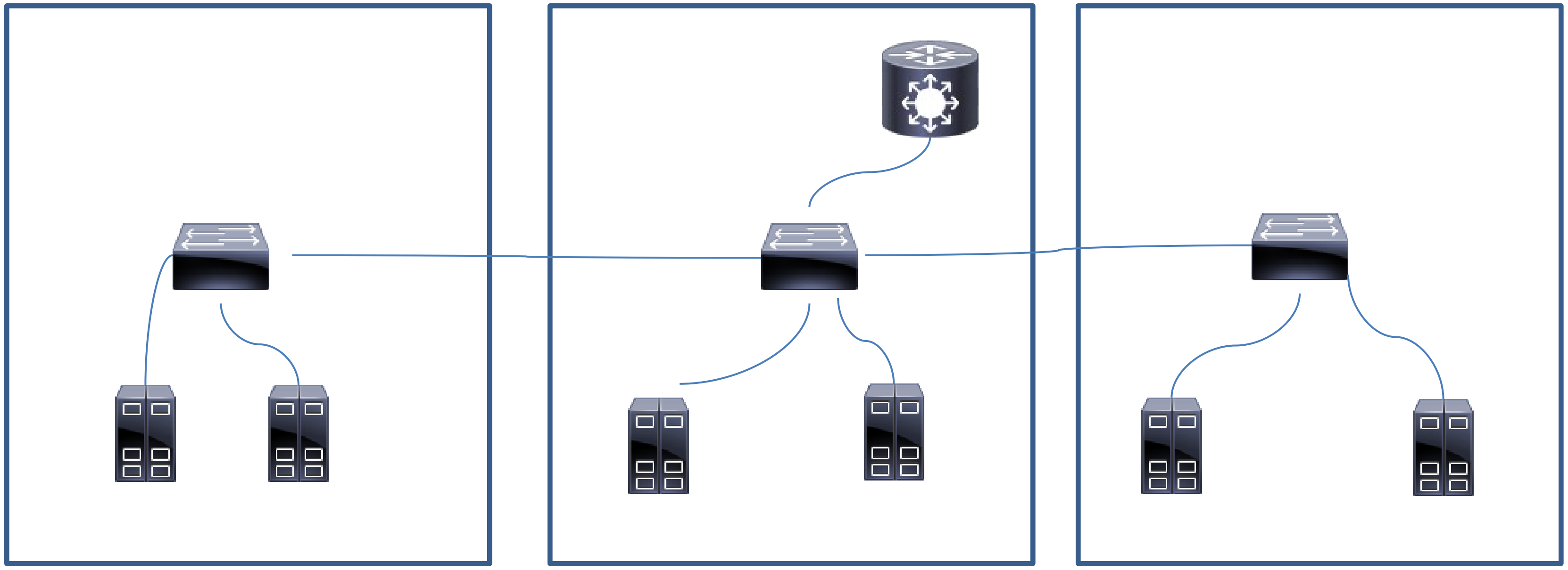
- We will meet on:
  - 4-8.3 laboratory sessions
  - 18-22.3 homework
- All materials on UPEL
- Grading:
  - Showing up => +1
  - Hard work => +2
  - Results on UPEL => +1
  - Extra activity => +1

Environment preparation

- You can use your laptops
  - You will need python
- Log in to desktops
  - Select Ubuntu image

