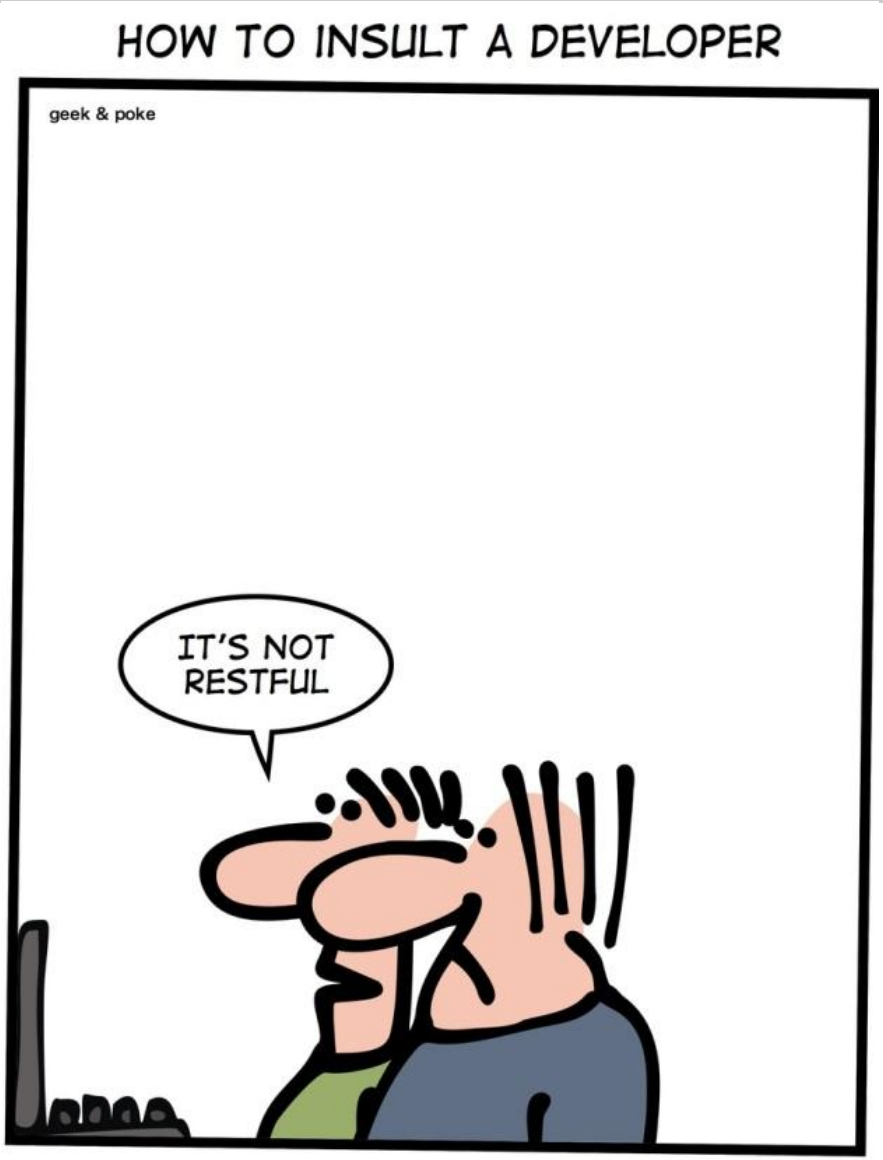
Your Service is not REST

Kostis Kapelonis

Thessaloniki Greece, Dev It Conference

May 2015

Classic comic

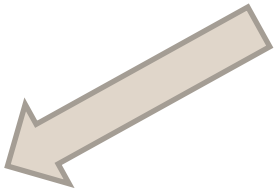


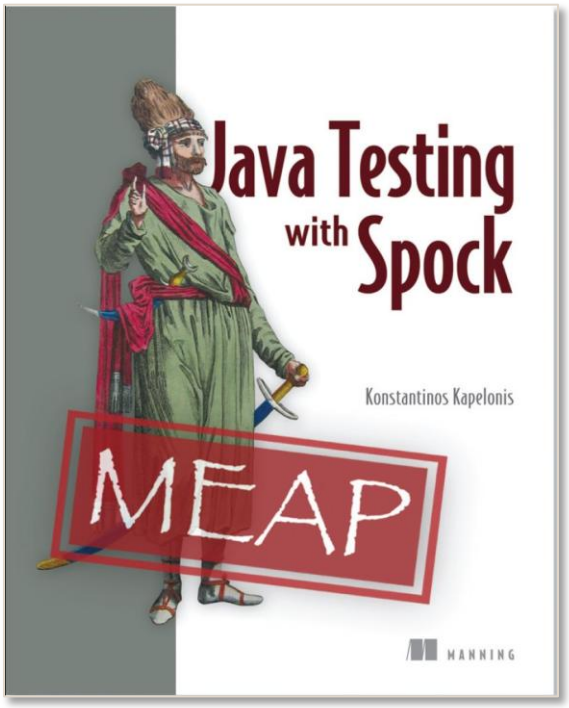
My day job – software engineer

Trasys – (http://trasys.be) – Greek branch in Athens

My night job – technical author

www.enlightenment.org





http://docs.spockframework.org/

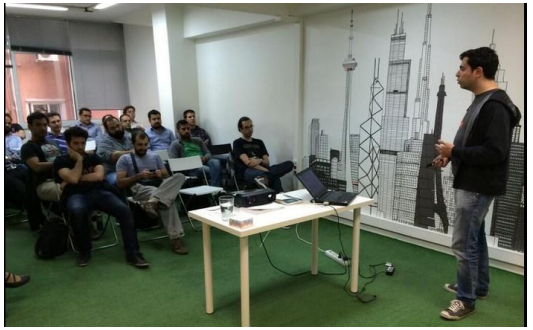
My community

Java Hellenic User Group (jhug.gr) - Presenter

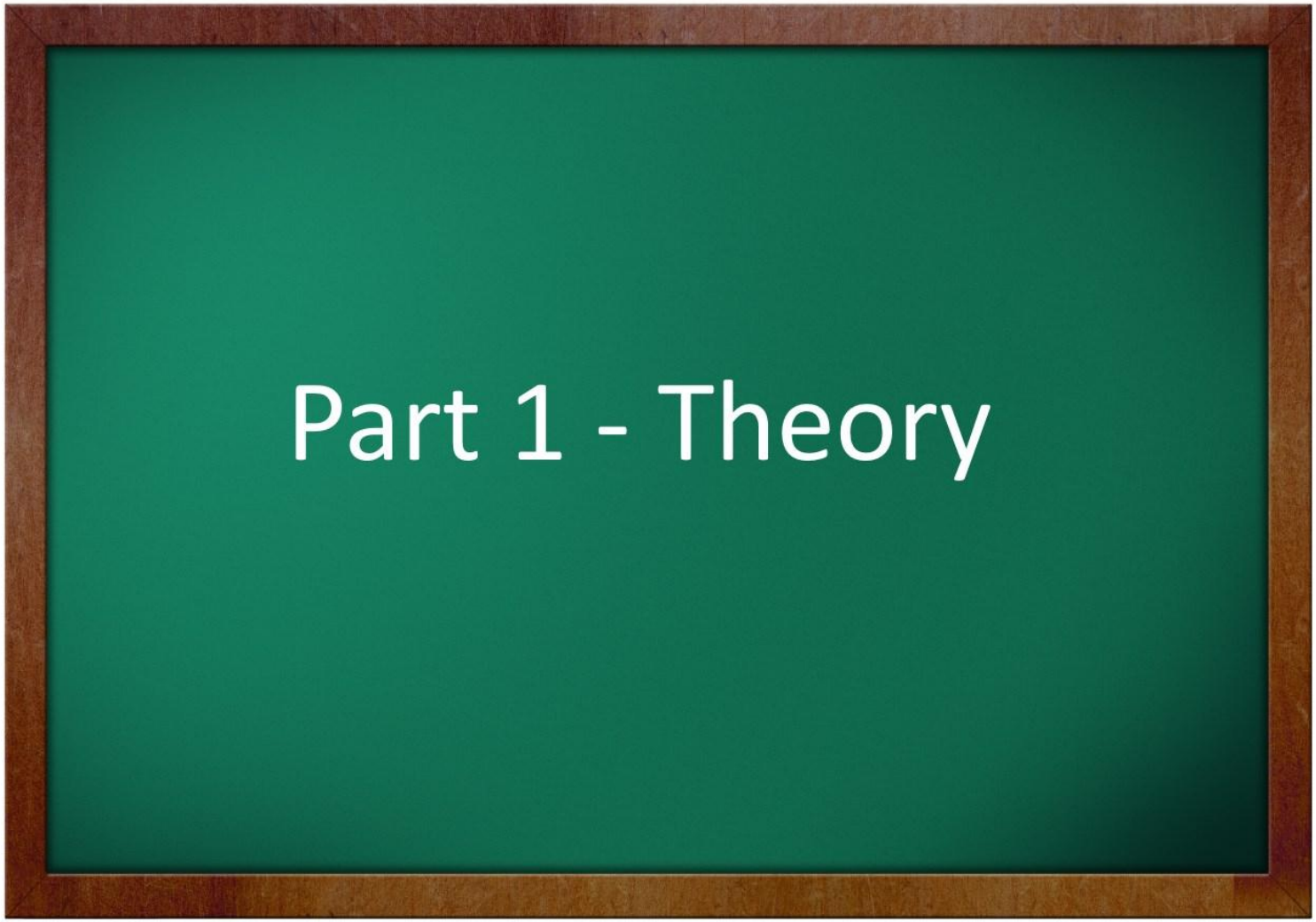
GrJS

Javascript (greecejs.org)

Amazon



Your service is not REST



