REST API Design

DRAFT

HTTP & REST Principles

Hany ahmed
namozag.com/hany

Outline

HTTP

HTTP Features

Web services

REST

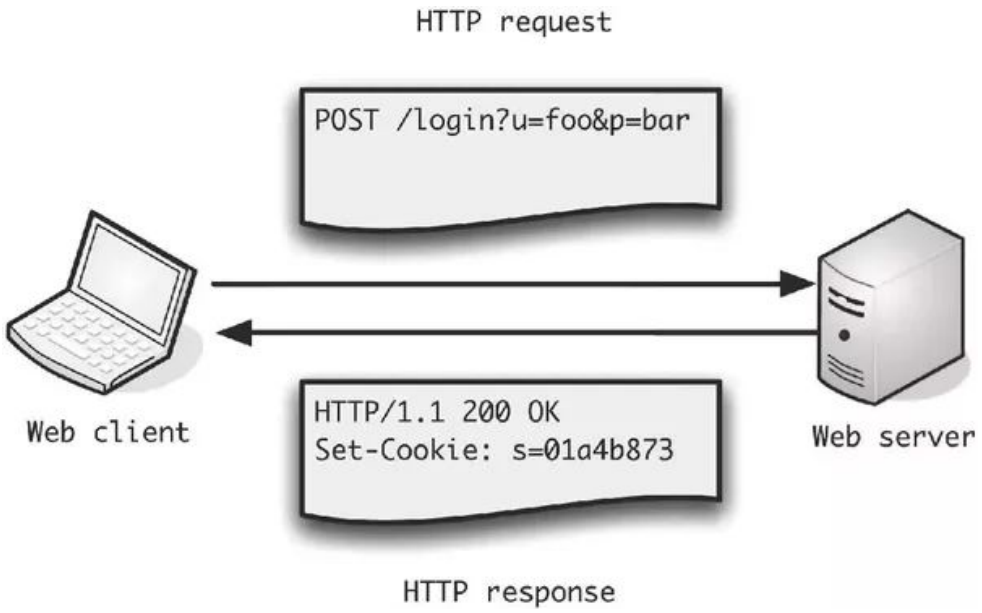
Demos

—

HTTP Protocol

HTTP

Application-level Communication protocol



HTTP

Stateless

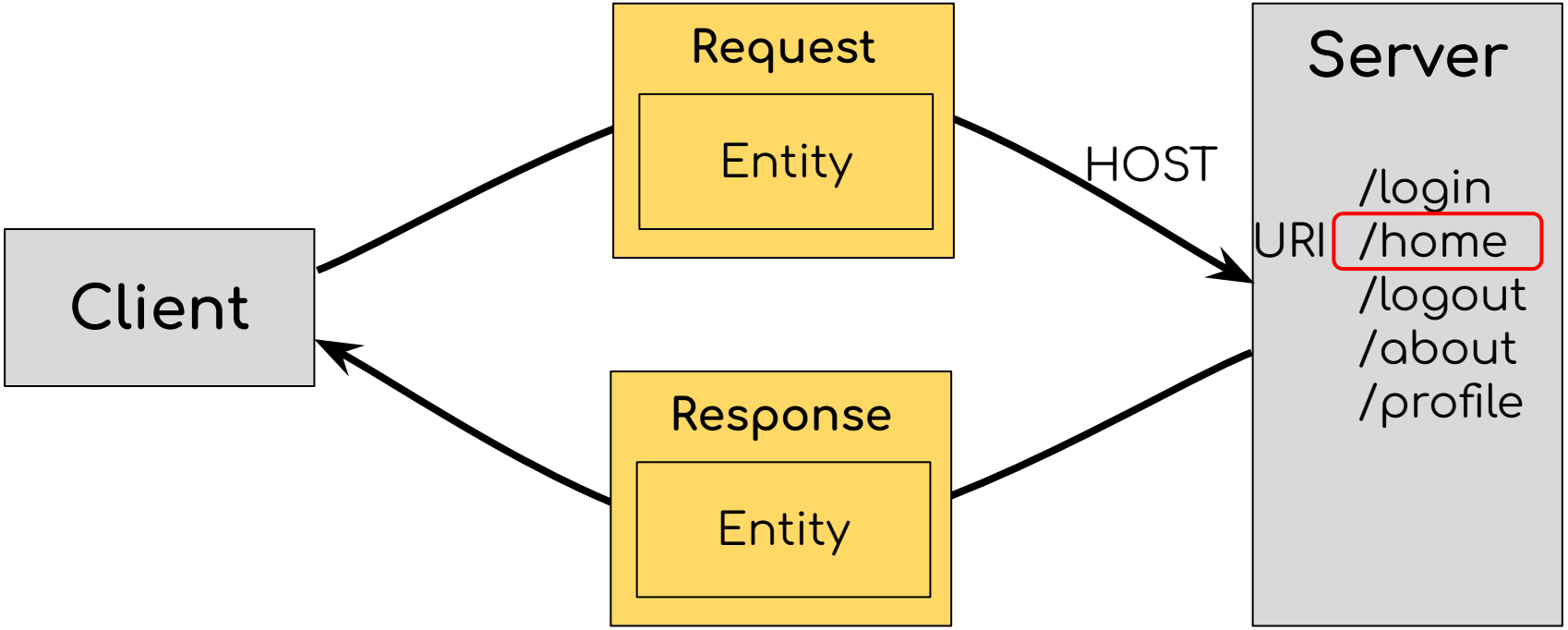
Half-duplex

Client-driven

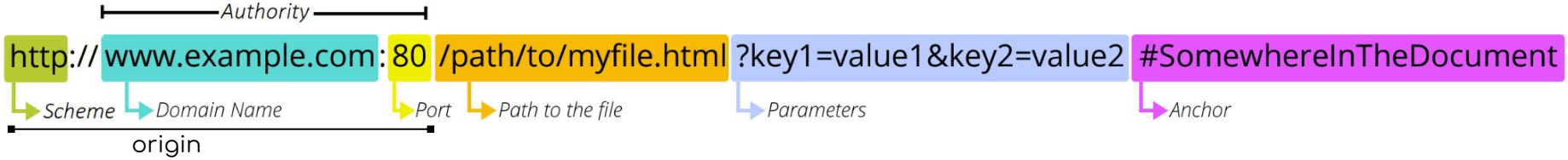
Verbose

—

HTTP Terminology



URL


Authority
http://
www.example.com
:
80
/path/to/myfile.html
?key1=value1&key2=value2
#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor
origin