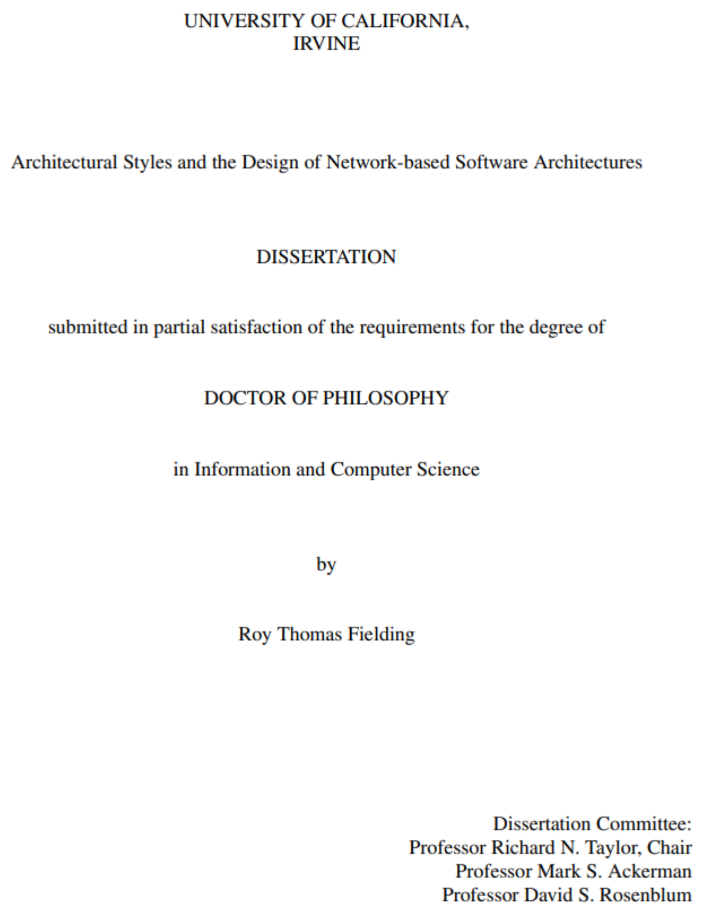
Environment preparation

- Wire-up all environment to have internet connection



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

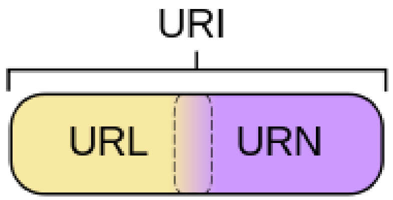
Roy Thomas Fielding

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

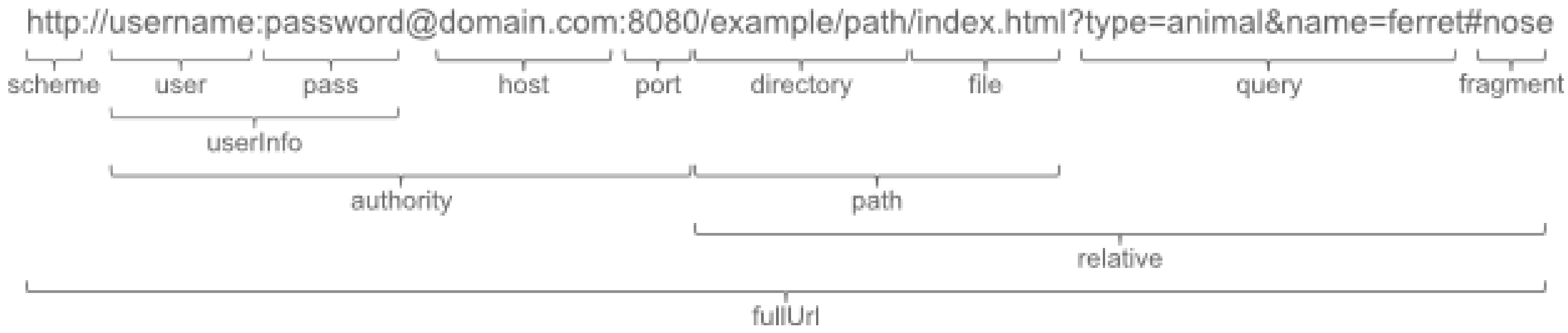
Basics

- 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)

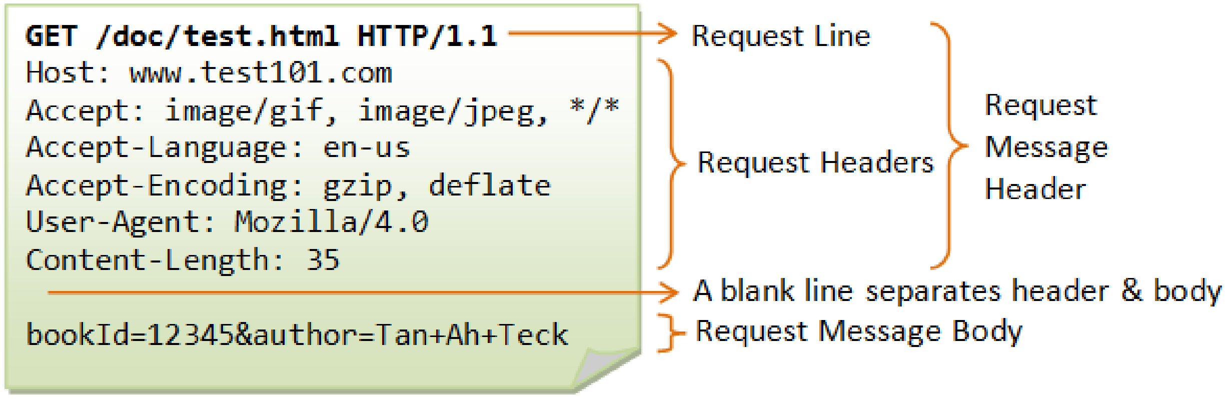


- URI or URL?
  - URI = URL || URN
  - URN is not so popular (urn:oasis:names:specification:docbook:dtd:xml:4.1.2)
  - Usually URI = Uniform Resource Locator (URL)
- Different types : File URL, FTP URL, HTTP URL


http://username:password@domain.com:8080/example/path/index.html?type=animal&name=ferret#nose
scheme
user
pass
host
port
directory
file
query
fragment
userInfo
authority
path
relative
fullUrl

