Designing APIs & REST APIs

Thomas Fritz

Many thanks to Reid Holmes and Elisa Baniassad

Agenda

1. API Design
2. REST API Design
3. Quiz and more

Note:

- *Next time Midterm (10.04.2024) ; be there on time*
- *Make sure to go over the information in the email you'll get today*
- *April 17 Tarek Alakmeh will teach the lecture on Modular Design & Design Principles*

Examinable skills

By the end of this class should be able to…

- Describe the importance and characteristics of a good API
- Describe the basic characteristics of REST and a REST API
- Describe & apply principles for designing a good (REST) API
- Critique the design of a (REST) API

API Design

Learning Objectives

Be able to:

- Describe the importance and characteristics of a good API

Recap: Modularity

"A complex system can be managed by dividing it up into smaller pieces and looking at each one separately. When the complexity of one of the elements crosses a certain threshold, that complexity can be isolated by defining a separate abstraction that has a simple interface. The abstraction hides the complexity of the element: the **interface indicates how the element interacts** with the larger system"

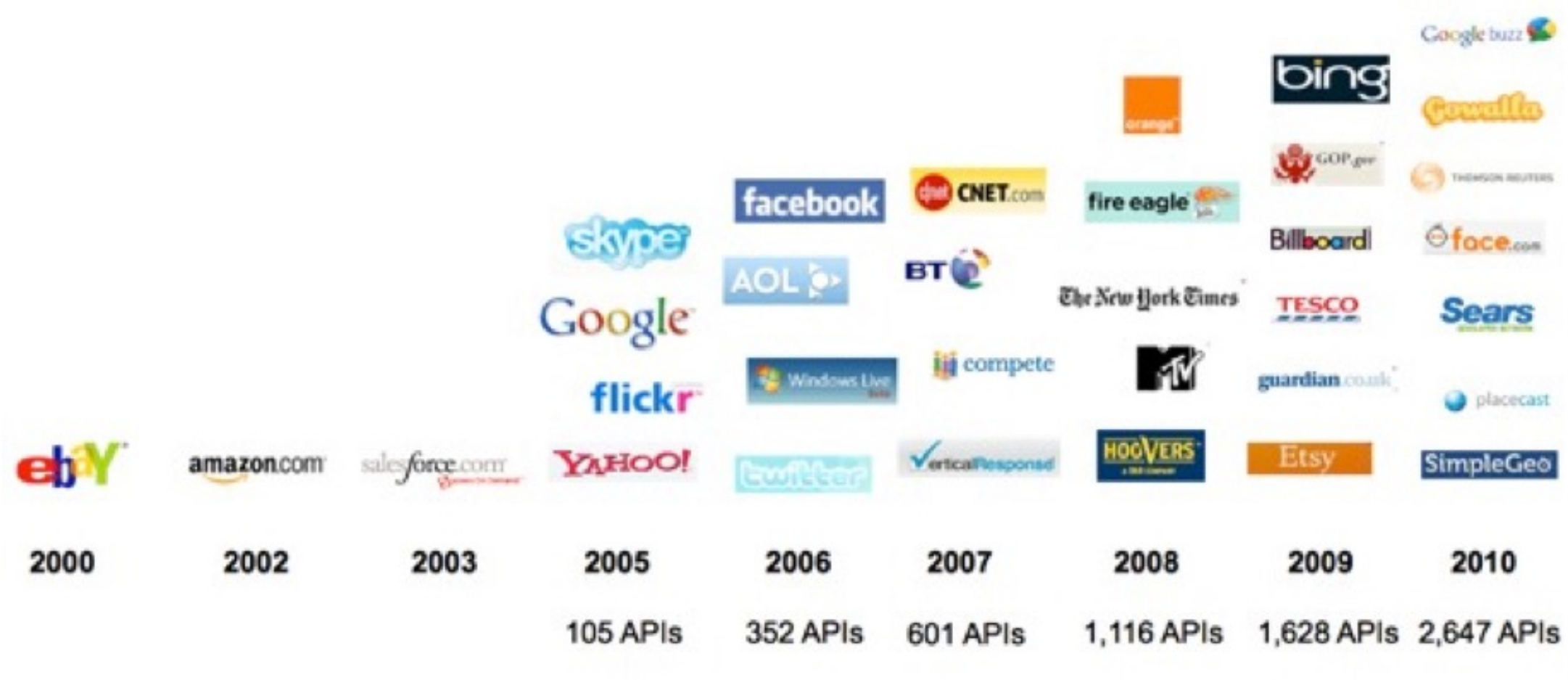
- Baldwin and Clarke 2000

Application Programming Interface

- APIs provide a predictable means for programmatic interaction
- APIs specify the protocols, names, parameters, and data formats for interacting with software components
- APIs hide implementation details
- Used in many different contexts and different levels:

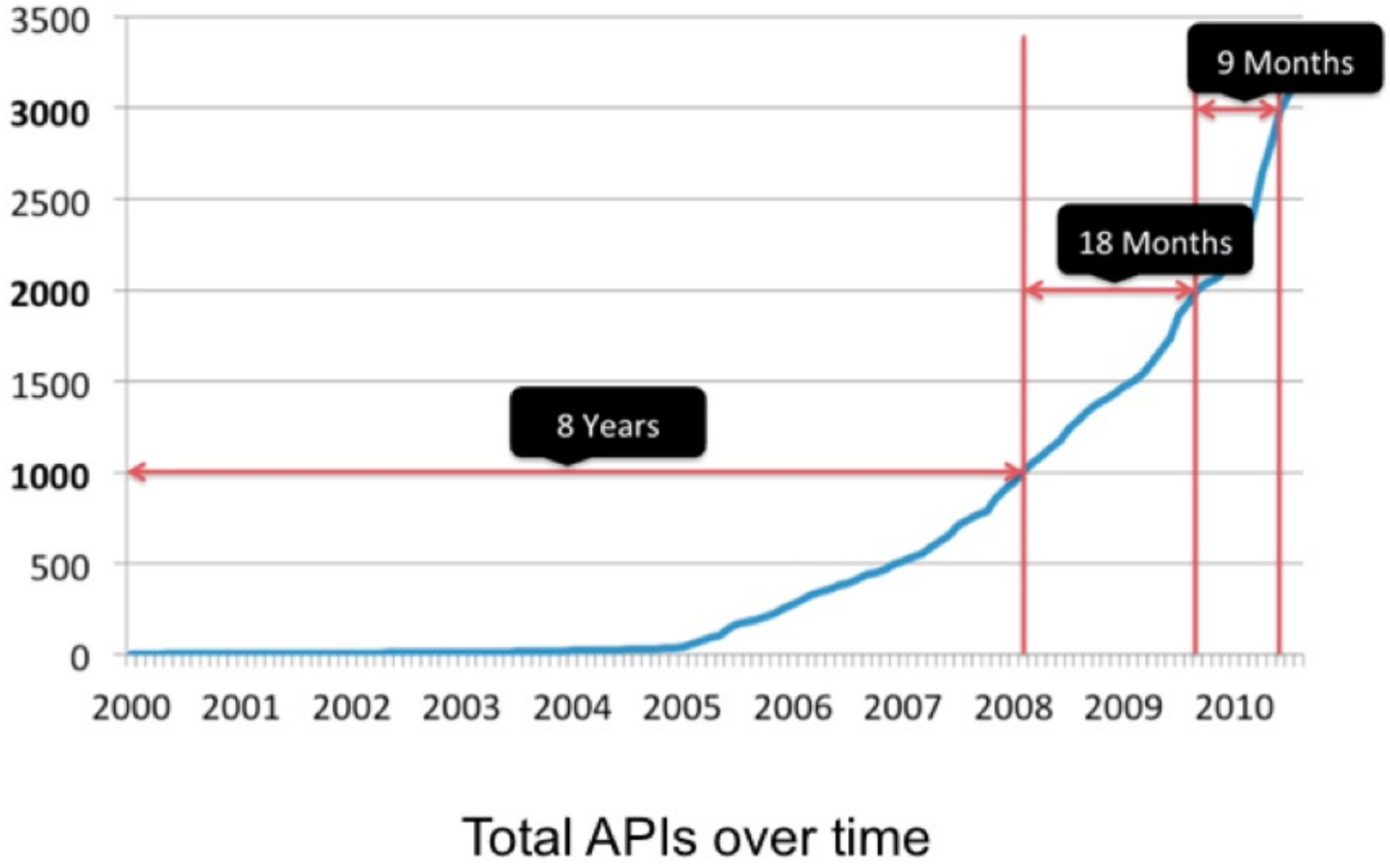
```
public List<Organization> getOrgs() <— API
GET /organizations?fields=id,name   <— REST API
```

