RESTful APIs

REST

Representational State Transfer

Architectural style, set of **design constraints**

Coined in Roy T. Fielding's dissertation (2000)

The Web is the largest implementation

Three important technologies: HTTP, URL, HTML

HTTP

HyperText Transfer Protocol

Request/Response protocol used by browsers to communicate with servers

All about applying **verbs** to **nouns**

Verbs: GET, POST, PUT, DELETE

Nouns: resources (i.e., concepts)

RESOURCES

Nouns

If your users might
“want to create a hypertext link to it, make or refute assertions about it, retrieve or cache a representation of it, include all or part of it by reference into another representation, annotate it, or perform other operations on it”
then, make it a **resource**

They can be anything: a document, a row in a database, the result of running an algorithm, etc.

URL

Uniform Resource Locator

Type of **URI** (`Identifier`)

Specifies the **location of a resource** on a network

Server responds with **representations** of resources and not the resources themselves

REPRESENTATION OF RESOURCES

When a client issues a GET request for a resource, server responds with **representations** of resources and not the resources themselves

Any **machine-readable** document containing any information about a resource

Server may send data from its database as HTML, XML, JSON, etc.

REPRESENTATIONAL STATE TRANSFER

Representations are transferred back and forth from client and server

Server sends a representation describing the **state of a resource**

Client sends a representation describing the **state it would like the resource to have**

MULTIPLE REPRESENTATIONS

A resource can have **more than one** representation: different languages, different formats (HTML, XML, JSON)

Client can distinguish between representations based on the value of **Content-Type** (HTTP header)

A resource can have **multiple representations**—one URL for every representation

Rest in Action

LOADING A PAGE IN A BROWSER

representations of resources

Browser

HTML

Other Resources

http://creativecommons.org

HTTP GET

```
http://creativecommons.org
<a><span id="home-button">
</span></a>
<div id="logo">
 <span>
  Creative Commons
 </span>
```

HTTP GET

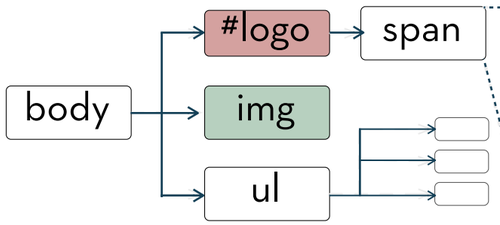
```
cforms.js
//Collap
String.p
function
  return
this.rep
```

```
creativecommons.css
topbar #home-button{
 position: relative;
 float: lef
 display: b
 height: 40
 width: 150
```

cc-logo.png

Rendered Page

Document Object Model (DOM)




```
topbar span {
  float: left;
  display: block;
  height: 40px;
  width: 150px;
  cursor:
pointer;
  z-index: 1;
  top: 0;
```