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

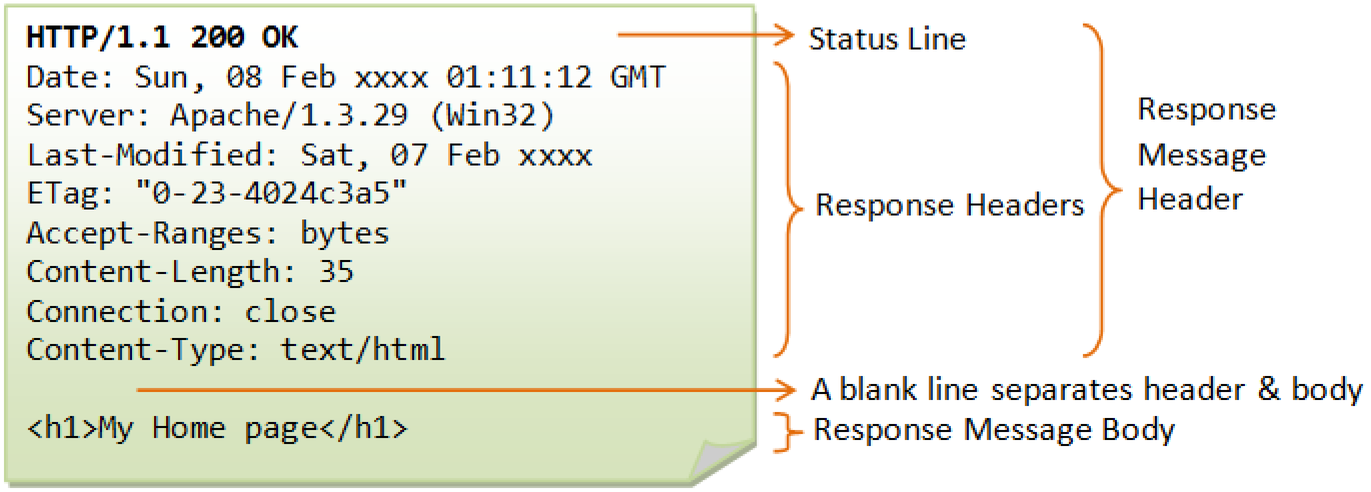


```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

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

HyperText Transfer Protocol (HTTP)

- First line contains code of the response
  - Succes: 2xx
    - Accepted, Created …
  - Redirections: 3xx
  - Error of client: 4xx
    - Not Found, Not Allowed, Unauthorized
  - Server error: 5xx
- Later headers, and body

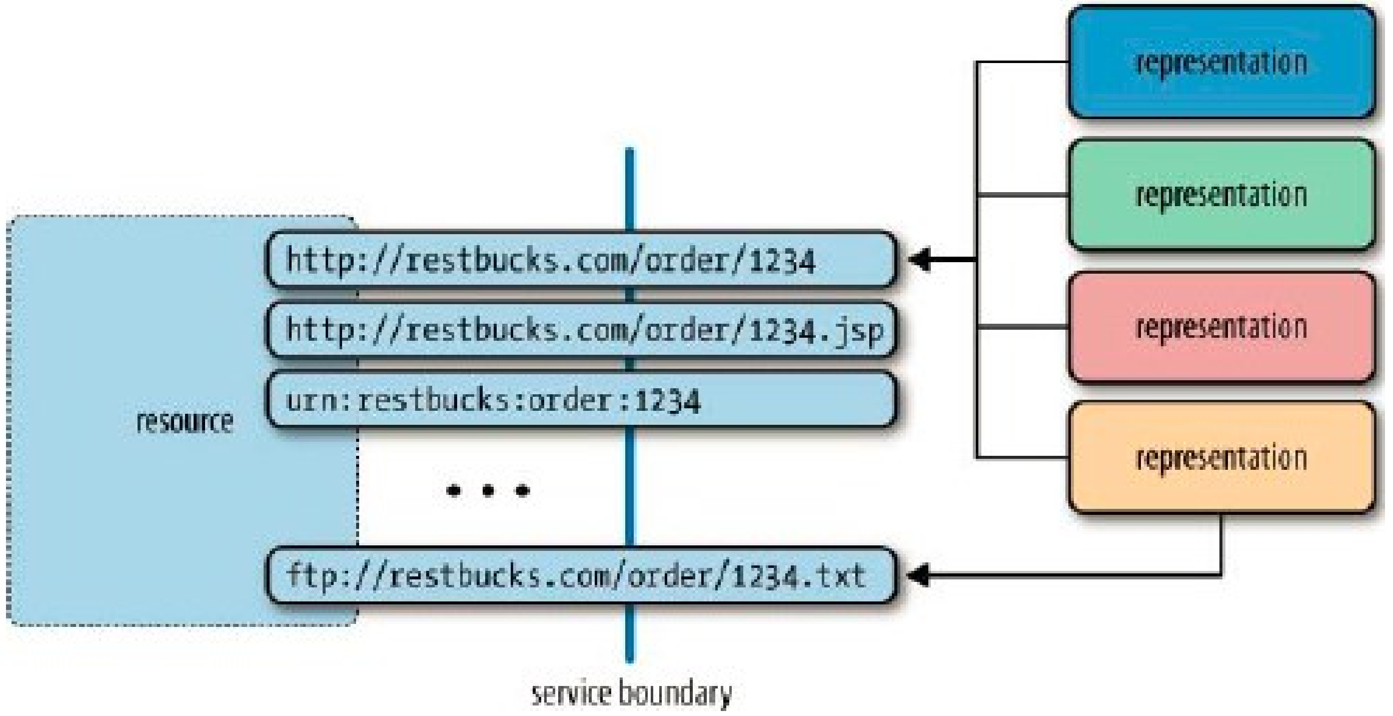
Sorry.
(for giving you a site error)
I may have shredded the power cord.
Oh, and your favorite shoes.
Can you please try again?



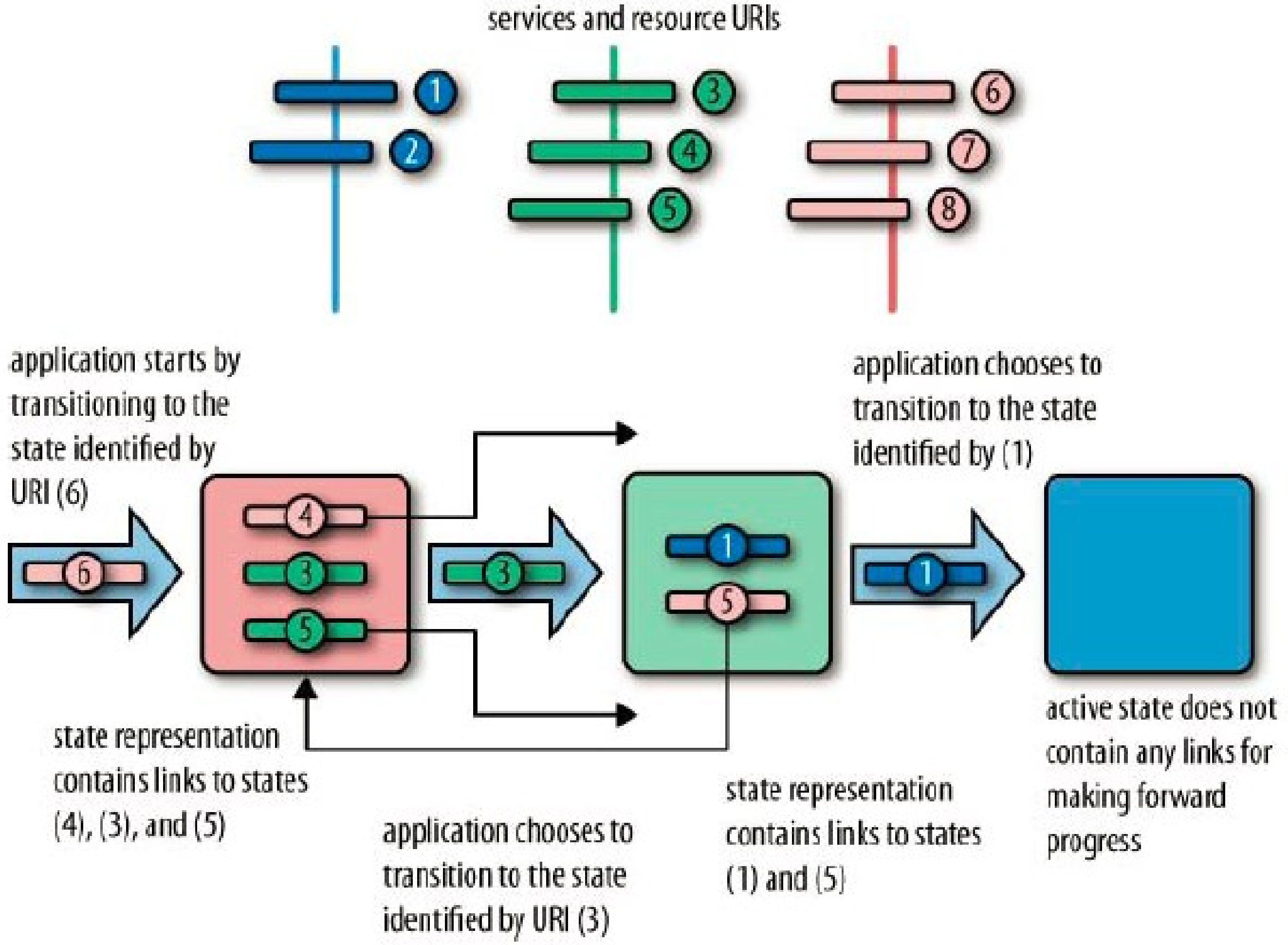
```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