Open API growth



from programmable web

API growth



Total APIs over time

from John Musser

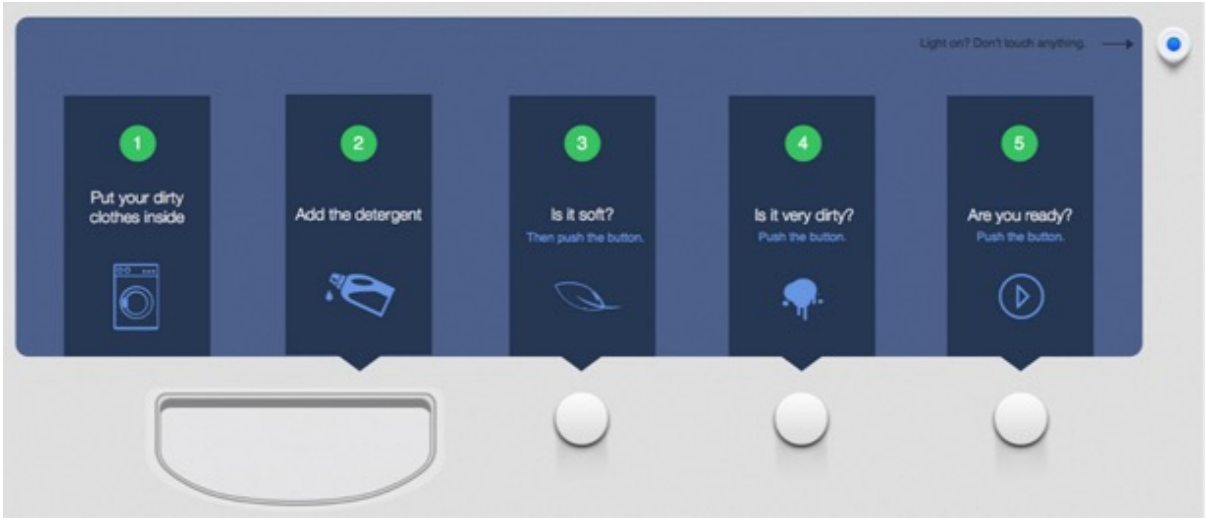
APIs

- APIs provide a lot of value
- Once clients start to use them, they are difficult to change

  “APIs are forever”
- Spend time designing APIs ‘right’

What’s important when designing an interface?

What's important when designing an interface?

