CLEMSON UNIVERSITY

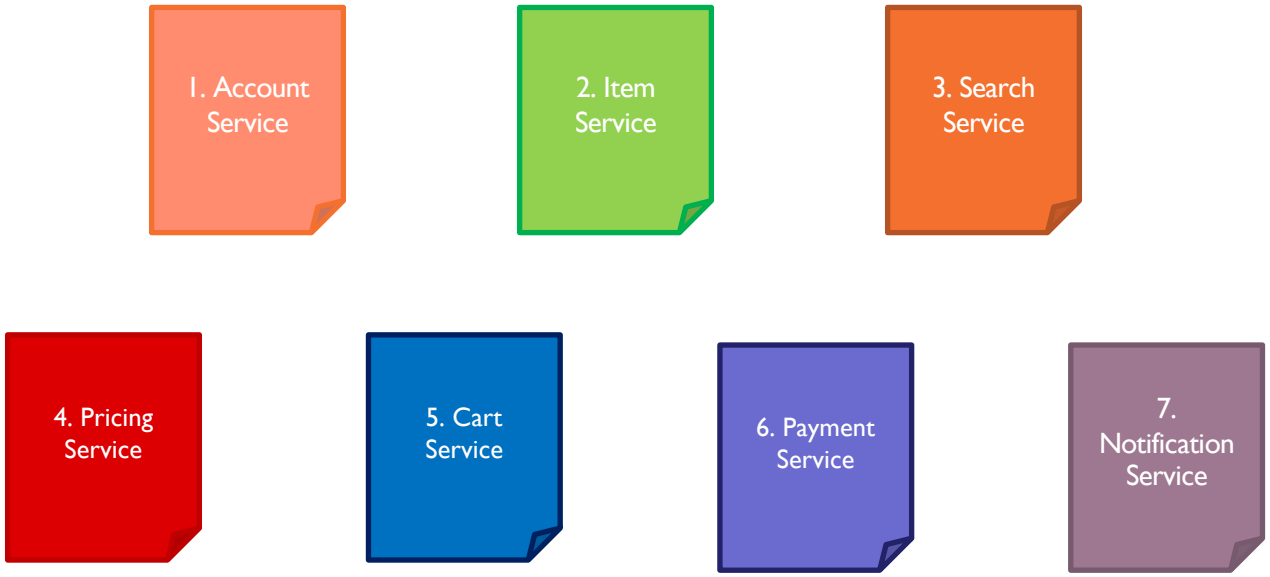
CPSC 3720
Lesson 14

Connie Taylor
Professor of Practice

School of COMPUTING

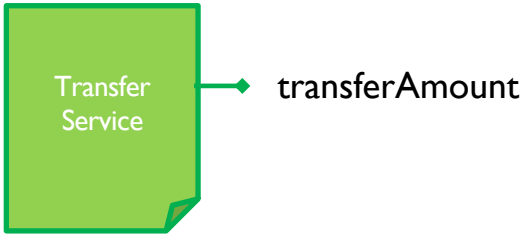
CUSports – Entities and API Contracts

CUSPORTS MODULES/COMPONENTS:



CUSPORTS BUSINESS FLOW:

API Contract



Input

```
{
  "fromAccount" : "X10001",
  "toAccount" : "X10002",
  "amount": 100.00,
  "currencyCode": "USD"
}
```

Output

```
{
  "transactionId" : "TXN10001",
  "transactionStatus" : "PENDING"
}
```