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

Services

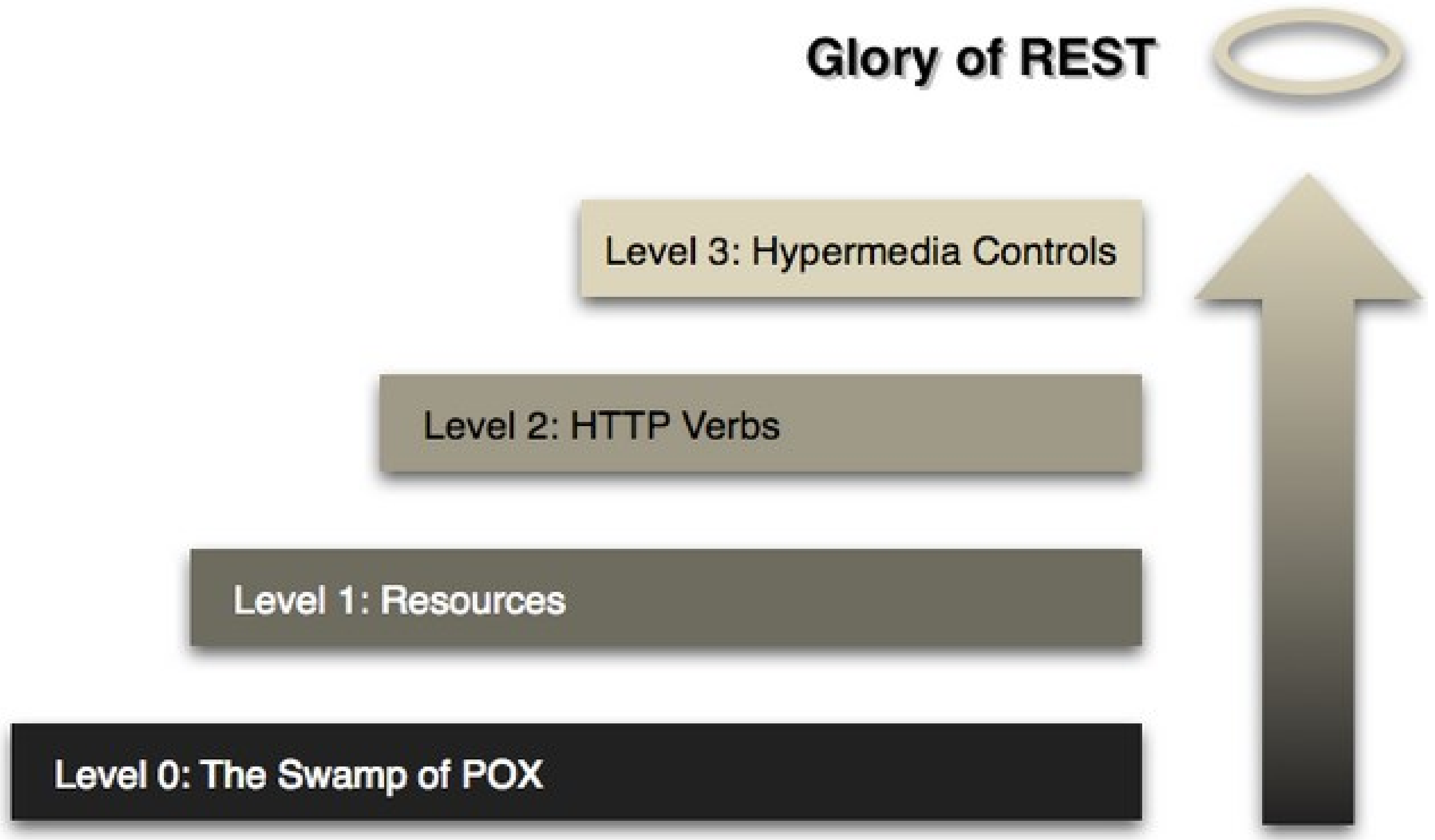
- Entire state of system is exposed as resources