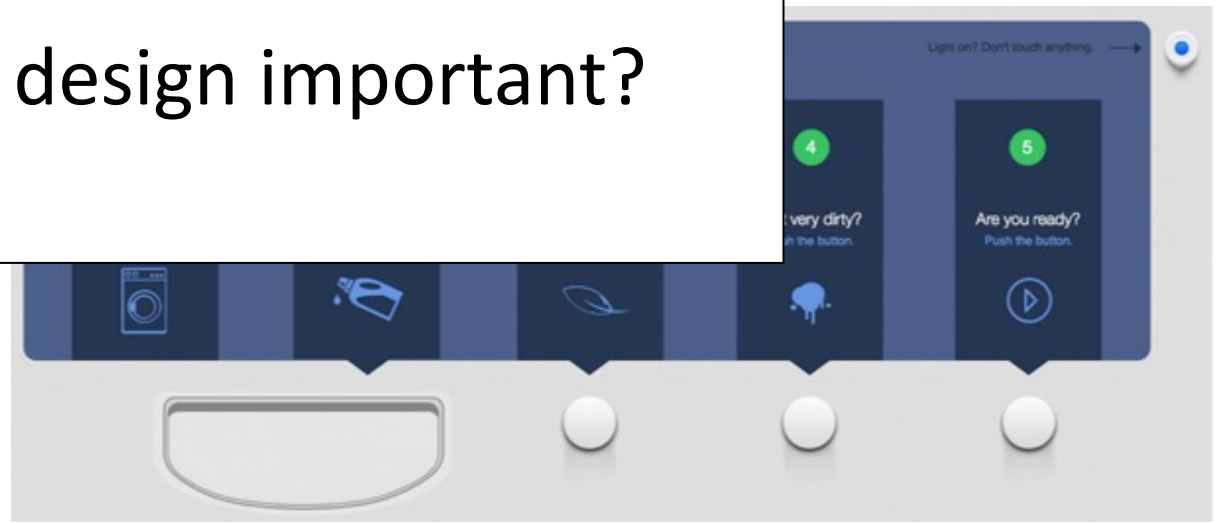


http://www.core77.com/posts/26451

What's important when designing an interface?

Is this similar/different for an API?

Why is a good API design important?



http://www.core77.com/posts/26451

Usability principles (e.g. D. Norman)

Good visibility

- possible actions and states must be exposed

Good conceptual model

- helpful, consistent, and complete abstractions clarify the correct model of the system

Good mapping between actions and results

- natural

Good feedback about results of actions

- full and continuous

Characteristics of a good API (Josh Bloch)

- easy to learn
- easy to use, even without documentation
- hard to misuse
- easy to read and maintain code that uses it
- sufficiently powerful to satisfy requirements
- easy to understand
- appropriate to audience

https://www.youtube.com/watch?v=aAb7hSCtvGw

Principles for designing good APIs (J. Bloch)

Do one thing and do it well

- Functionality should be easy to explain
- Users should not be surprised by API behavior

APIs should be as small as possible but no smaller

- When in doubt, leave it out. APIs are forever

Implementation should not impact API

- Leaking details into API is a fundamental mistake

Minimize access, maximize information hiding

Names matter

- should be largely self-explanatory

Documentation matters

And there are more…

- Avoid long parameter lists
- Return descriptive objects
- Avoid exceptional returns
- Handle exceptional circumstances
- Favor immutability
- Favor private classes, fields and methods
- …

Designing an API

- Address two main questions:
  1. What is the goal (use cases) of this API?
  2. Who will be using this API
- Start with one-page API specification
- talk to others/users about it and iterate over API
- write multiple clients before release (rule of 3)
- document
- API design is tough, not a solitary activity, and perfection is unachievable, but try anyway
- Apply principles / guidelines

Osmdroid map API example

OSMDroid: **open source** library for Android apps to integrate interactive maps using OpenStreetMap data (similar to Google Maps)

```
// set the way the tiles in the map (MapView) are presented
// MAPNIK refers to default style with roads, buidlings,...
map.setTileSource(TileSourceFactory.MAPNIK);
...
// add default zoom buttons
map.setBuiltInZoomControls(true);
// add ability to zoom with 2 fingers
map.setMultiTouchControls(true);
...
```

https://javadoc.io/doc/org.osmdroid/osmdroid-android/latest/index.html

https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library-(Java)

Class Exercise – Designing an API [bit.ly/3Vz8mSw]

In groups (approx. 10 to 15mins):

- create API design for a set of features of a “WhatsApp” like messaging app
- For simplicity, the app has 3 classes: MsgApp, User, Message (the msgapp has a list of users, each user has a list of contacts he/she can msg to and for each contact, the user contains a list of messages)
- Design an API for the class MsgApp and User that allows you to
  - register and update a user with name, age, image, description,…
  - search for users
  - Send messages to a user
- How to send a message to users who are older than 25 and called Tim?