HTTP GET Request

method url version

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/xml,application/
xml,application/xhtml+xml,text/html*/*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
<blank line>
```

request headers

HTTP GET Response

version

status code

text explanation

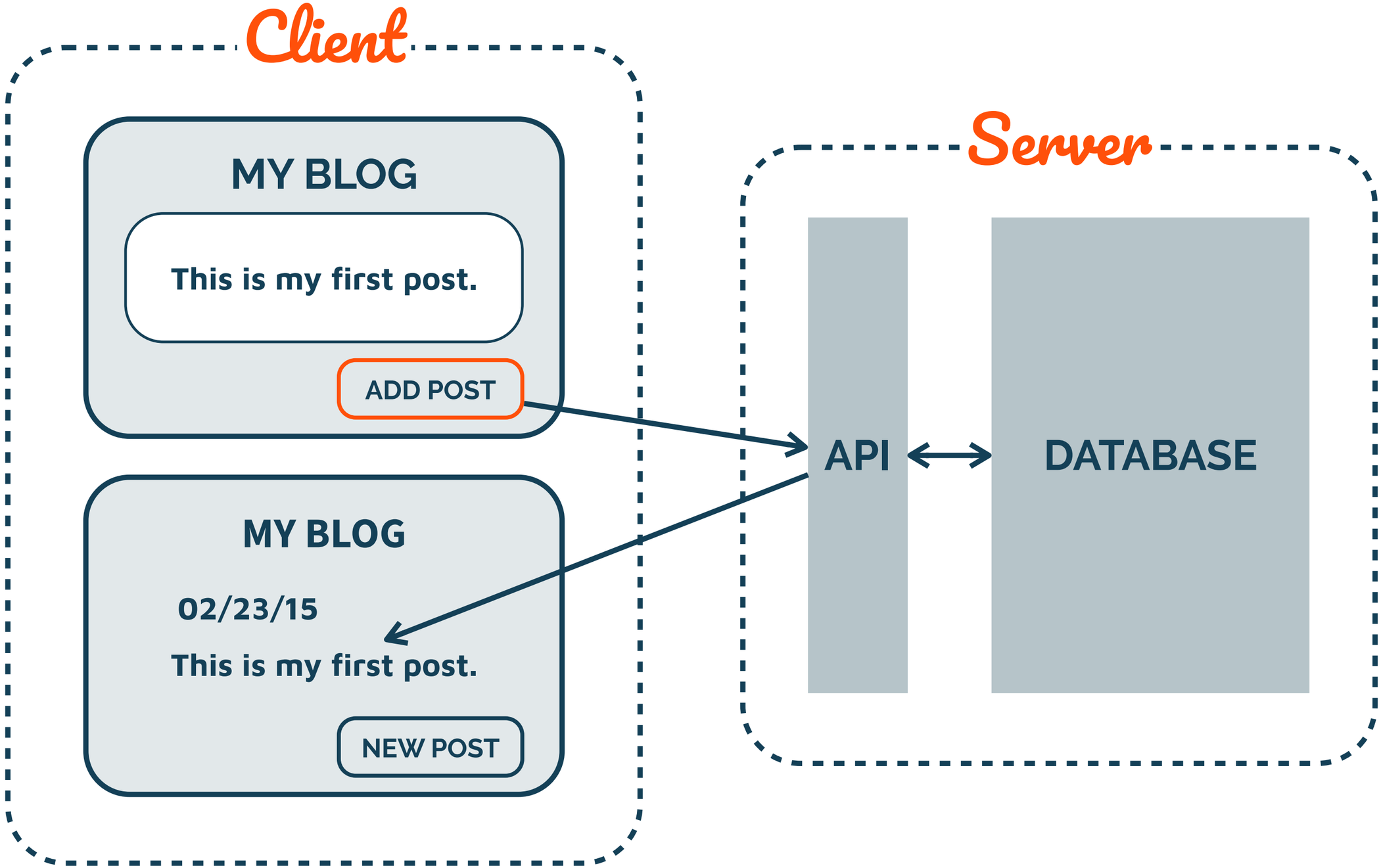
```
HTTP/1.1 200 OK
```

response headers