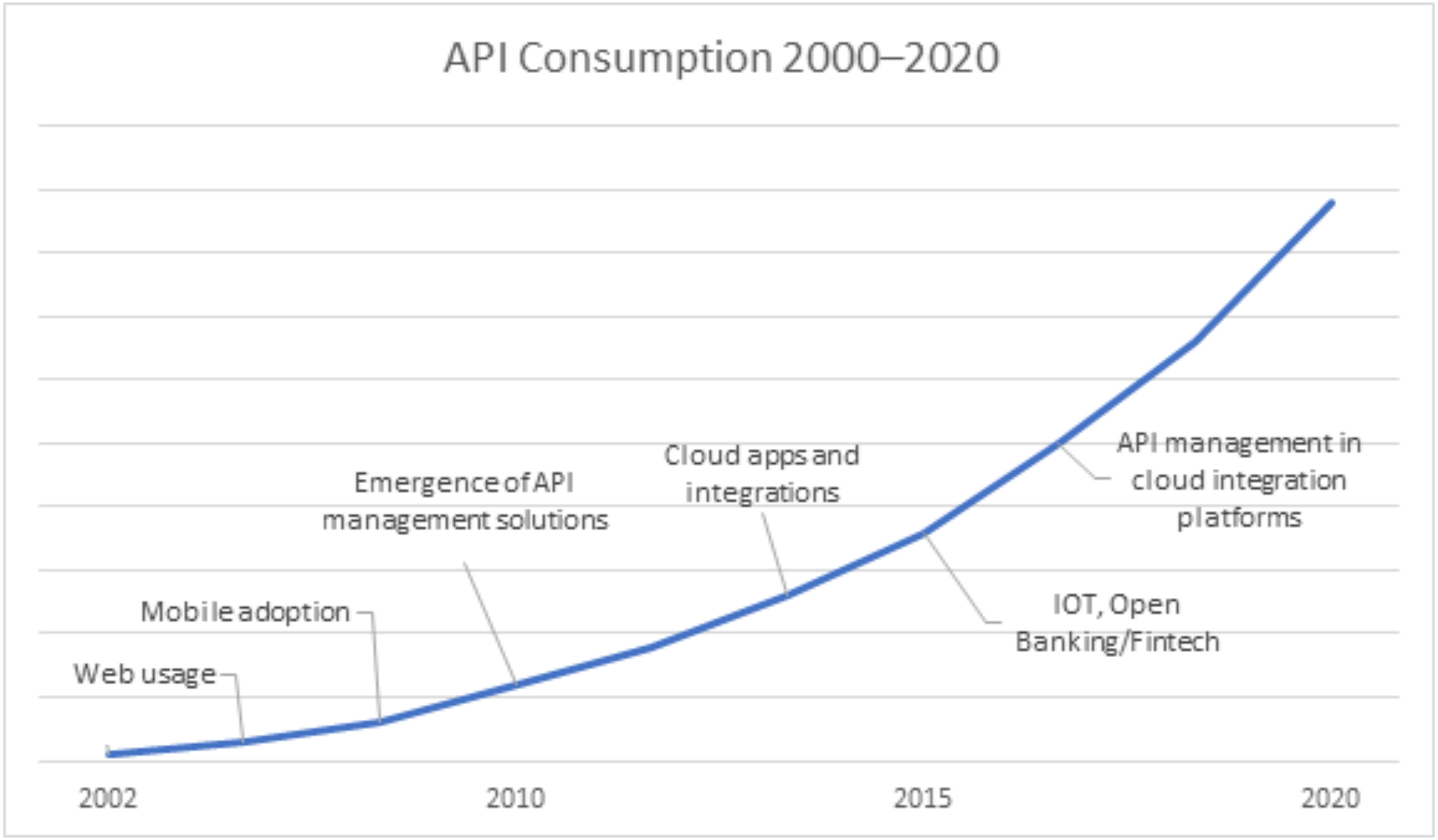
```
{
  "transactionId" : "TXN10001",
  "transactionStatus" : "REJECTED",
  "reasonCode": "EX101",
  "message": "Exceeded limit for the day"
}
```

API Contract

An **API** (Application Programming Interface) is best thought of as a contract provided by one piece of computer software to another.

A World of API Services

A World of API Services



A RESTful World of Services

- REST (Representational State Transfer) completely changed software engineering after 2000.