HTTP Request

```
GET /httpgallery/introduction/
HTTP/1.1
Accept: */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT
6.3; WOW64; Trident/7.0; rv:11.0)
like Gecko
Host: www.httpwatch.com
Connection: Keep-Alive
```

HTTP Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/8.0
Date: Mon, 04 Jan 2015 12:04:43 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 4.0.30319
Cache-Control: no-cache, no-store
Expires: -1
Content-Type: text/html;
charset=utf-8
Content-Length: 14990

<!DOCTYPE html>  <html>...
```

Status code

Explains the HTTP response status

1xx - Informational

2xx - Successful

3xx - Redirection

4xx - Client Error

5xx - Server Error

2xx - Successful

200 OK

201 Created

202 Accepted

204 No content

3xx - Redirection

301 Moved permanently

302/307 Moved temporarily

303 Not modified → read from local cache

4xx - Client Error

400 Bad request

401 Unauthorized

403 Forbidden

404 Resource not found

405 Method not allowed

415 Unsupported media type

5xx - Server Error

500 Internal server error

501 Not implemented

503 Service Unavailable

HTTP Methods

GET HEAD

POST DELETE

PUT PATCH

OPTIONS

Safe vs. Idempotent

Safe	return a	(Cacheable)
Idempotent	a=1	(~Cacheable)
	[a, b, c] - c	
Not safe/idempotent	a++	

Safe

Read only

Doesn't change the representation of the resource

Idempotence

Making multiple identical requests has the same effect as making a single request
It's about the effect produced on the representation of the resource and not the response status code received

May lead to side effects

GET

Safe✓ Idempotent✓ Cacheable✓ HTML✓

URL

```
GET /products/2105
```

```
200 OK
```

Headers

```
Content-Type: application/json
```

Body

```
{
    id: 2105,
    name: “Lenovo thinkpad”,
    price: 8500
}
```

The same for HEAD, OPTIONS

POST

Safe ✗ Idempotent ✗ Cacheable ✗ HTML ✓

URL

```
POST    /products
```

```
201 CREATED
```

Headers

```
Content-Type: application/json
```

```
Location: /products/11
```

Body

```
{
    name: “X Box”,
    price: 5200
}
```

POSTing the same request again would result on /products/12

PUT

Safe ✗ Idempotent ✓ Cacheable! HTML ✗

URL

```
PUT     /products/4811
```

```
204 No Content
```

Headers

```
Content-Type: application/json
```

Body

```
{
    name: “X Box special”,
    price: 5000
}
```

* May result in 201 Created, 200 OK if response has body

PUT vs. PATCH

PUT

=Replace

Affects the whole entity

PATCH

=Update

Affects only some fields

PATCH

Safe ✗ Idempotent! Cacheable! HTML ✗

URL

```
PATCH   /products/4811
```

```
201 CREATED
```

Headers

```
Content-Type: application/json
```

```
Location: /products/4811
```

Body

```
{
    price: 5200
}
```

PATCH may have different way of implementation (JSON Patch)
* May return 204

DELETE

Safe ✗ Idempotent ✓ Cacheable ✗ HTML ✗

URL