```
Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

Content-Type: text/html; charset=UTF-8

Content-Length: 131
```

body

```
<!DOCTYPE html>

<html>

…

</html>
```


Client
MY BLOG
This is my first post.
ADD POST
MY BLOG
02/23/15
This is my first post.
NEW POST
Server
API
DATABASE

HTTP POST Request

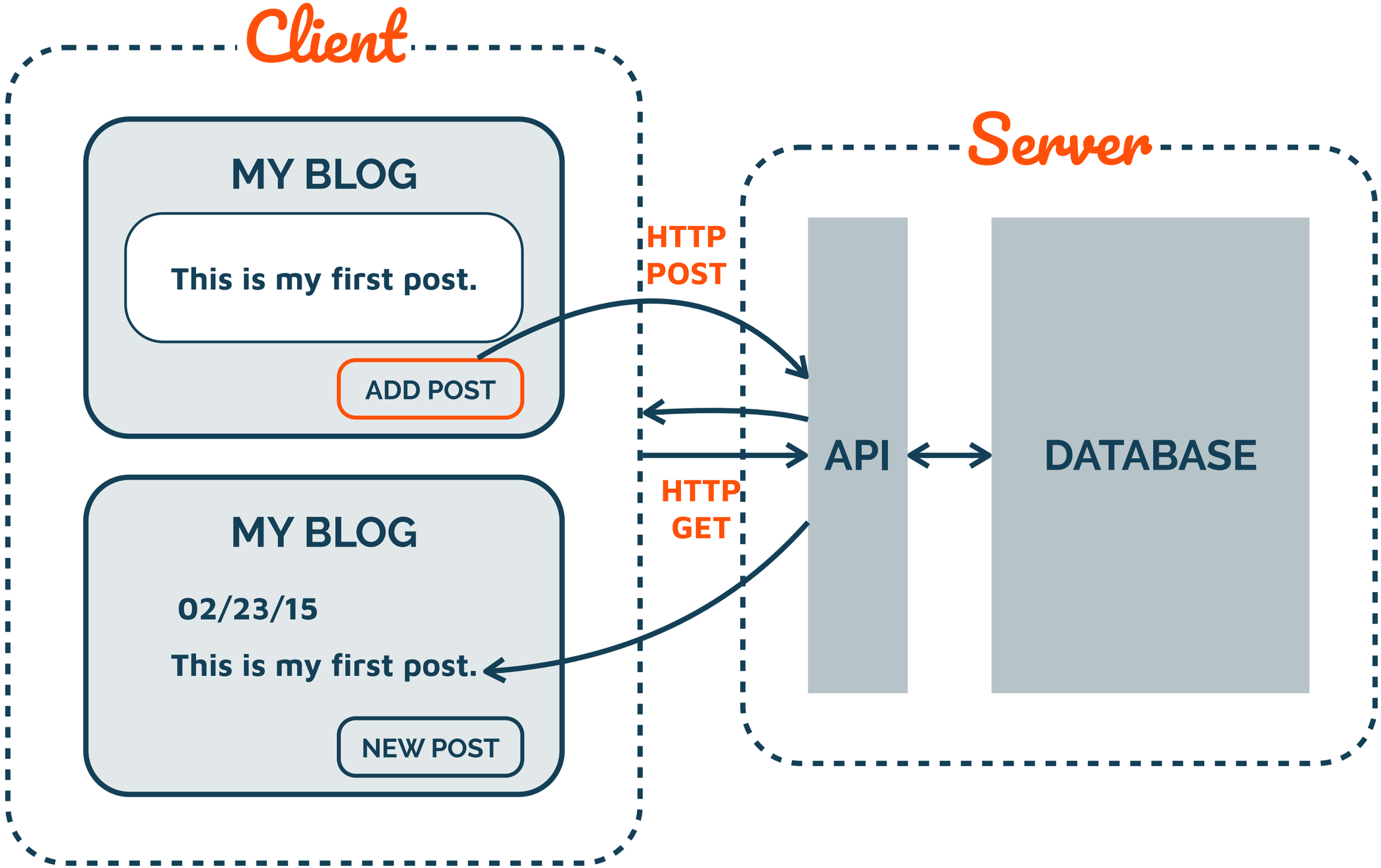
```
POST /messages HTTP/1.1