- This approach to developing web projects and services was defined by Roy Fielding, father of the HTTP specification, in his dissertation entitled "Architectural Styles and the Design of Network-based Software Architectures".

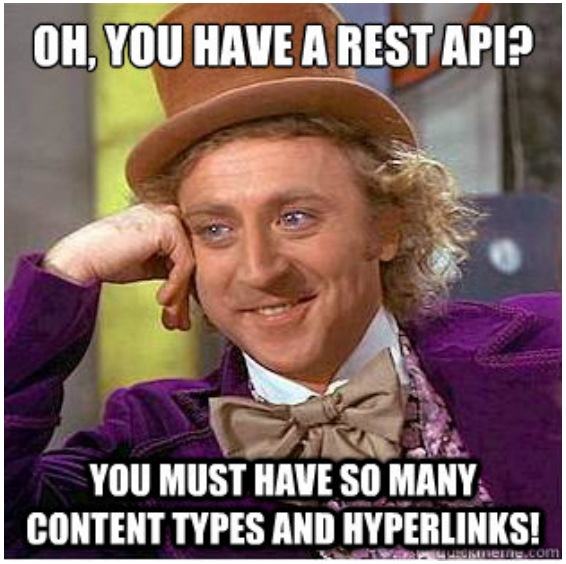
- REST today is the "be all-end all" in service app development.

- Most software projects and applications expose REST APIs for the creation of services based on this software. Twitter, YouTube, Facebook identification systems… hundreds of companies generate business thanks to REST.

- **So what is REST?**

What is REST?

- REST is an architecture style for designing **loosely coupled applications** over HTTP, that is often used in the development of web services.
- REST does not enforce any rule regarding how it should be implemented; it just puts high level design guidelines and leave you to think of your own implementation.



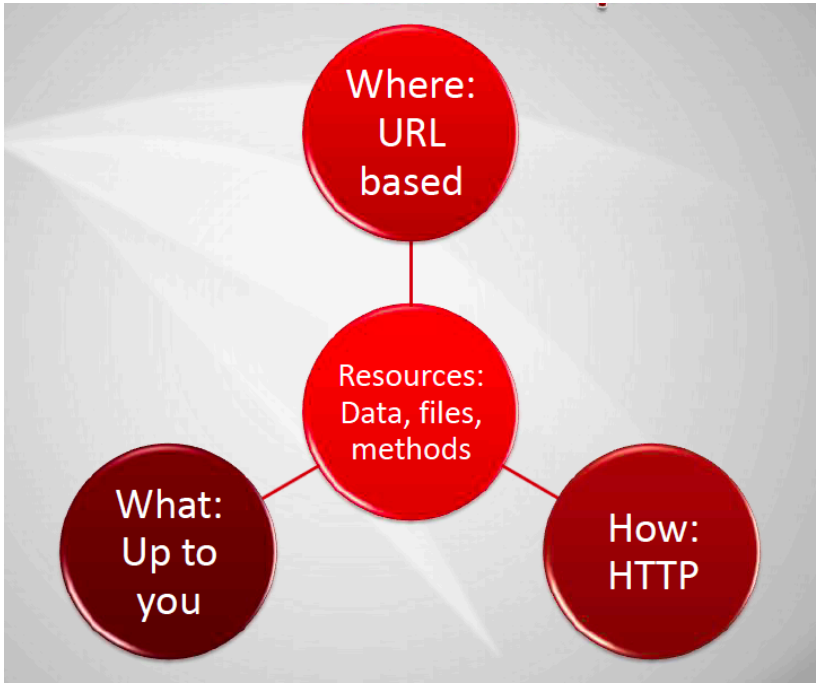
- REST is an increasingly popular alternative to other standard data exchange protocols such as SOAP (Simple Object Access Protocol), which are more complex.

What is REST?