A little history

A little history

1991

HTTP 0.9 documented

A little history

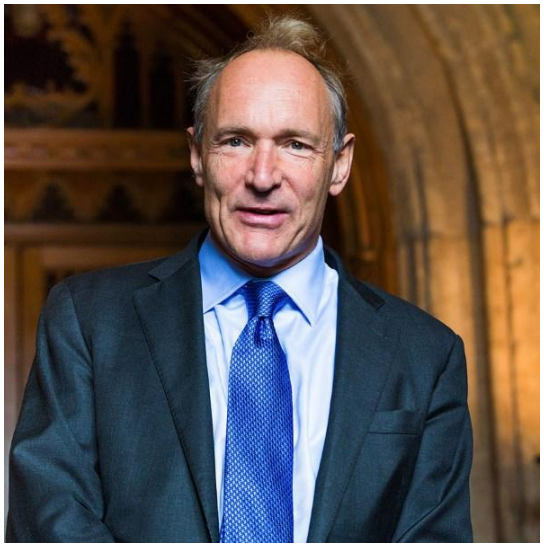
1996

HTTP 1.0 released

Production Ready

Defined in RFC 1945

A little history



1996

HTTP 1.0 team includes

Roy Fielding (and Tim Berners Lee)

Roy Tomas Fielding

Roy was part of the HTTP 1.0 Team

A little history

1999

HTTP 1.1 released

Not really groundbreaking

Defined in RFC 2616

Roy Tomas Fielding

Roy was part of the HTTP 1.1 Team

A little history

2000

A PHD dissertation is released.

(Probably the most hyped over the last decade)

2000