Host: www.anotherblogpost.com

Content-type: application/x-
www-form-urlencoded

<blank line>

entity-body
```

HTTP POST Response

HTTP/1.1 303 See Other

Content-type: text/html

Location: http://
www.anotherblogpost.com/
messages/3486152


Client
MY BLOG
This is my first post.
ADD POST
MY BLOG
02/23/15
This is my first post.
NEW POST
HTTP
POST
HTTP
GET
Server
API
DATABASE

Http Methods

Verbs

GET	Get a representation of resource
DELETE	Destroy resource
POST	Create a new resource based on the given representation
PUT	Replace resource state with the one described in the given representation
HEAD	Get the headers that would be sent with a representation, but not the representation itself
OPTIONS	Discover which HTTP methods this resource responds to
PATCH	Modify part of the state of this resource based on the given representation

GET

Retrieve representations of resources

Safe Method: no side effects, not intended to change any resource state

Response codes: 200 (OK), 302 (Moved Permanently), 404 (Not Found)

DELETE

Destroy a resource on the server

Success response codes: 200 (OK), 204 (No Content), 202 (Accepted)

Not safe, but **idempotent**

POST

Upload data from the browser to server

Usually means “create a new resource,” but can be used to convey **any kind of change**: PUT, DELETE, etc.

Data contained in request body

Success response codes: 201 (Created), Location header contains URL for created resource; 202 (Accepted), new resource will be created in the future

Not safe or idempotent

PUT

Request to **modify** resource state

Success response codes: 200 (OK), 204 (No Content)

Can also be used like POST

Idempotent

PATCH

representations can be big: PUTs can be inefficient

send the server the **parts of the document** you want to change

neither safe nor idempotent

Rest Constraints

CLIENT-SERVER

separation between clients from servers

servers and clients be replaced and developed independently as long as the interface between them is not altered

en.wikipedia.org/wiki/Representational_state_transfer#Stateless

STATELESSNESS

server doesn’t know about client’s application state

client has no direct control over resource state

pass representations around to change state

UNIFORM INTERFACE

Identification of resources

manipulation of resources through these representations

self-descriptive messages

hypermedia as the engine of application state (HATEOAS)

OTHER CONSTRAINTS

cacheable

layered system

code-on-demand (optional)

Web Apis

WEB APIs

application program interface to a defined request-response message system between clients and servers

accessible via standard HTTP methods

request URLs that transfer representations (JSON, XML)

spf13.com/post/soap-vs-rest

REST vs SOAP

resources vs operations

REST new-hotness

SOAP security, ACID transactions, reliable messaging

spf13.com/post/soap-vs-rest

XMLHttpRequest

most widely deployed API client in the world

a copy in every web browser

most sites today are built on top of APIs designed for consumption by XMLHttpRequest

arRESTed Development

SEMANTIC CHALLENGE

Learning one API doesn’t help a client learn the next one

ProgrammableWeb: the world's largest API repository, GROWING DAILY

Search Over 13,068 APIs Search APIs

Filter APIs

By Category By Protocols/Formats Include Deprecated APIs

API Name	Description	Category	Updated
Google Maps	The Google Maps API allow for the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript interface or a Flash interface. It is designed to work on both mobile...	Mapping	12.05.2005
Twitter	The Twitter micro-blogging service includes two RESTful APIs. The Twitter REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user...	Social	12.08.2006
YouTube	The Data API allows users to integrate their program with YouTube and allow it to perform many of the operations available on the website. It provides the capability to search for videos, retrieve...	Video	02.08.2006
Flickr	The Flickr API can be used to retrieve photos from the Flickr photo sharing service using a variety of	Photos	09.04.2005

API Directory Search

Search over 13,068 APIs updated daily

Search APIs, mashups, developers

Browse by Category

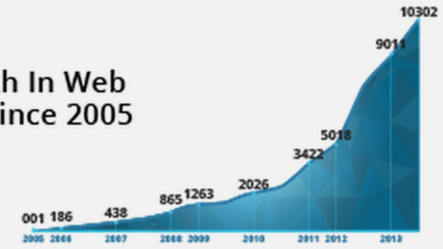
Newest APIs

Latest Mashups

Add an API +

PW Research Center

Our data. Your PowerPoints. Use our API research for your next presentation. See all



Designing Restful Apis

blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/

www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api

Apply Verbs to Nouns

Resources

Http Methods

COLLECTIONS

<VERB> http://example.com/users

GET Return all the objects in the collection

POST Create a new entry in the collection; automatically assign new URI and return it

PUT and DELETE **not generally used**

ELEMENTS

`<VERB> http://example.com/users/12345`

`GET` Return the specific object in collection

`PUT` Replace object with another one

`DELETE` Delete element

`POST` not generally used

USING PARAMETERS

```
<VERB> http://example.com/users?
where={"num_posts":{"$gt":100}}}
```

other parameters can be used to select fields, sort, etc.

parameters can also be URL-encoded

ONE-TO-FEW

How would you access the address of a particular user?

ONE-TO-FEW

```
GET http://example.com/users/12345
```

ONE-TO-MANY

How would you access the posts of a particular user?

ONE-TO-MANY