Services

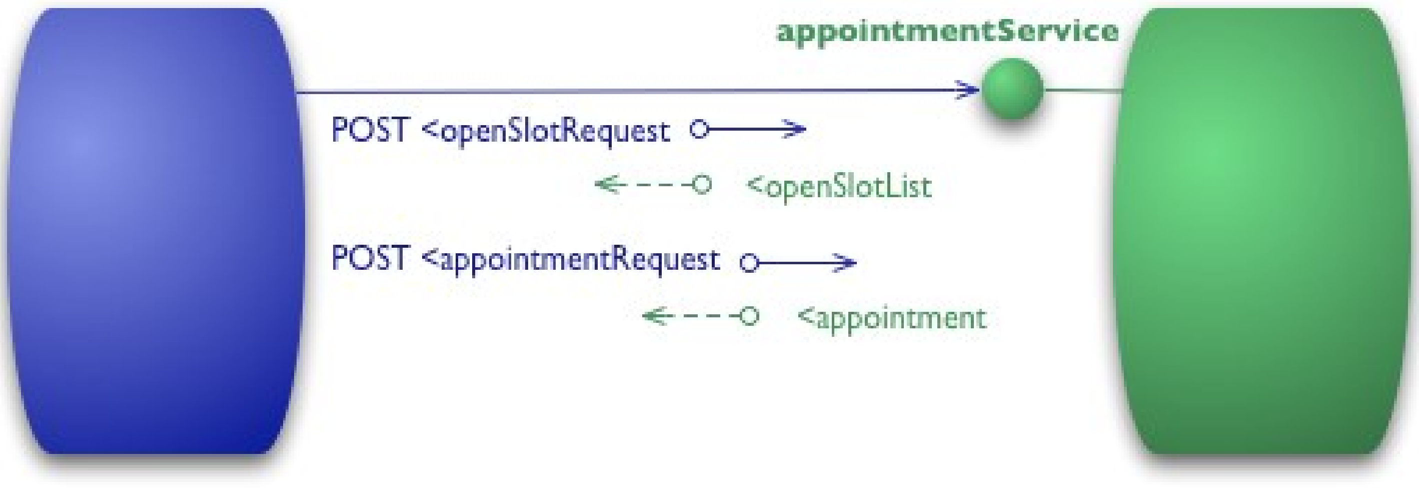

services and resource URIs
1
2
3
4
5
6
7
8
application starts by transitioning to the state identified by URI (6)
6
4
3
5
3
1
5
1
application chooses to transition to the state identified by (1)
state representation contains links to states (4), (3), and (5)
application chooses to transition to the state identified by URI (3)
state representation contains links to states (1) and (5)
active state does not contain any links for making forward progress

RESTful Maturity Model



Level 0

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

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



Level 0

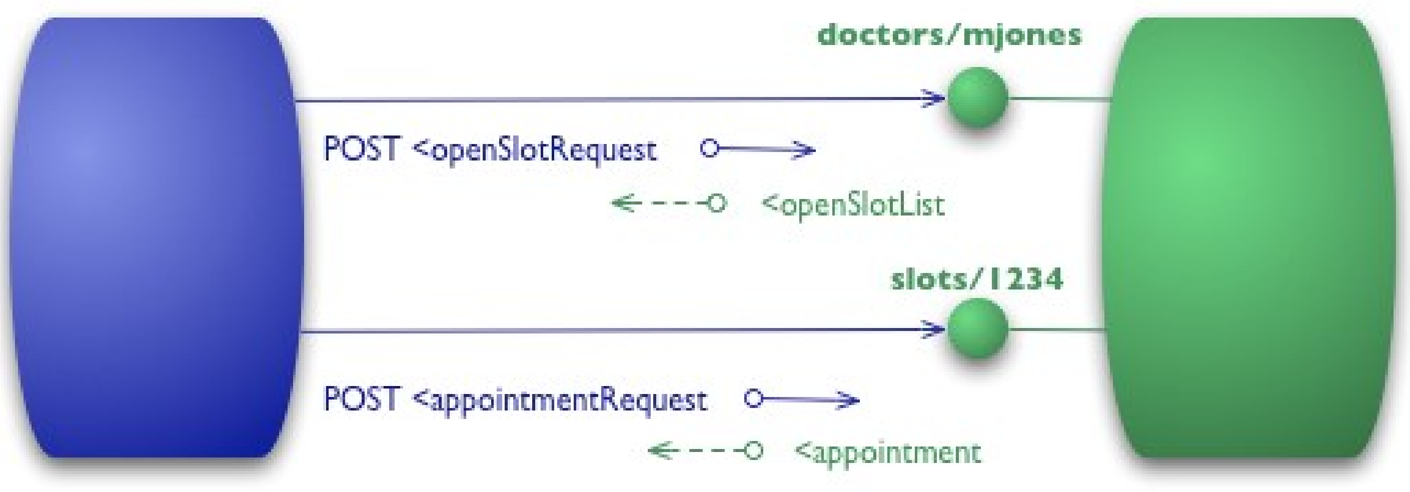
```
HTTP/1.1 200 OK
[various headers]

<openSlotList>
  <slot start = "1400" end = "1450">
    <doctor id = "mjones"/>
  </slot>
  <slot start = "1600" end = "1650">
    <doctor id = "mjones"/>
  </slot>
</openSlotList>
```

```
HTTP/1.1 200 OK
[various headers]

<appointmentRequestFailure>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <reason>Slot not available</reason>
</appointmentRequestFailure>
```

Level 1

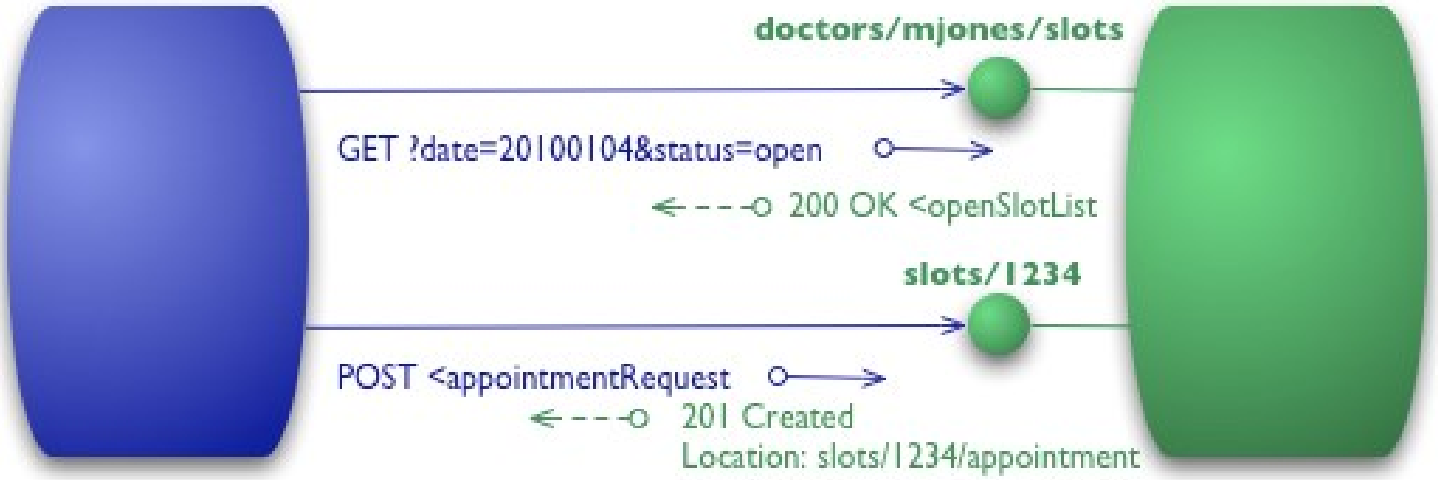
```
POST /doctors/mjones HTTP/1.1
[various other headers]

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