```
DELETE /products/4811
```

```
204 No Content
```

Headers

Body

* May result in 200 OK if response has body, 404 when not found

HTTP Headers

Nodes sends them to each other to know more about each others and the sent messages

Request headers

User-Agent

Accept:*/*

Accept-Language: en-gb

Accept-Encoding: gzip, deflate

Host: www.abc.com

Response Headers

Cache-Control: no-cache

Content-Length: 2748

Content-Type: image/gif

Date: Wed, 4 Oct 2004 12:00:00 GMT

—

HTTP Capabilities

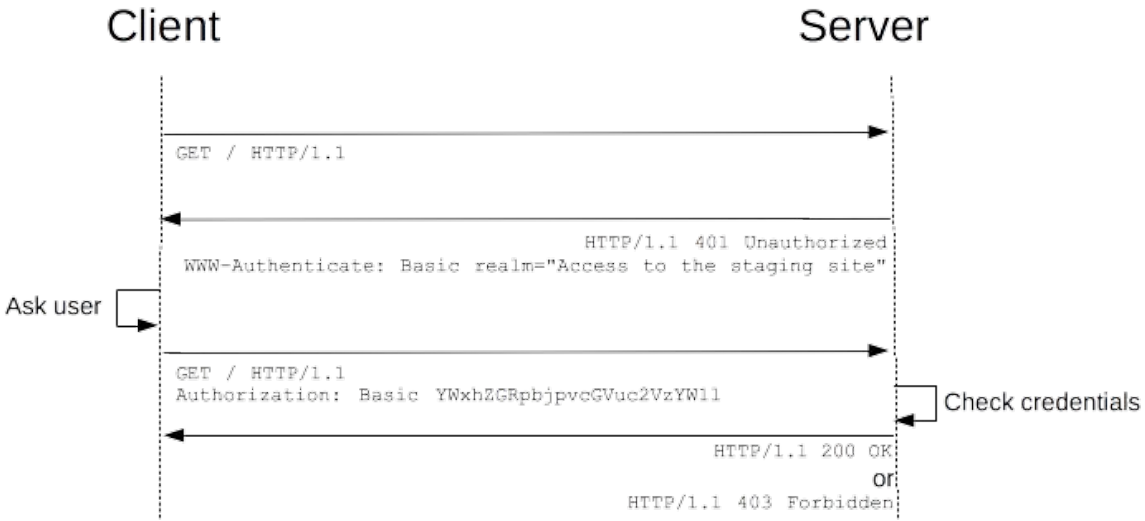
HTTP Capabilities

Authentication
Caching
Client hints
Conditionals
Content negotiation
CORS
...

Authentication

WWW-Authenticate

Authorization



Caching

Age

Cache-Control

Expires

Conditionals

Last-Modified

If-Modified-Since

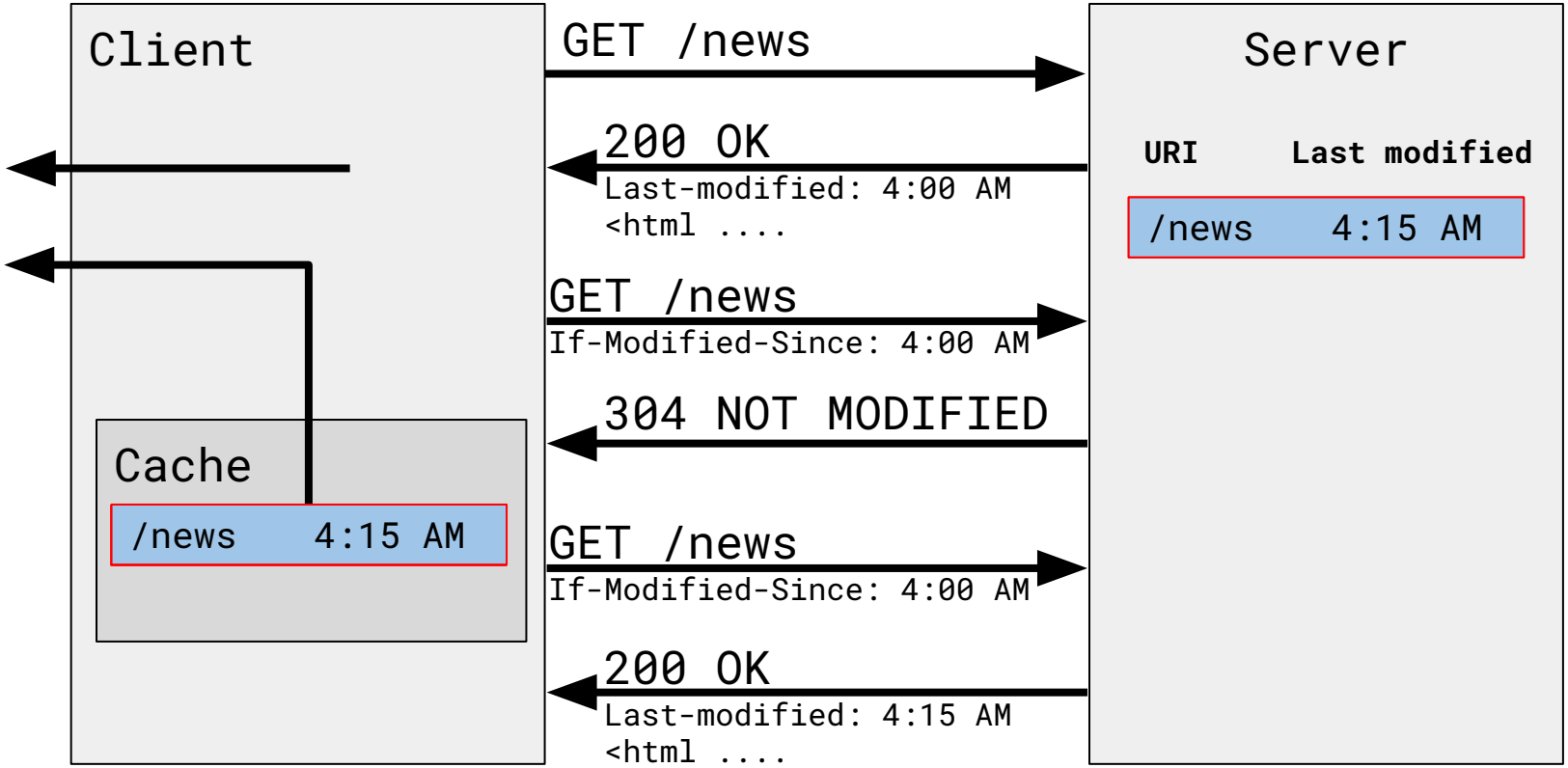
If-Unmodified-Since

ETag

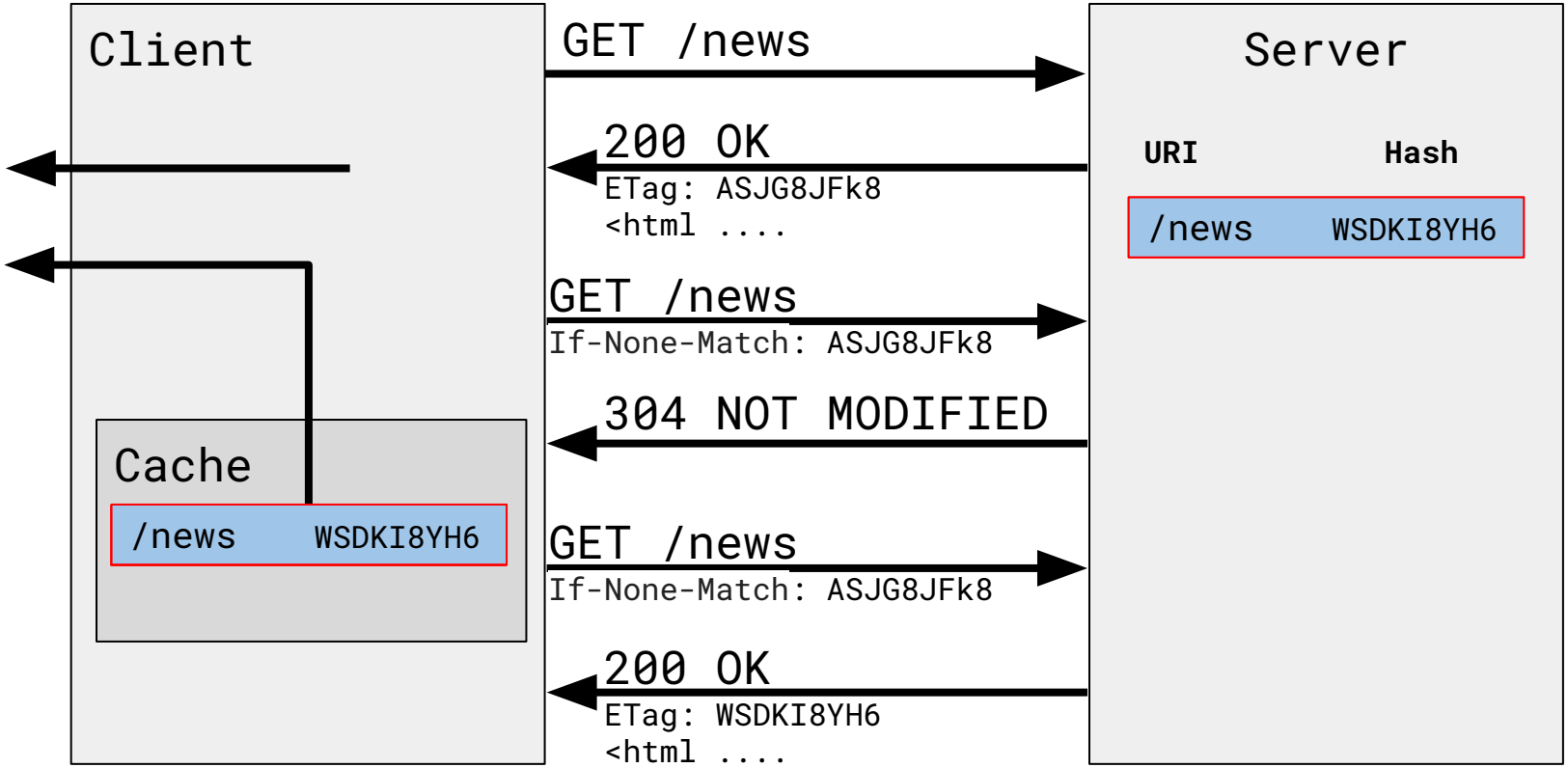
If-Match

If-None-Match

Last-modified



ETag


Client
Cache
/news WSDKI8YH6
GET /news
200 OK
ETag: ASJG8JFk8
<html ....
GET /news
If-None-Match: ASJG8JFk8
304 NOT MODIFIED
GET /news
If-None-Match: ASJG8JFk8
200 OK
ETag: WSDKI8YH6
<html ....
Server
URI Hash