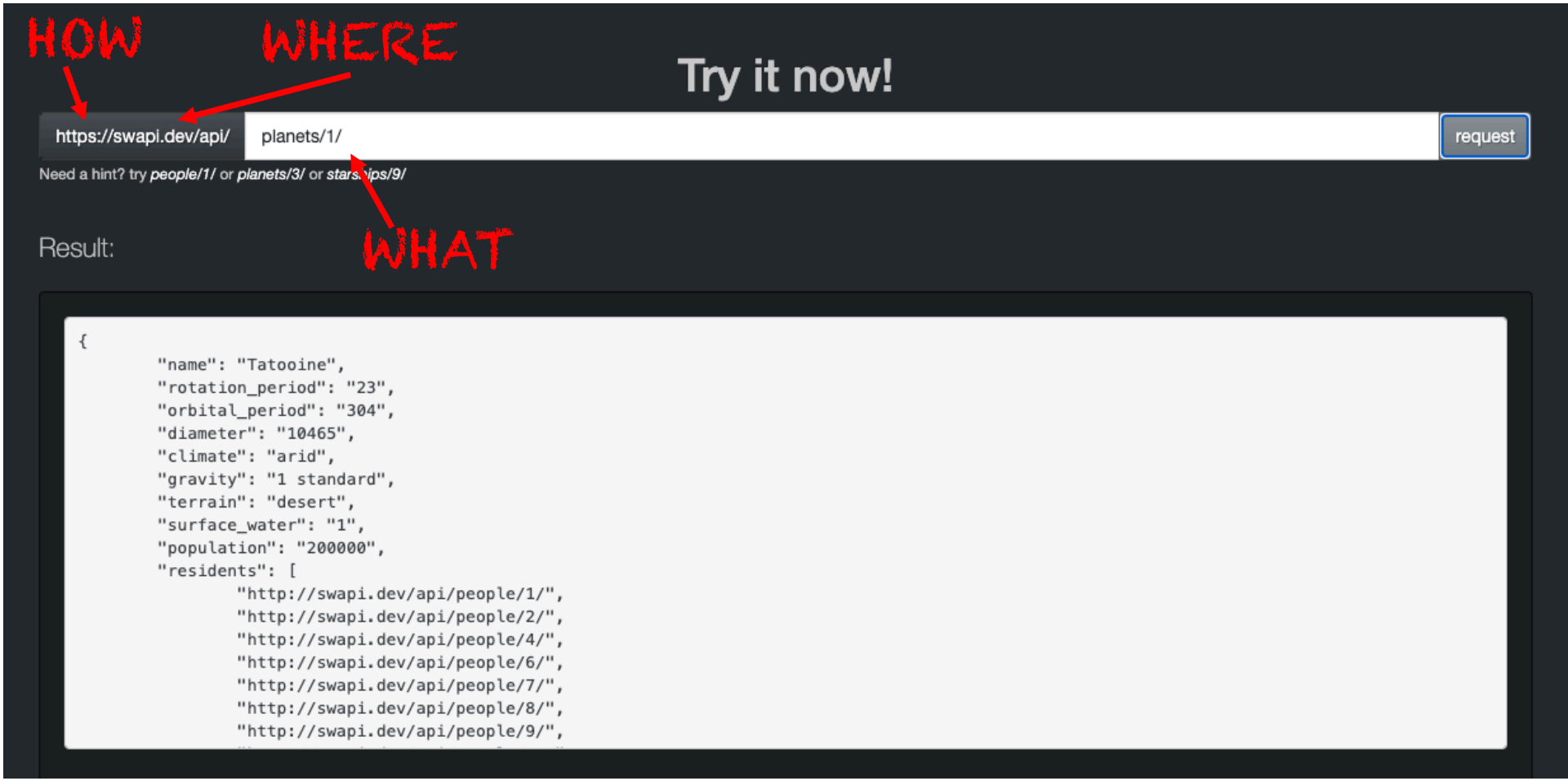
- In a REST based architecture everything is a **Resource**.
- A resource is accessed via a common interface based on the HTTP standard methods.
- You typically have a REST server which provides access to the resources and a REST client which accesses and modifies the REST resources.