Possible questions to reflect on

1. What is technical debt?
   → concept in SE that reflects cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer
2. What are the implications of API changes (or deletions) and how do you assess their cost? What is the procedure for changing APIs?
3. Why not make everything public?
4. Do we have to ensure that old versions still work?
5. When do you define an API, i.e. can you define it after the system is implemented?
6. Are private methods in Java also considered part of the API?
7. Is an API the same as a Java interface?

   → API (in Java) is usually used to distinguish the public part from private part

REST API Design

Learning Objectives

Be able to:

- Describe the basic characteristics of REST and a REST API
- Describe & apply principles for designing a good (REST) API
- Critique the design of a (REST) API

API calls so far

- An object calls the method of another object
  - e.g., User u = new User();
    user.setName(“Gina”);

- How to do this in a distributed setting, e.g. with a client and a server?
  - Remote Procedure Call (RPC)
  - REST – Representational State Transfer

REST

REpresentational **S**tate **T**ransfer

- Invented by Roy Fielding during his PhD

- "A network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."

REST – REpresentational State Transfer

Stateless (usually based on HTTP)

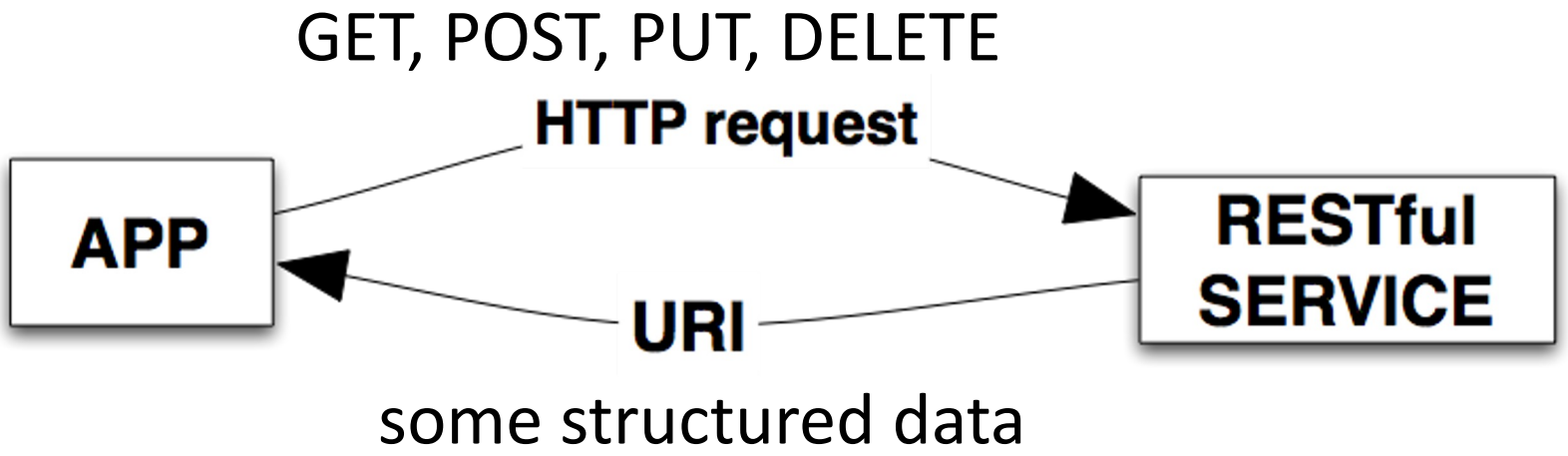
- every request is self-contained
- clients maintain their own state; server does not retain state about clients