/news WSDKI8YH6

Content negotiation

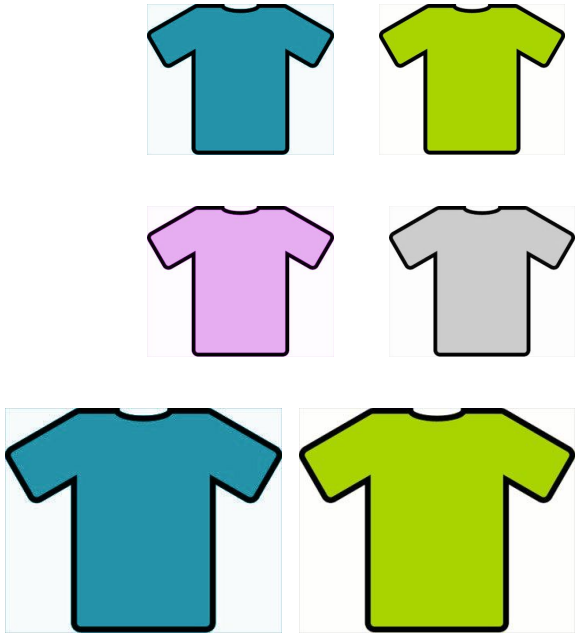
/shirts/4 color: Green size: M

200 OK

/shirts/4 color: Gray size: L

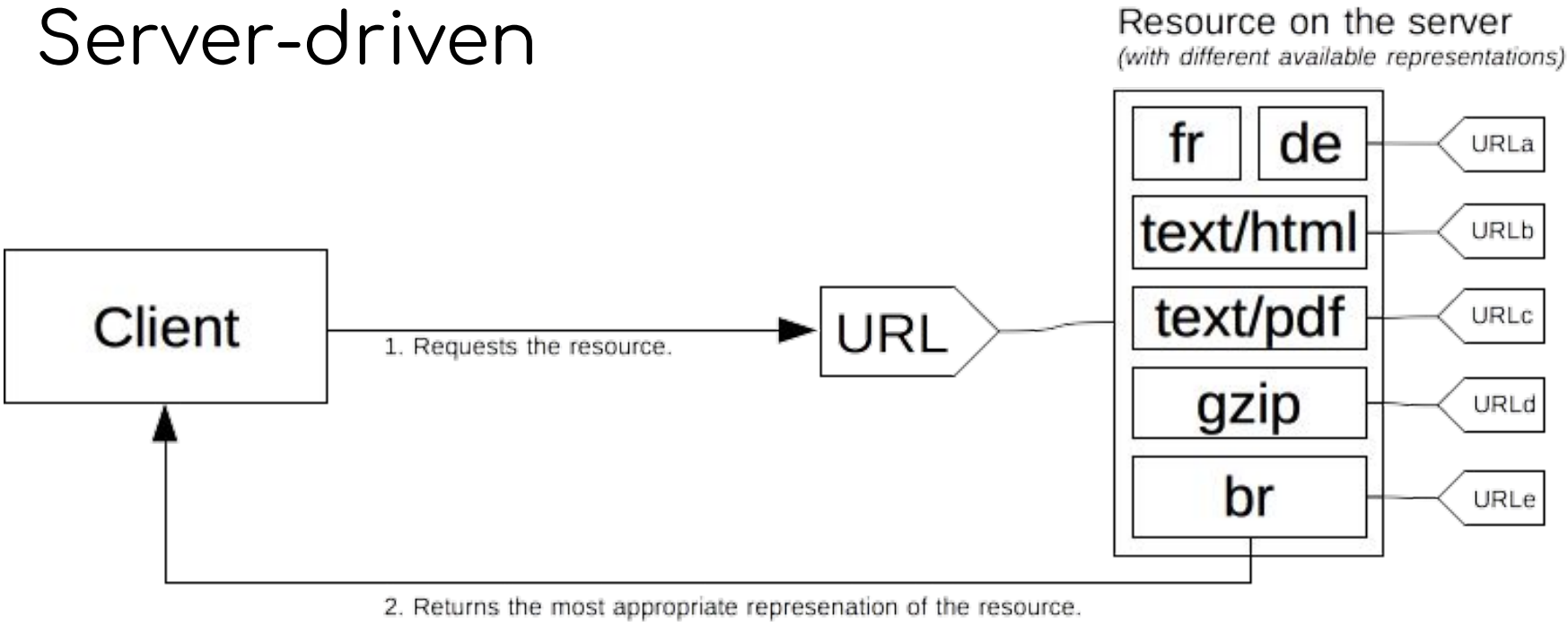
406 NOT ACCEPTABLE

Resource representations



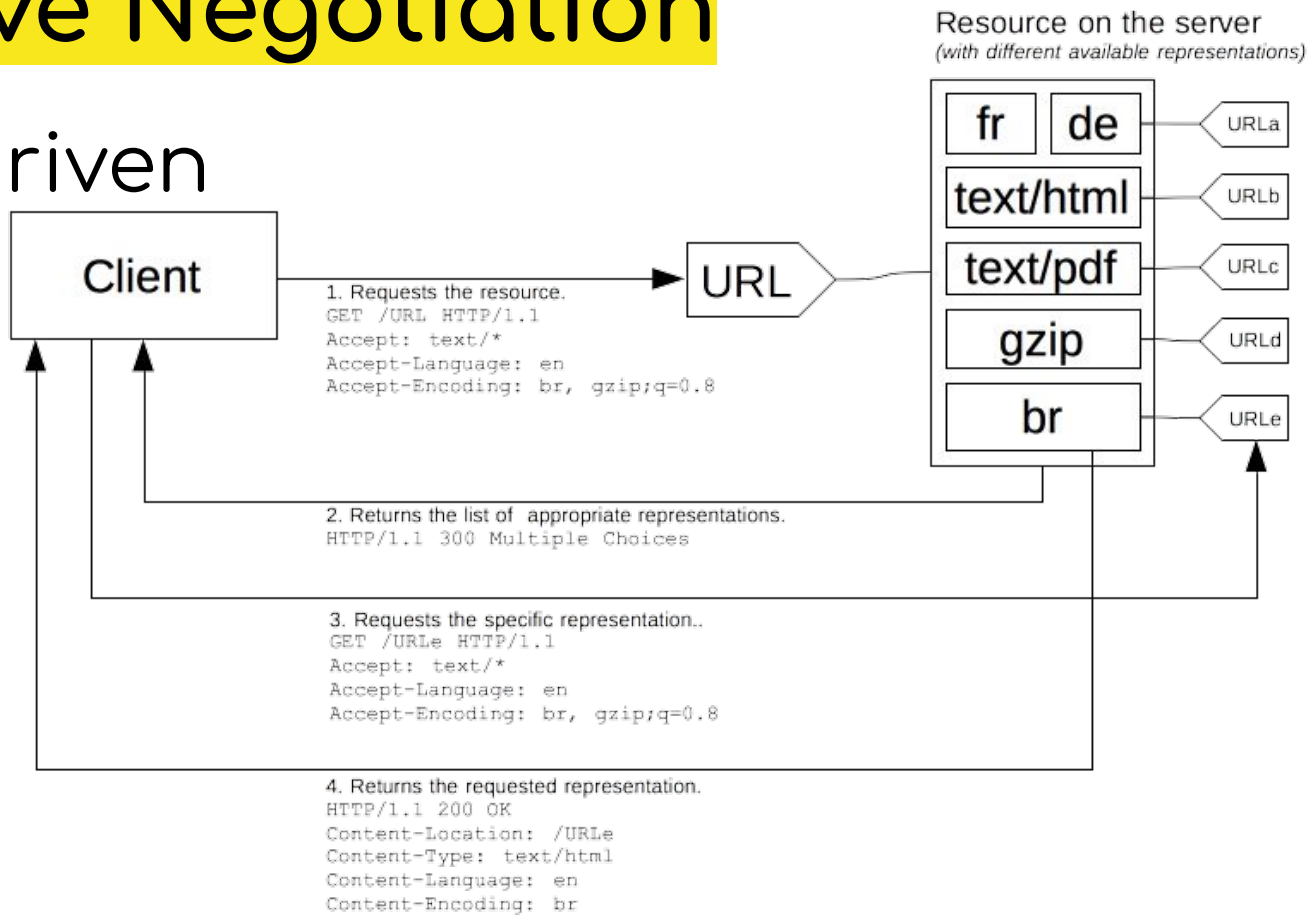
Proactive Negotiation

Server-driven



Reactive Negotiation

Agent-driven



Content negotiation headers

Accept

Accept-Charset

Accept-Encoding

Accept-Language

Content-Type

Content-Encoding

Content-Language

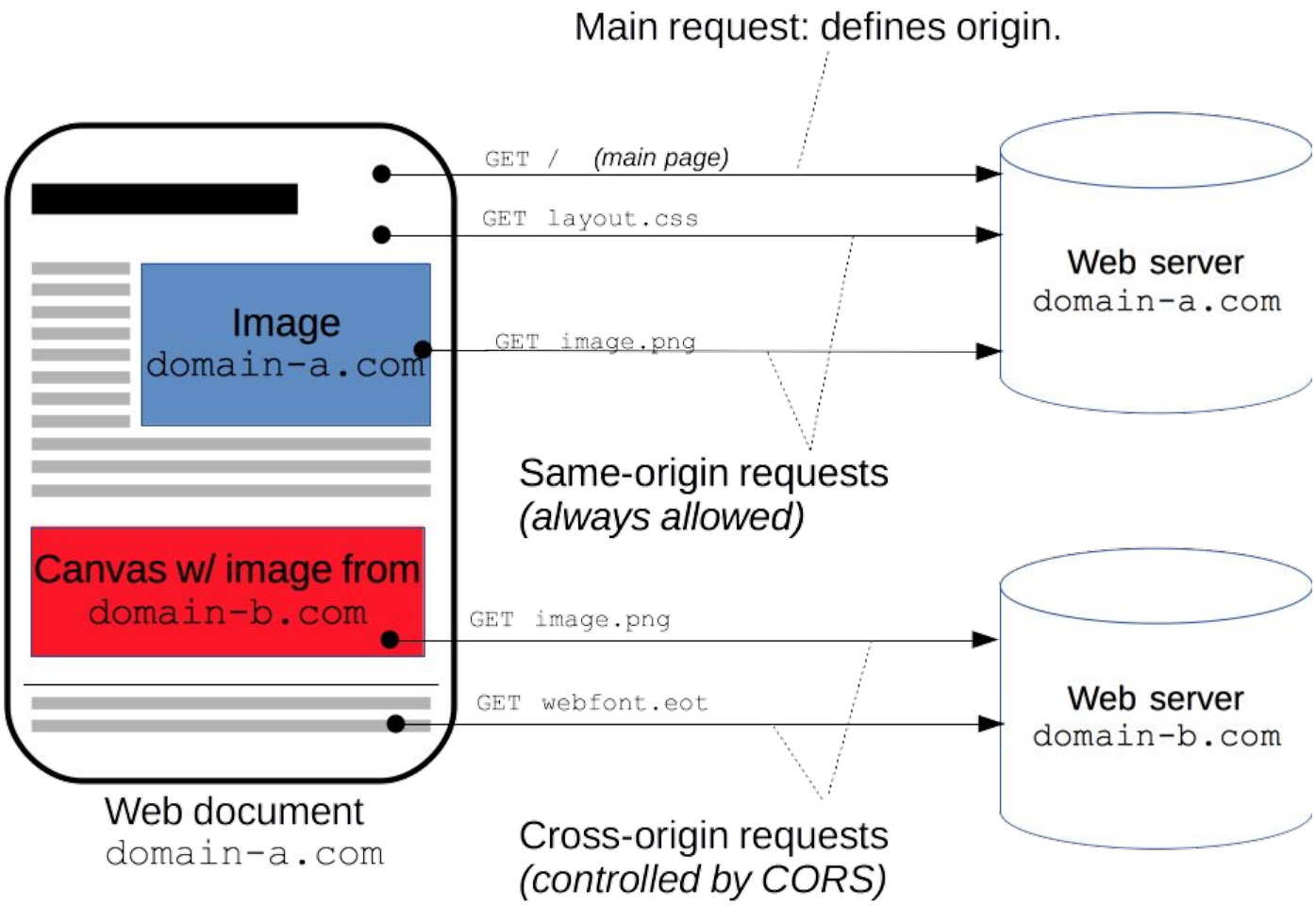
Content-Length

Cookies

Set-Cookie

Cookie

CORS


Main request: defines origin.
GET / (main page)
GET layout.css
GET image.png
Image
domain-a.com
Web server
domain-a.com
Same-origin requests
(always allowed)
Canvas w/ image from
domain-b.com
GET image.png
GET webfont.eot
Web server
domain-b.com
Web document
domain-a.com
Cross-origin requests
(controlled by CORS)

CORS

Origin

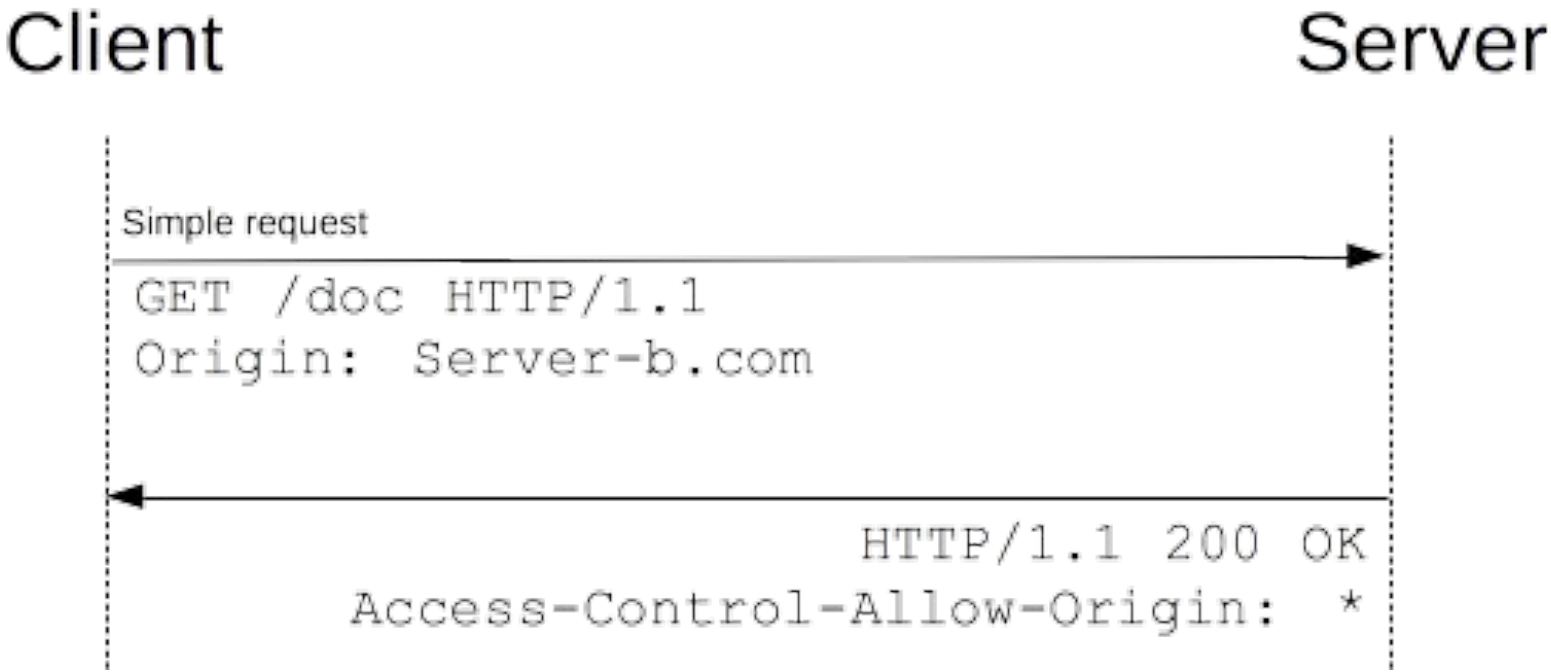
Access-Control-Allow-Origin

Access-Control-Allow-Credentials

Access-Control-Allow-Methods

Access-Control-Allow-Headers

CORS Simple requests



CORS Preflighted requests

1. Preflight request