```
POST /slots/1234 HTTP/1.1
[various other headers]

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```



Level 2

```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
Host: royalhope.nhs.uk
```



Level 2

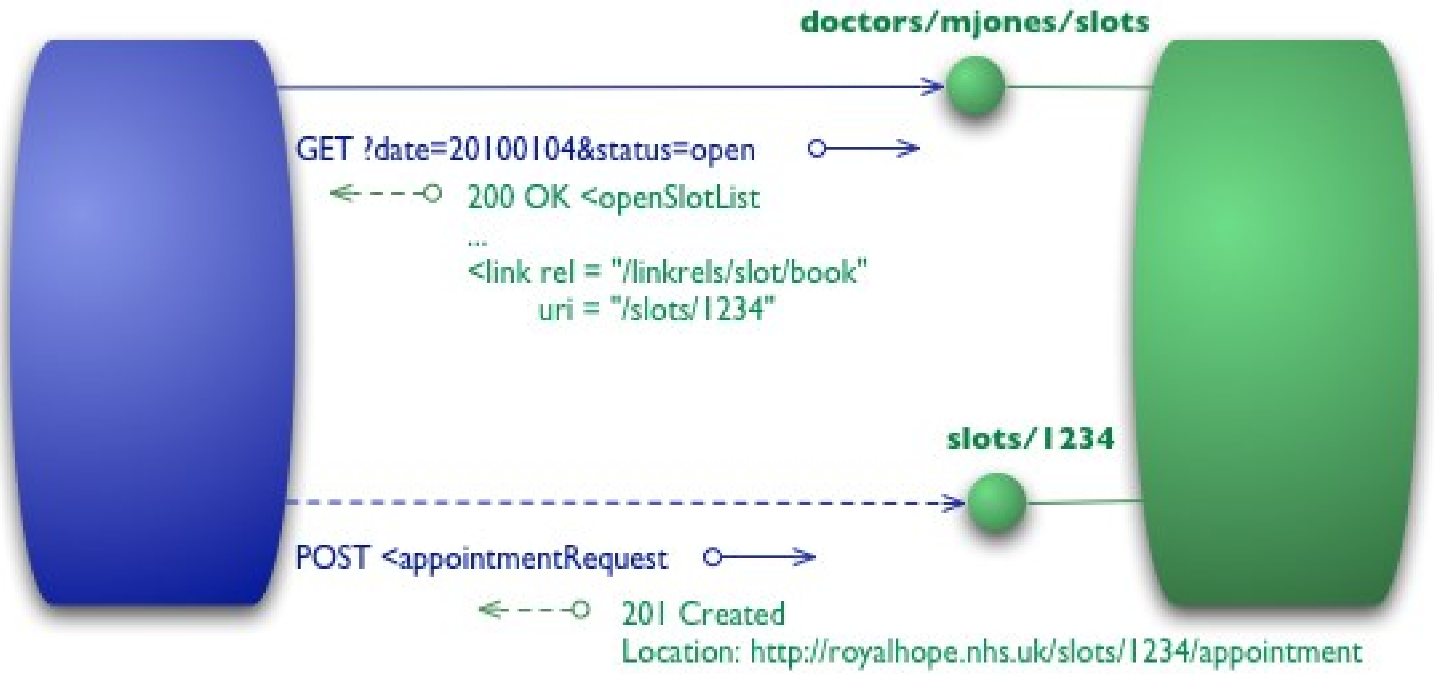
```
HTTP/1.1 201 Created
Location: slots/1234/appointment
[various headers]

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>
```

```
HTTP/1.1 409 Conflict
[various headers]

<openSlotList>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

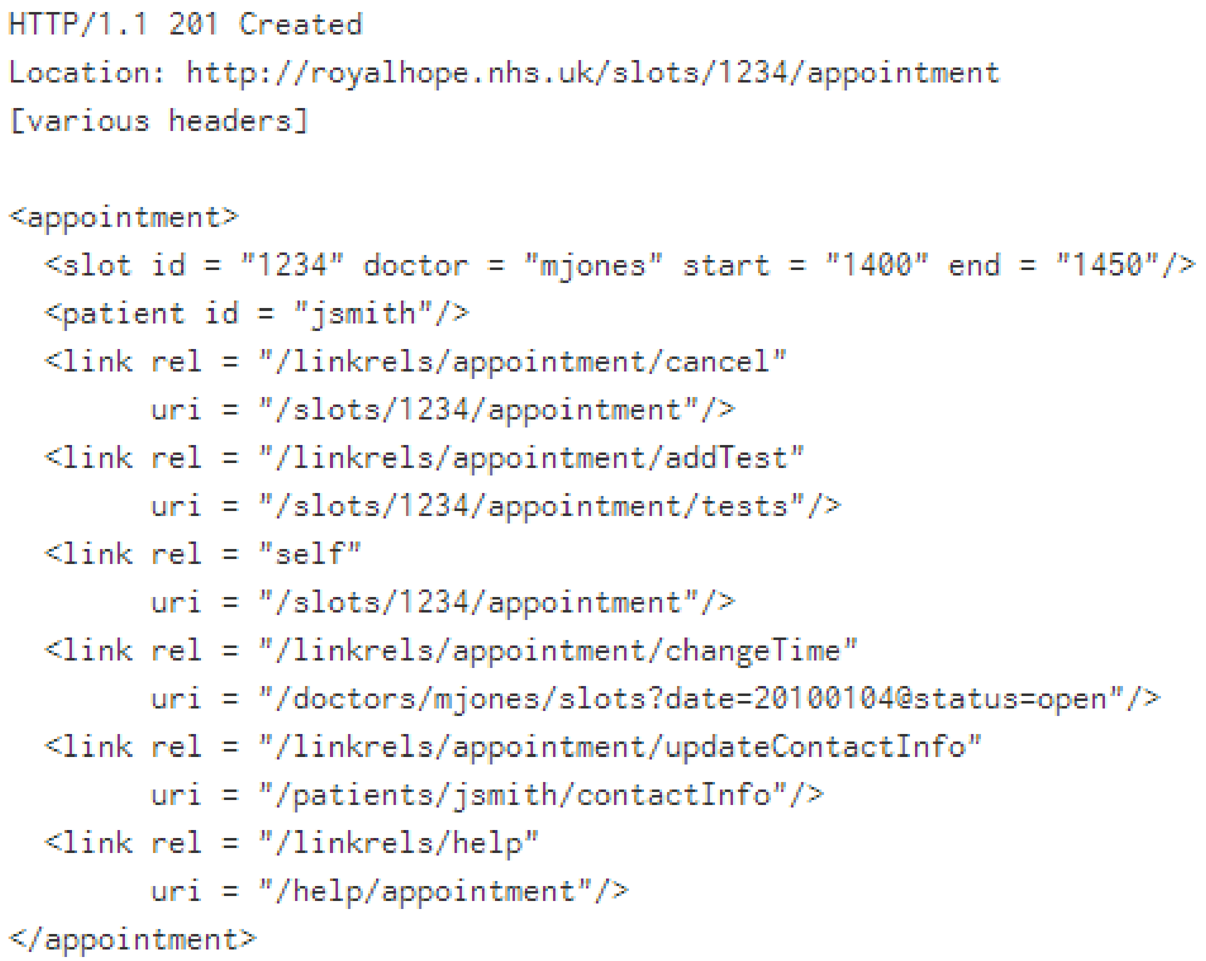
HTTP Methods as Verbs

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

Level 3


doctors/mjones/slots
GET ?date=20100104&status=open
200 OK <openSlotList
...
<link rel = "/linkrels/slot/book"
uri = "/slots/1234"
slots/1234
POST <appointmentRequest
201 Created
Location: http://royalhope.nhs.uk/slots/1234/appointment

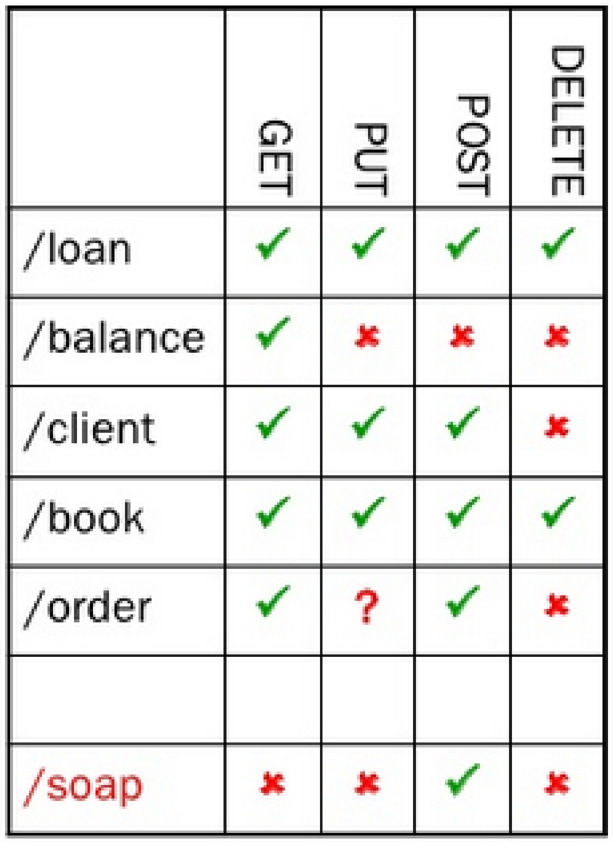
Hypermedia as the Engine of Application State (HATEOAS)