Resource-Oriented

- key elements of any RESTful service are resources
- every resource is identified by a “Uniform Resource Identifier” (URI)
- Server maintains resource structure and passes back relevant links to related resources (connectedness)

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

REST and its methods



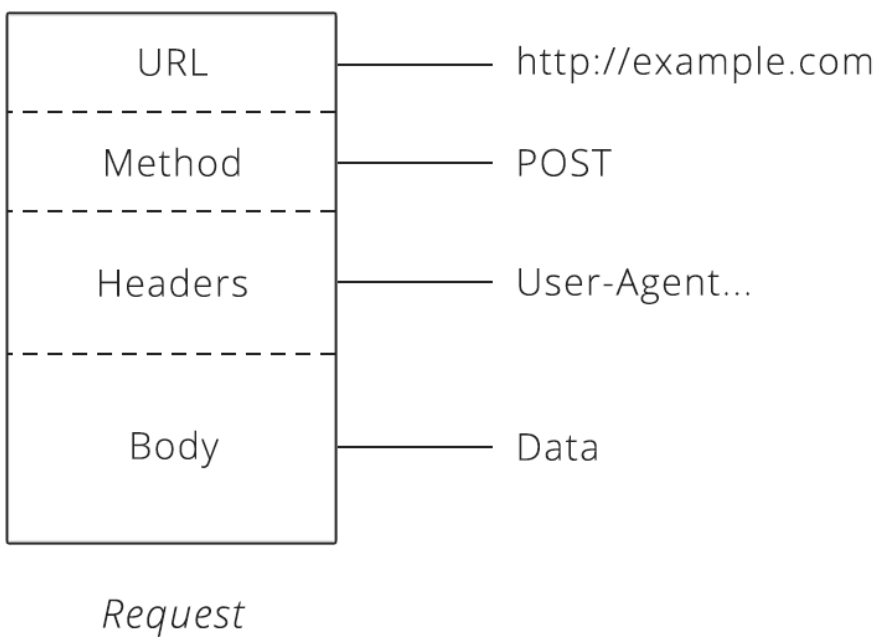
Twitter example:

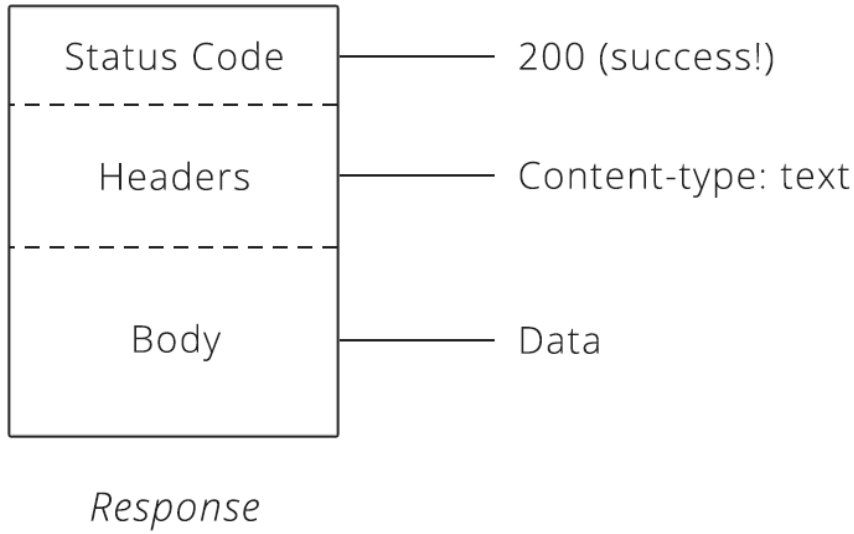
POST https://api.twitter.com/1.1/statuses/update.json?status=Maybe%20he%27ll%20finally%20find%20his%20keys.%20%23peterfalk

HTTP methods (subset) to manipulate resources

- GET: get a resource
- PUT: update an existing resource
- POST: create a resource
- DELETE: delete a resource

REST – an example





Copied from https://restful.io/an-introduction-to-api-s-cee90581ca1b when it was still available.