REST Resources

- Resources are identified by global IDs (which are typically URIs or URLs).
- REST allows that resources have different representations, e.g., text, XML, JSON etc.



Example:
Star Wars API

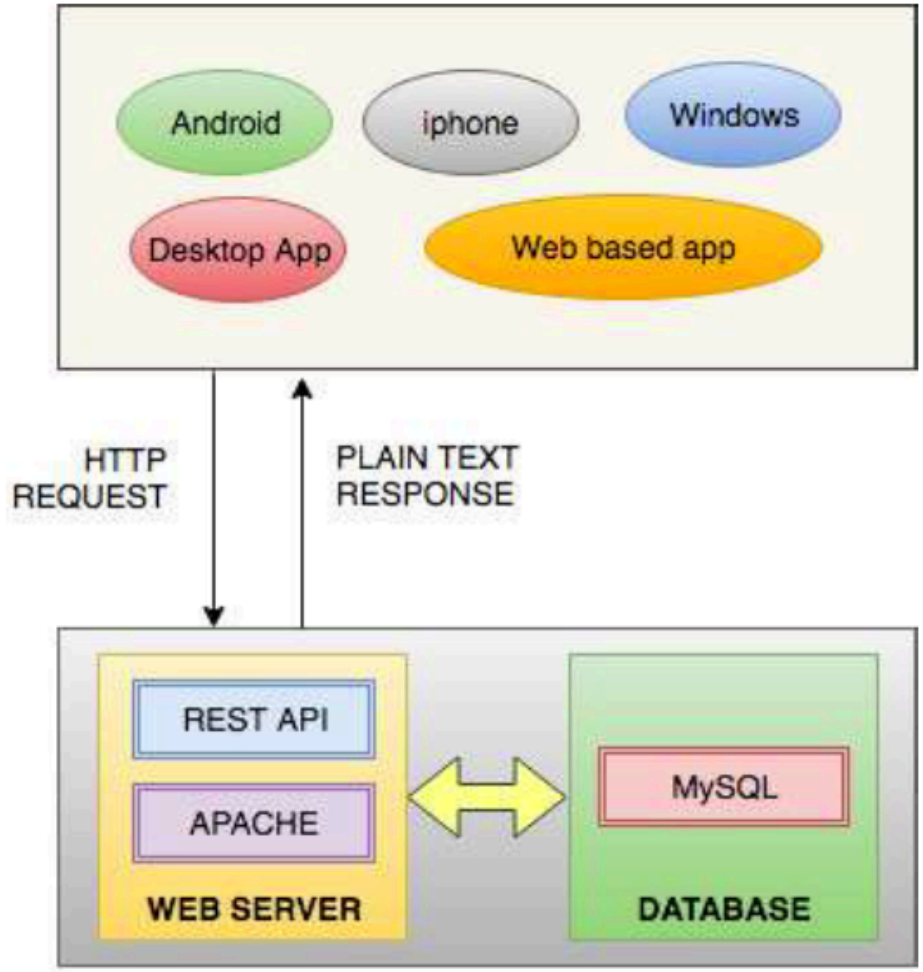


REST Methods

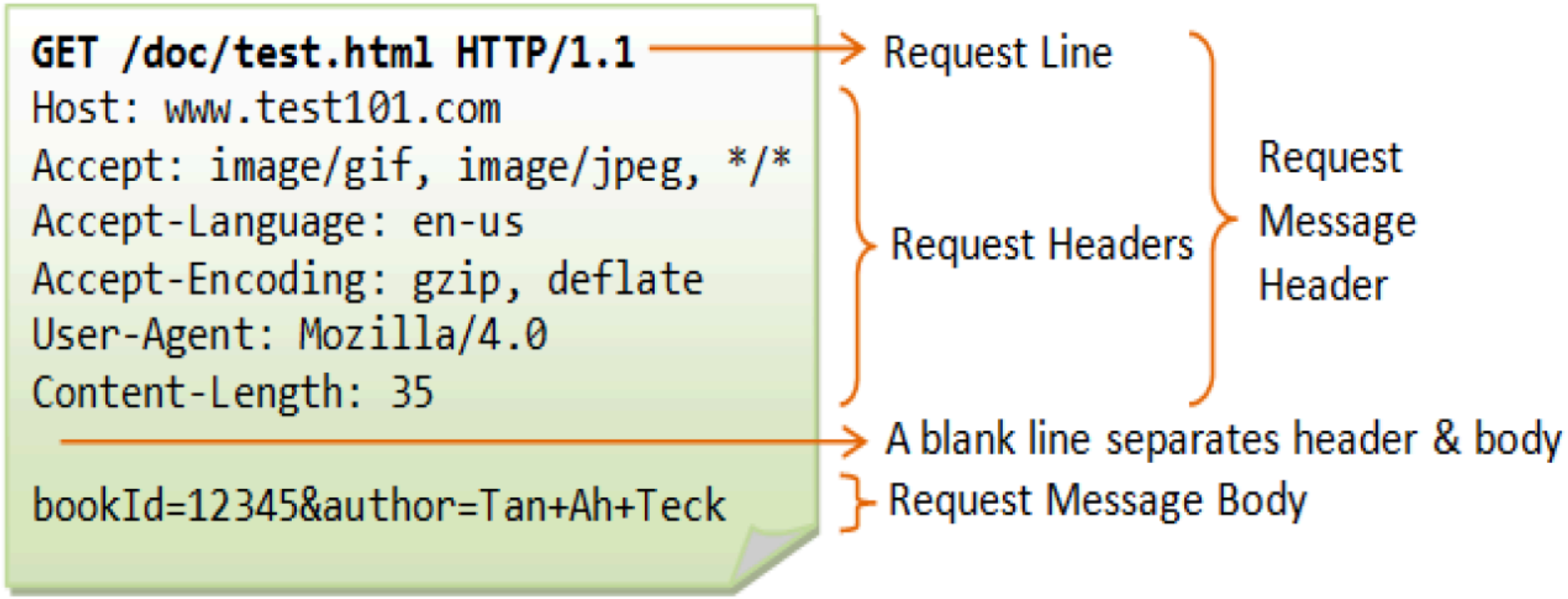
- HTTP Methods are a key cornerstone in REST.
- They define the action to be taken with a URL.

HTTP METHOD	CRUD OPERATION	DESCRIPTION
POST	INSERT	Adds to an existing resource
PUT	UPDATE	Overrides an existing resource
GET	SELECT	Fetches a resource
DELETE	DELETE	Deletes a resource