```
OPTIONS /doc HTTP/1.1
Origin: Server-b.com
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
```

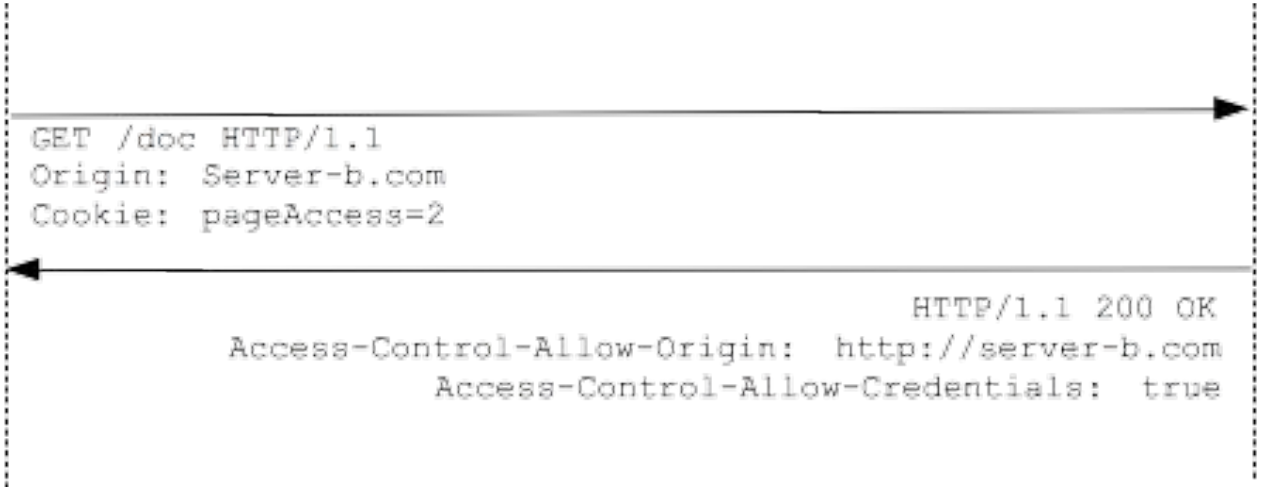
2. Main Request