REST Response Representations

- For the response body there are many options:
  - Plain Text: Hard to parse & not self descriptive; not great for REST services
  - XML: <response><code>200</code>…</response>: more verbose; attributes can be handy
  - JSON: {code: 200}: Smaller; easy to write by hand (and read); can be treated as JavaScript objects

Class Exercise Part I - REST API [bit.ly/49fzbOO]

- In groups (total of ~15mins for both parts)
- Go to https://transport.opendata.ch/docs.html

  Swiss public transport API

- **Task: fastest way to Zurich HB**

  I'm standing in front of the IFI building and want to go downtown. How can I use the API to find the closest and fastest **tram** to bring me to Zurich HB?

- **Good / bad API and why?**

Class Exercise - REST API cont'd [bit.ly/49fzbOO]

- Now try the same with Translink (**Vancouver**)
- https://developer.translink.ca/ServicesRtti/ApiReference

- **Task: fastest way downtown**

  I'm standing at Burrard and 10th (crossing of two streets) and want to go downtown. How can I use TransLink API to find the closest and fastest bus to bring me downtown?

- **Good / bad API and why?**
- **What's the difference to the Swiss one?**

Designing REST APIs

Same principles as for API Design apply!

+ REST specific principles

Designing a REST API - Scenario

- Let's design a REST API for dogs



- How to design it for getting specific dogs, getting all dogs, checking health, getting location, feeding dogs, getting location,….

scenario taken from Brian Mulloy's slides
pictures from
http://www.apartmentguide.com/blog/how-to-walk-a-dog-in-your-apartment-community/
https://i.ytimg.com/vi/opKg3fyqWt4/hqdefault.jpg
https://i.ytimg.com/vi/opKg3fyqWt4/hqdefault.jpg

Designing a REST API for dogs

/getAllDogs

/getDog

/getRedDogs

/checkHealth

/getLocation

/foodNeeded

/getDogOwner

/changeDogOwner

….

Designing a REST API for dogs

/getAllDogs

/getDog

/getRedDogs

/checkHealth

/getLocation

/foodNeeded

/getDogOwner

/changeDogOwner

….

/newDogForOwner

/saveParentDogs

/giveMedication

/getVaccinationsOfDog

/getDogAge

/verifyDogRegistration

….

Designing a REST API for dogs

- Keep it simple!
- Only two base types of resources
  - Collection resources **`/dogs`**
  - Instance resources **`/dogs/1234`**

Designing a REST API for dogs (if time allows)

- Keep it simple!
- Only two base types of resources
  - Collection resources **`/dogs`**
  - Instance resources **`/dogs/1234`**

Note:

- *Questions neglect Json/XML/text that's passed*

Designing a REST API for dogs

- Keep it simple!
- Only two base types of resources
  - Collection resources **`/dogs`**
  - Instance resources **`/dogs/1234`**

Resource	POST create	GET read	PUT update	DELETE delete
`/dogs`		list dogs		
`/dogs/1234`		show Bo		

Designing a REST API for dogs

- Keep it simple!
- Only two base types of resources
  - Collection resources **`/dogs`**
  - Instance resources **`/dogs/1234`**

Resource	POST create	GET read	PUT update	DELETE delete
`/dogs`	create a new dog		bulk update dogs	
`/dogs/1234`	error		if exists update Bo if not error	

Designing a REST API for dogs

- Keep it simple!
- Only two base types of resources
  - Collection resources **`/dogs`**
  - Instance resources **`/dogs/1234`**

Resource	POST create	GET read	PUT update	DELETE delete
`/dogs`	create a new dog	list dogs	bulk update dogs	delete all dogs
`/dogs/1234`	error	show Bo	if exists update Bo if not error	delete Bo