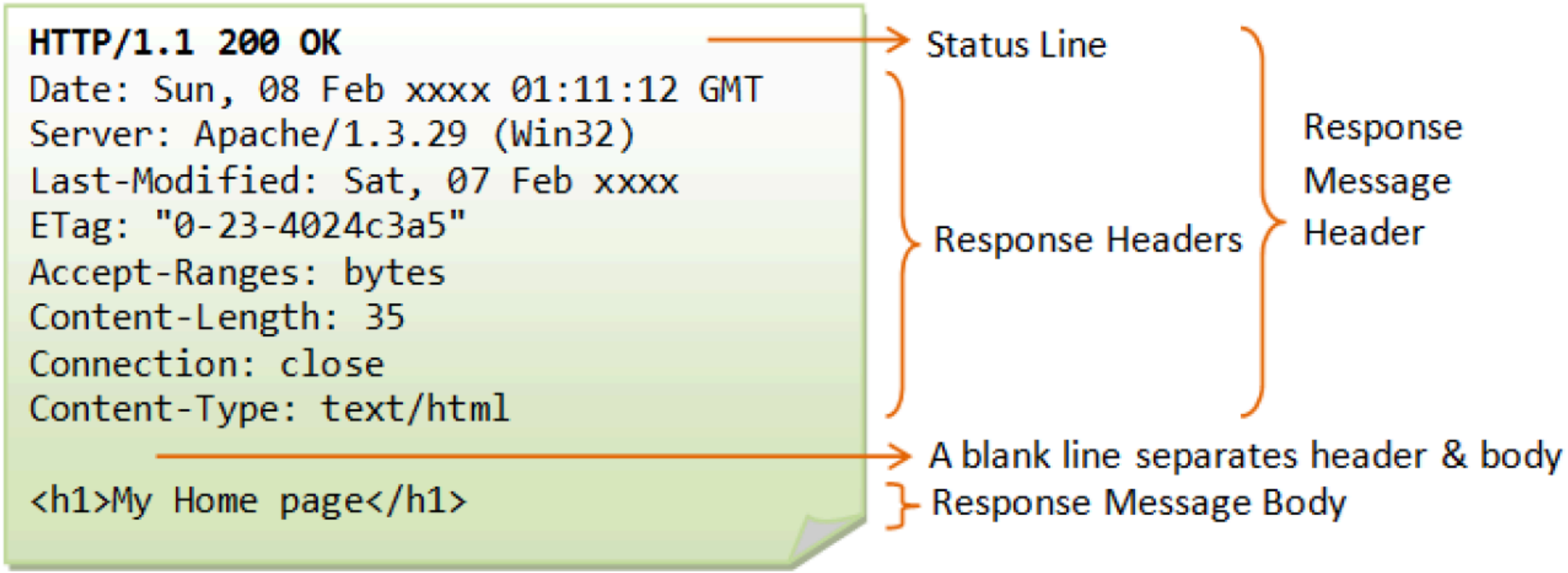
REST in Action



HTTP Request



HTTP Response



HTTP Response Codes	Description
1xx	Informational Codes
2xx	Successful Codes
3xx	Redirection Codes
4xx	Client Error Codes
5xx	Server Error Codes

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

REST Architecture Constraints

- REST defines 6 architectural constraints which make any web service a true RESTful API.
  - Client–server
  - Stateless
  - Cacheable
  - Uniform interface
  - Layered system
  - Code on demand (optional)

REST Architecture Constraints: Client-Server

- Essentially means that the client and server application MUST be able to evolve separately without any dependency on each other.
- A client should know only resource URIs, and that’s all.
- Servers and clients may also be replaced and developed independently, as long as the interface between them is not altered.

REST Architecture Constraints: Stateless

- Each request from client to server must contain all information necessary to understand the request
- The server will treat every request as new – no session or history.
- If the client application needs to be a stateful application for the end-user, then each request from the client should contain all the information necessary to service the request – including authentication and authorization details.

REST Architecture Constraints: Cacheable

- The caching of data and responses is of utmost importance wherever they are applicable/possible.
- Caching enables performance improvements on the client and better scalability for the server. HOW?
- Caching shall be applied to resources when you can, and the resources MUST declare themselves cacheable.
- Caching can be implemented on the client or server.

REST Architecture Constraints: Uniform

- By applying the principal of generality to the component interface, the architecture is simplified and the visibility of interactions is improved.
- In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components.
- Once a developer becomes familiar with one of your APIs, he should be able to follow a similar approach for other APIs.
- REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

REST Architecture Constraints: Layered System

- The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.

- You can deploy the APIs on server A, and store data on server B and authenticate requests in Server C, for example. A client cannot ordinarily tell whether it is connected directly to the end server or an intermediary along the way.