```
POST /doc HTTP/1.1
X-PINGOTHER: pingpong
Content-Type: text/xml; charset=UTF-8
Origin: Server-b.com
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://foo.example
```

Requests with credentials

Requests that are aware of HTTP cookies and HTTP Authentication information



...

User-Agent

Referer

Host

Date

Forwarded

X-Forwarded-For

Location

Date

Web services

Web service

Machine to machine communication over a network.

Data transfer protocol

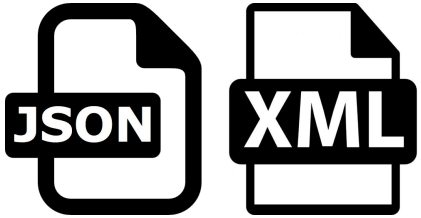
+

Standard data exchange format

=

Web service

http://



Web service

Loosely coupled

Interoperable

Platform / OS / Language independent

SOAP

A protocol.

Message oriented.

Message Envelope is XML.

WSDL as definition language.

Sample SOAP Request

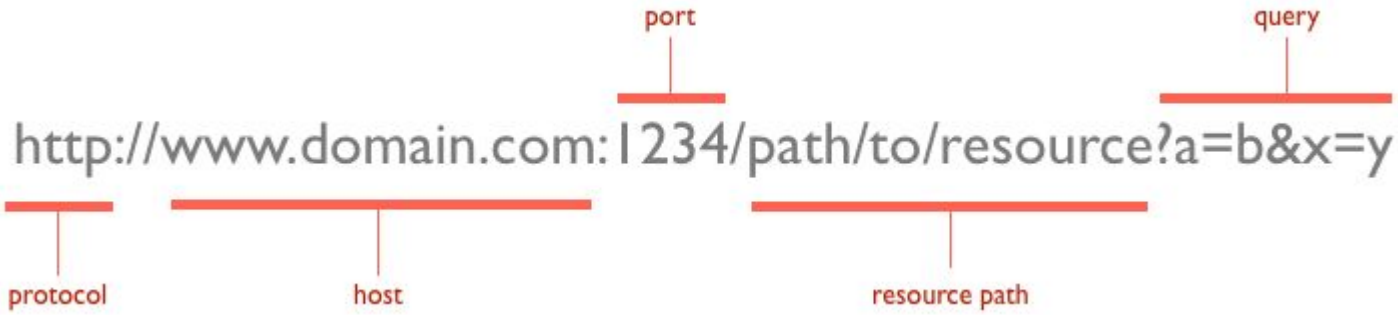
```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