```
GET http://example.com/users/12345
```

```
GET http://example.com/posts?
where={"_id":{"$in":[...]}}
```

PAGINATION

```
GET http://example.com/users?
        offset=60&limit=20
```

`offset` *ith object*

`limit` *number of returned objects*

can also use **`Link`** header to specify next, prev, first, last URLs

CHECKLIST: BASICS

Use nouns but no verbs

Use plural nouns

Don’t expose irrelevant nouns

GET method and query parameters should not alter the state

CHECKLIST: BASICS

Use parameters to filter, sort, and select fields from collections

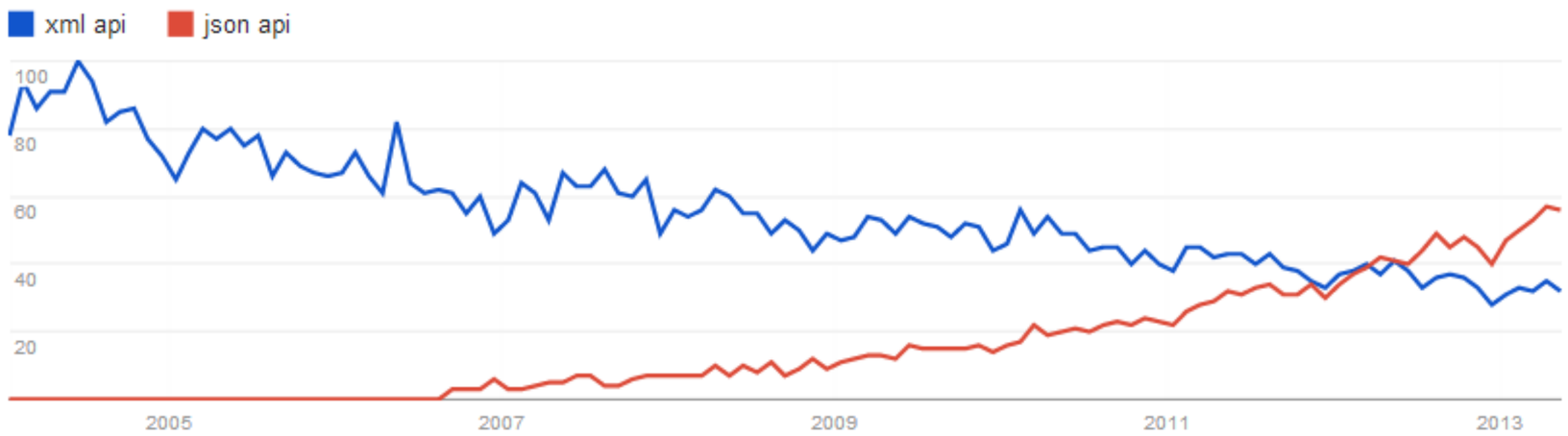
Use offset and limit parameters to paginate results

CHECKLIST: RELATIONS

if a relation is usually requested alongside the resource, **embed the relation's representation** within the output representation of the resource

if a relation can exist independently, **include an identifier** for it within the output representation of the resource

CHECKLIST: FORMATS



`Content-Type` and **`Accept`** headers

Can also explicitly declare format in URL

CHECKLIST: INTERFACING WITH CONSUMERS

Handle Errors with HTTP status codes

An API is only as good as its documentation

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="https://bank.example.com/accounts/12345/deposit" />
    <link rel="withdraw" href="https://bank.example.com/accounts/12345/withdraw" />
    <link rel="transfer" href="https://bank.example.com/accounts/12345/transfer" />
    <link rel="close" href="https://bank.example.com/accounts/12345/close" />
</account>
```

CHECKLIST: HATEOS?

Hypermedia as the Engine of Application State

navigate the Web by following links

should the API consumer create links or should they be provided?

Better to assume the user has access to the documentation & include resource identifiers in the output representation

Advantages: stored data and data over the network minimized, ids more stable than URLs

CHECKLIST: PREVENT ABUSE

Rate Limiting

Authentication

CHECKLIST: CACHING

`ETag` contains a hash or checksum of the representation validated against client's **`If-None-Match`**. If match, the API returns a 304 Not Modified status code

`Last-Modified` contains a timestamp which is validated against **`If-Modified-Since`**

NEXT CLASS:
ASYNC PROGRAMMING

https://uiuc-web-programming.gitlab.io/fa21/