REST Architecture Constraints: Code on Demand

- (Optional) REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

JSON

- **J**ava**S**cript **O**bject **N**otation
- Used to format data
- Commonly used in Web as a vehicle to describe data being sent between systems

JSON Example

- Despite the name, JSON is a (mostly) language-independent way of specifying objects as name-value pairs

- Example (http://secretgeek.net/json_3mins.asp):
  -
```
{"skillz": {
    "web":[
        { "name": "html",
          "years": 5
        },
        { "name": "css",
          "years": 3
        }]
    "database":[
        { "name": "sql",
          "years": 7
        }]
}}
```

JSON Syntax

- An *object* is an unordered set of name/value pairs
  - The pairs are enclosed within braces, { }
  - There is a colon between the name and the value
  - Pairs are separated by commas
  - Example: { "name": "html", "years": 5 }

- An *array* is an ordered collection of values
  - The values are enclosed within brackets, [ ]
  - Values are separated by commas
  - Example: ["html", ”xml", "css"]

JSON Syntax Cont'd

- A *value* can be: A string, a number, true, false, null, an object, or an array
  - Values can be nested
- *Strings* are enclosed in double quotes, and can contain the usual assortment of escaped characters
- *Numbers* have the usual C/C++/Java syntax, including exponential (E) notation
  - All numbers are decimal--no octal or hexadecimal
- *Whitespace* can be used between any pair of tokens

Mapping between JSON and Java entities

JSON	Java
string	java.lang.String
number	java.lang.Number
true | false	java.lang.Boolean
null	null
array	java.util.List
object	java.util.Map

Homework

Getting started with APIs and Postman:

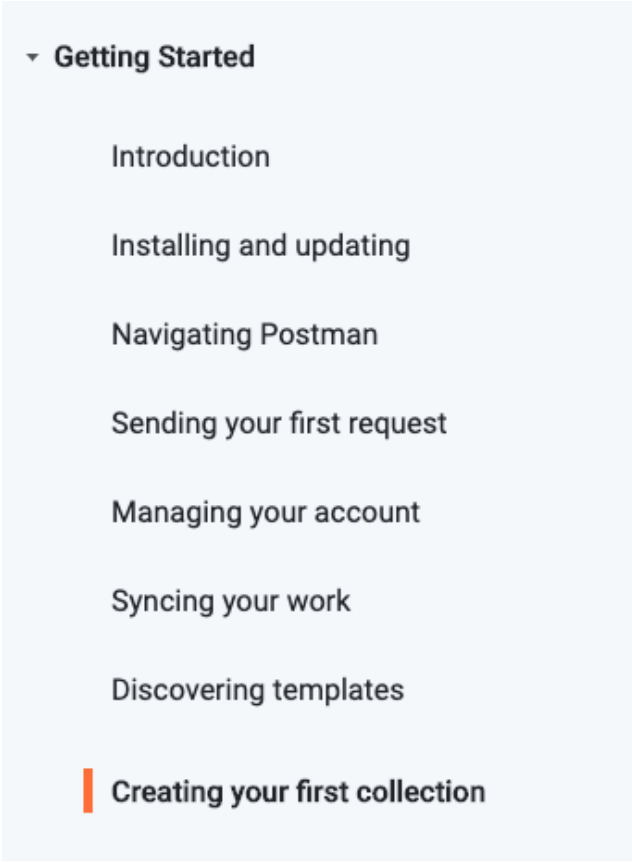
https://learning.postman.com/docs/getting-started/introduction/ (will include install instructions)

Go through Creating your first Collection

https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/mocking-with-examples/

Show your first Collection and Mock example using a screenshot from the web page (with your name visible).

Turn it in on Canvas EOD Tuesday; 5% of Project Grade



Next Quiz

Quiz 2 due 11pm Wed Sept 23 (will open on Sept 21)

The questions are from **Lessons 7-10**. You cannot collaborate with your classmates about this quiz prior to, or during the taking of the quiz. **NOTE:** Once you start the quiz you will have only 15 minutes to complete it. Be sure to review the lessons before you begin.