      <soap:Body xmlns:m="http://www.example.org/stock">
        <m:GetStockPrice>
          <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
      </soap:Body>

</soap:Envelope>
```

REST

Architectural style.

Make use of HTTP protocol features:

URI, Methods, Headers



SOAP	REST
Protocol	Architectural style
HTTP POST only	HTTP GET, POST, ...
Message oriented	Resource oriented
XML Envelope	No Envelope
WSDL definition	OpenApi definition

REST

HTTP Representation

- Entity
  - Payload information of a request or response
- Content negotiation
  - The mechanism for selecting the appropriate representation when servicing a request
- Representation
  - An entity included with a response that is subject to content negotiation

REST Terminology

Resource: an object or representation of something

Collections: set of resources

Media/Content type: the way a resource can be represented

REST Maturity Levels

#0 Single URL

#1 Resources

#2 HTTP verbs

#3 HATEOAS

Level 0 Single URL

Single URL for all requests

Much like SOAP

```
POST /api

{ “code”:”updateUser”, “data”: { … } }
```

Level 1 Resources

Identify your application resources

Resources may be mapped but not limited to Entities

End point	Entity	Table
/users/add	User	user
/products	Product	product

Level 2 HTTP Verbs

GET	/users /11	→ Retrieve
POST	/users	→ Create
PUT	/users/11	→ Update (Replace)
PATCH	/users/11	→ Update (Partial)
DELETE	/users/11	→ Delete

Data-oriented Approach

SOAP → functions

REST → Domain entities

Standard way to access HTTP resources, like the standard way you access DB data

Designing representation

Resource Representation

Users can view the representation of a resource in different formats, called **media types**.

JSON

2 Data structures: Object, Array
Limited data types: Null, Boolean, Number, String

No standard way to represent date

shorter than XML and faster in parsing

Relations in JSON

```
{
    "name": "Ahmed",
    "company": "abc"
}
```

```
{
    "name": "Ahmed",
    "company-link":
"http://api.com/companies/abc"
}
```

Links Example