```
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

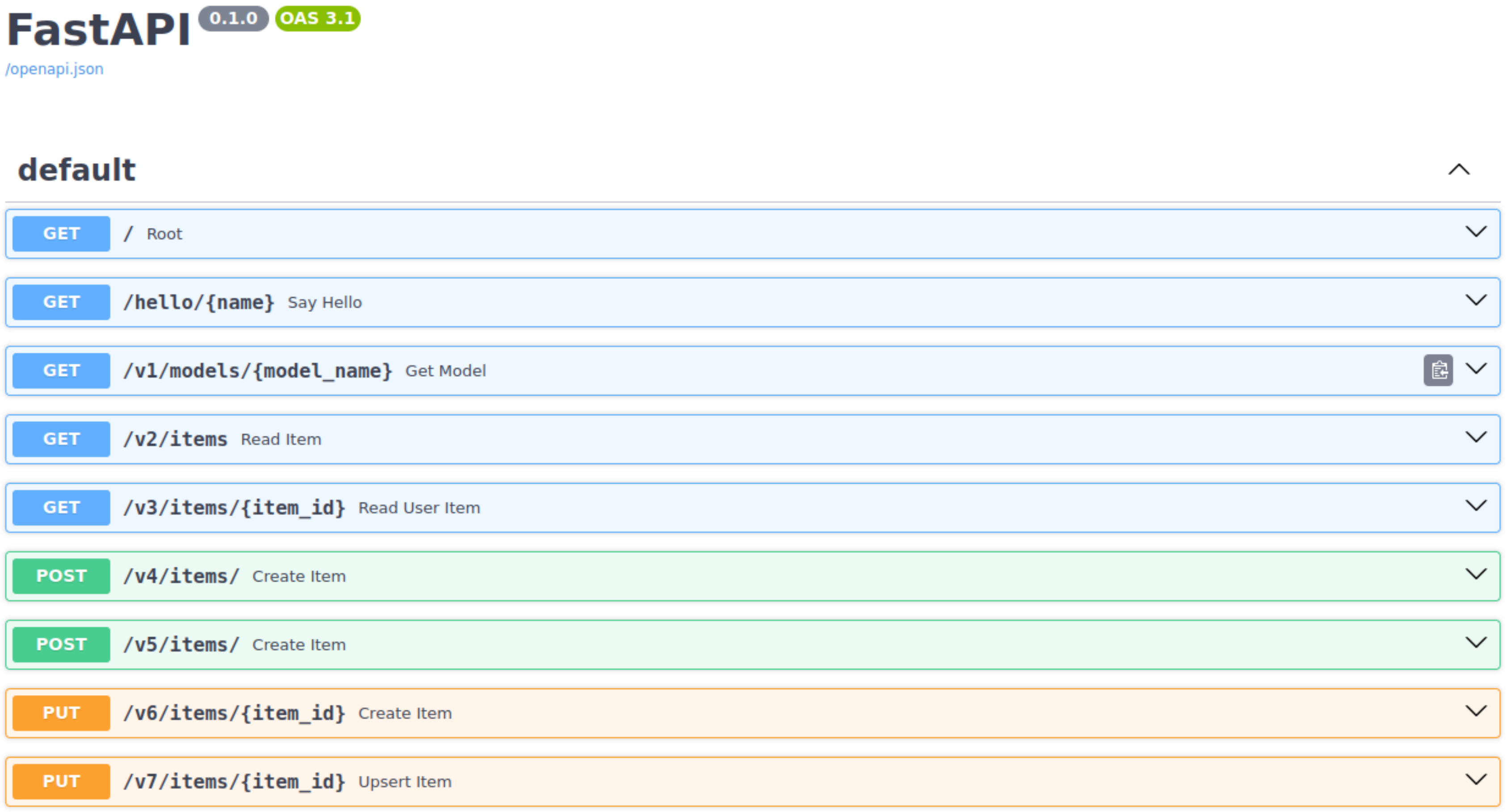
- Open attached distributed.py file
- You will need following packages:
  - pip3 install fastapi
- Start web server
  - Type following comment where the distributed.py is located
  - uvicorn distributed:app --reload