Designing a good REST API

- Nouns are good (for resources), verbs are bad
- Plurals are better `/dogs vs /dog`

- Examples
  - Twitter resources include tweets, users, followers (http://twitter.com/user/tweets)
  - Facebook resources include users, friends, posts

Designing a good REST API

- Association
  - How to get all dogs of owner 3426?

```
GET owners/3426/dogs
```

- Complex variations behind “?”

```
/dogs?color=red&state=running
```

Designing REST APIs

- Error handling is important
  - use HTTP status codes as much as possible
    - 200 – OK
    - 400 - Bad request (invalid…)
    - 401 – Unauthorized
    - 500 - Internal Server Error - API
  - actual message should be as descriptive as possible and meaningful: help the developer to understand what's wrong

Designing REST APIs

- Versioning (why would you want this?)
  - various options, use one

  `/2017-01-01/dogs, /v1/dogs, ?v=1.0`
- Specify what's needed

  `/john.smith/friends?fields=id,name,picture`
- Specify return format, e.g. **`/dogs.json`**
- For non-resource-y stuff: use verbs

  `/search?q=fluffy+fur`

Class Exercise – REST API Design [bit.ly/3PFuO8R] (depending on time)

In groups

- create & critique design for a set of features of a “WhatsApp” like messaging app
- Design a REST API that allows you to
  - create and update a user profile with name, image, description,…
  - search for users
  - Send messages to a user
  - [depending on time] find messages for a specific user and day
- What are resources & responses for each method?
- How to create a user profile?
- How to send message to users who are older than 25 and called Tim?

Quiz & more

Additional Resources – REST API

- Joshua Bloch's "How to Design a Good API and Why it Matters" (https://www.youtube.com/watch?v=aAb7hSCtvGw)
- David Koelle's "Golden Rules of API Design"
- Brian Mulloy's "RESTful API Design"
- A few REST APIs with good documentation:
  - search API of twitter: https://dev.twitter.com/rest/public/search
  - github API: https://developer.github.com/v3/
- Multiple approaches to build and test (REST) APIs
  - Postman (https://www.postman.com/)
  - Insomnia (https://insomnia.rest/)

Additional Resources

Difference between patch, put and post: https://dzone.com/articles/the-simple-guide-to-http-verbs-patch-put-and-post

Difference REST and HTTP: http://restcookbook.com/Miscellaneous/rest-and-http/

Midterm

	KOH-B-10	KOL-E-18 (NTA)
10:00 - 10:15	Vorbereitung	Vorbereitung
10:15 - 10:30		
10:30 - 10:45	*Einlass*	*Einlass*
10:45 - 11:00		
11:00 - 11:15	**Prüfung**	**Nachteilsausgleich**
11:15 - 11:30		
11:30 - 11:45		
11:45 - 12:00		
12.00 - 12.15	*Auslass*	
12.15 - 12.30		
12.30 - 12.45		*Auslass*
12.45 - 13.00		

- Be on time
- Material will be all that was covered so far (also readings and videos), and including todays lecture

Quiz

1. When designing an API, which practices are good?
    A. Using descriptive objects
    B. Using long parameter lists
    C. Favouring immutability
    D. Avoiding exceptional returns

Quiz

2. Imagine you are creating a Rest API for a course booking system at a university that allows students to register for courses. You are currently working on a draft of the API which you want to ensure to be RESTful and well designed. For each of the following API calls, mark which one is a well designed REST API call.

A. GET /courses
B. GET /courses/next
C. GET /students/alice
D. GET /v2/courses

Quiz

3. What does it mean for a REST API request to be idempotent (for the methods other than POST)?

- Making the request multiple times has the same effect as making it once.
- The request changes server state with each call.
- The request always returns a different response.

4. Accessing a REST based service always happens through a uniform resource identifier. [TRUE]