```
{
 "id": 1,
 "url": "https://api.github.com/repos/octocat/Hello/issues/1347",
 "repository_url": "https://api.github.com/repos/octocat/Hello",
 "comments_url": "https://api.github.com/repos/octocat/Hello/issues/1347/comments"
 "user": {
  "login": "octocat",
  "id": 1,
  "avatar_url": "https://github.com/images/error/octocat_happy.gif",
  "url": "https://api.github.com/users/octocat",
  "repos_url": "https://api.github.com/users/octocat/repos",
  "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}"
 }
}
```

Resource Common properties

Type/Kind

Self link/URL

Navigation links: next, previous, first, last

Envelope

Use it when needed only

May be needed to

- Represent errors
- wrap a paginated response
- not able to use some headers

Designing Resources URLs

Naming convention

Noun	`products`
Plural (collections)	`products/1/buyer`
kebab-case: Lowercase hyphen-separated	`action-items`

Query parameters

Filtering

```
GET /products?state=active&category=tv
```

Sorting

```
GET /products?sort=-price,created_at
```

Paging

```
GET /products?offset=10&limit=10  //Facebook
             ?page=2&rpp=10       //Twitter
             ?offset=10&limit=10  //Linkedin
```

Subresources

Path parameter

`/users/11/products`

Matrix parameter

`/users;11/products`

`/users;id=11/products`

`/users;name=ali/products`

* Each path segment represents one relationship traversal

Actions

A service that calculate, translate, convert

```
POST   /translate?value=cat&from=en&to=ar
GET    /translations?value=cat&from=en&to=ar

POST      /products/:id/star / unstar
POST      /products/:id/star
DELETE    /products/:id/star
```

Partial response

Get exactly what you need in the response

Linked In

`/people:(id,first-name,last-name,industry)`

Facebook (And Google)

`/joe.smith/friends?fields=id,name,picture`

Idempotency

You can make idempotent calls any number of times without concern that the server creates or completes an action on a resource more than once

You can retry idempotent calls

Idempotency on POST request

On Paypal

To enforce idempotency on REST API POST calls, use the PayPal-Request-Id request header

a unique user-generated ID that the server stores for a period of time.

API Design Guidelines

API First approach

File ▾ Preferences ▾ Generate Server ▾ Generate Client ▾ Help ▾ ✔ Processed with no error

```yaml
# this is an example of the Uber API
# as a demonstration of an API spec in YAML
swagger: '2.0'
info:
  title: Uber API
  description: Move your app forward with the Uber API
  version: "1.0.0"
# the domain of the service
host: api.uber.com
# array of all schemes that your API supports
schemes:
  - https
# will be prefixed to all paths
basePath: /v1
produces:
  - application/json
paths:
  /products:
    get:
      summary: Product Types
      description: |
        The Products endpoint returns information about the *Uber* products
        offered at a given location. The response includes the display name
        and other details about each product, and lists the products in the
        proper display order.
      parameters:
        - name: latitude
          in: query
          description: Latitude component of location.
          required: true
          type: number
          format: double
        - name: longitude
          in: query
          description: Longitude component of location.
          required: true
          type: number
          format: double
      tags:
        - Products
      responses:
        200:
          description: An array of products
          schema:
```

Uber API

Move your app forward with the Uber API

Version 1.0.0

Filter operations by a tag:

Products Estimates User

Paths

/products

GET /products Products

Summary

Product Types

Description

The Products endpoint returns information about the *Uber* products offered at a given location. The response includes the display name and other details about each product, and lists the products in the proper display order.

Parameters

Name	Located in	Description	Required	Schema
latitude	query	Latitude component of location.	Yes	number (double)
longitude	query	Longitude component of location.	Yes	number (double)

Responses

Code	Description	Schema
200	An array of products	[▸ Product { }

Evolving APIs

Two versions running concurrently

- Embed version in the url `/api/v2/trips`

Robustness principle

- Tolerate Unrelated Changes (minor changes)

...

Permalink

use UUID if name is changeable

Real life Examples

GitHub Repository

PayPal Payment

JIRA Issue

FaceBook Photo

YouTube Video

—

GitHub Repositories resource

```
GET /user/repos

POST /user/repos

GET /repos/:owner/:repo/contributors

DELETE /repos/:owner/:repo

POST /repos/:owner/:repo/transfer
```

PetStore Pet Resource

POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

PetStore User Resource

POST	/user	Create user
GET	/user/login	Logs user into the system
GET	/user/logout	Logs out current logged in user session
GET	/user/{username}	Get user by user name
PUT	/user/{username}	Updated user
DELETE	/user/{username}	Delete user

PayPal Payment Resource

```
POST /v1/payments/payment

GET /v1/payments/payment

GET /v1/payments/sale/{sale_id}

POST /v1/payments/sale/{sale_id}/refund
```

JIRA Issue Resource

```
POST /rest/api/2/issue

GET /rest/api/2/issue/{issueIdOrKey}

PUT /rest/api/2/issue/{issueIdOrKey}

DELETE /rest/api/2/issue/{issueIdOrKey}

PUT /rest/api/2/issue/{issueIdOrKey}/assignee

GET /rest/api/2/issue/{issueIdOrKey}/comment
```

Facebook Photo Resource

```
POST /v2.12/{page_id}/photos

GET /v2.12/{photo-id}

GET /v2.12/{photo-id}/likes

GET /v2.12/{object-id}/comments

POST /v2.12/{photo-id}

DELETE /v2.12/{photo_id}
```

GMail Message Resource

```
GET /userId/messages

GET  /userId/messages/id

POST  /userId/messages/id/modify

DELETE  /userId/messages/id

POST  /userId/messages/send

POST  /userId/messages/id/trash
```

YouTube Video Resource

GET /videos?id=#

POST /videos

DELETE /videos?id=#

PUT /videos?id=#

POST /videos/rate?id=#

Woocommerce Product Resource

```
GET  /v2/products

POST /v2/products

GET /v2/products/<id>

PUT /v2/products/<id>

DELETE /v2/products/<id>

GET /v2/products/<product_id>/reviews
```

WordPress Post Resource

GET /wp/v2/posts

POST /wp/v2/posts

GET /wp/v2/posts/<id>

POST /wp/v2/posts/<id>

DELETE /wp/v2/posts/<id>

Main references

- Web API Design: The Missing Link, apigee.com
- https://developer.mozilla.org/kab/docs/Web/HTTP
- https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design
- https://restfulapi.net/