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



FastAPI 0.1.0 OAS 3.1

/openapi.json

default

GET / Root

GET /hello/{name} Say Hello

GET /v1/models/{model_name} Get Model

GET /v2/items Read Item

GET /v3/items/{item_id} Read User Item

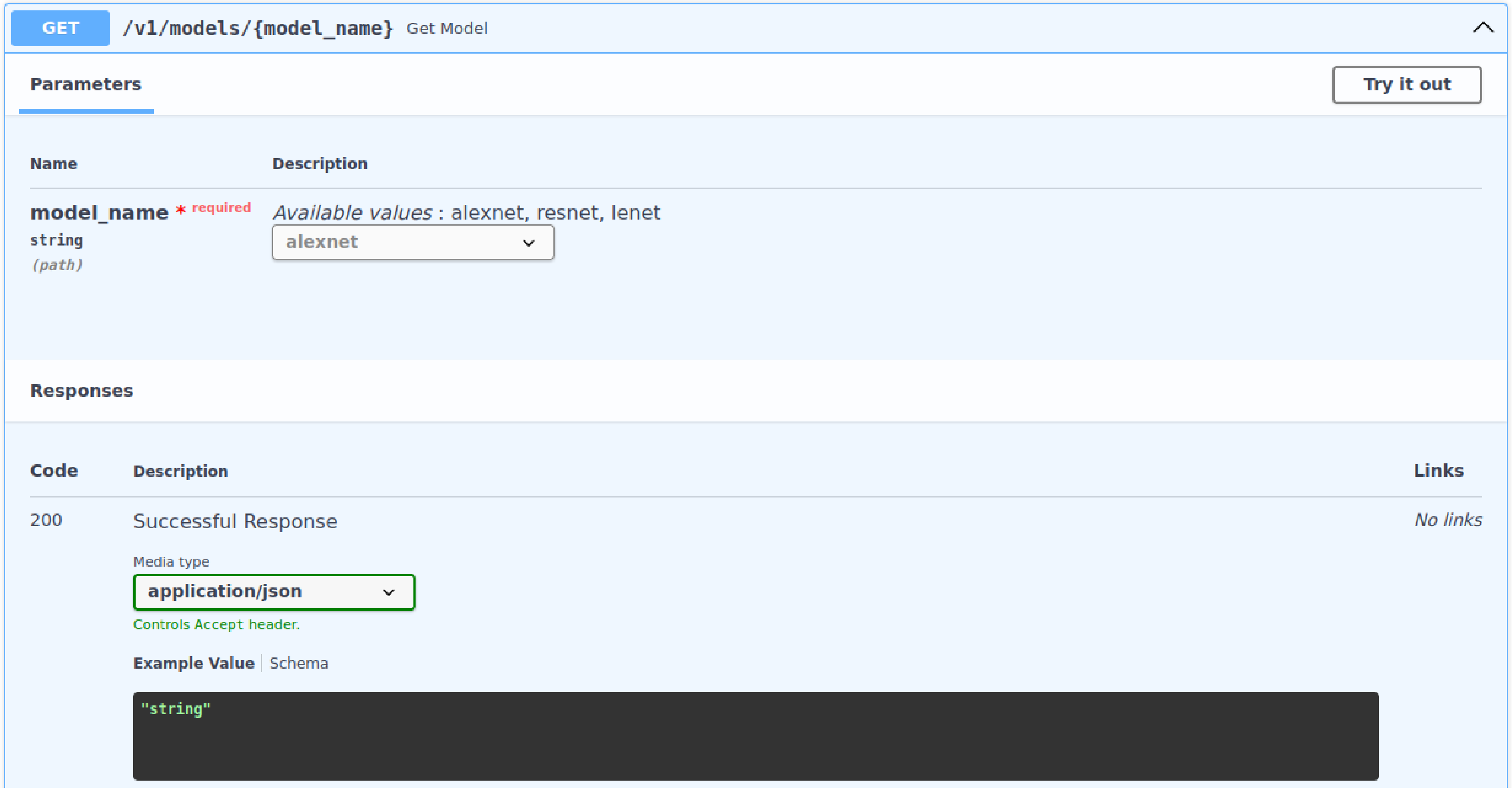
POST /v4/items/ Create Item

POST /v5/items/ Create Item

PUT /v6/items/{item_id} Create Item

PUT /v7/items/{item_id} Upsert Item

Task 0 – environment setup



/v1/models/{model_name} Get Model

Parameters

Name	Description
model_name * required string *(path)*	*Available values* : alexnet, resnet, lenet alexnet

Responses

Code	Description	Links
200	Successful Response Media type application/json Controls Accept header. **Example Value** | Schema	*No links*

```
"string"
```

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