“Architectural Styles and the Design of Network-based Software Architectures”

2000

The REST Architectural style (Representational state transfer)

Roy Tomas Fielding

Roy wrote this PHD!

Roy Tomas Fielding

This guy is important!

He has written the “book”

Roy Tomas Fielding

Part of the HTTP1.0 team

Part of the HTTP1.1 team

Author of “REST” PHD

Read it!

The “REST” PHD is available in HTML and PDF

A REST application

- Client – Server
- Stateless
- Cacheable
- Layered system
- Code on demand (What?)
- Uniform interface

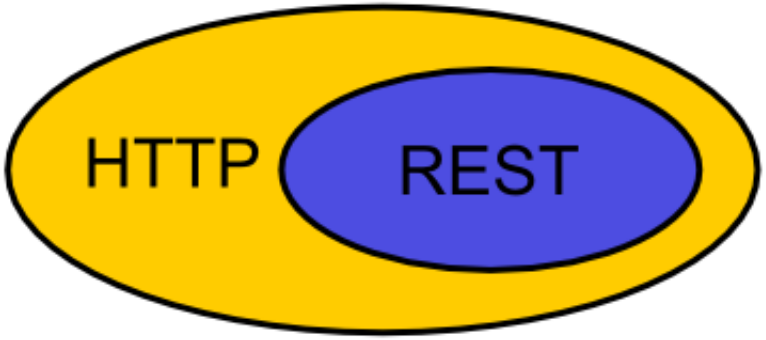
Quiz 1

What is the connection between REST and HTTP?

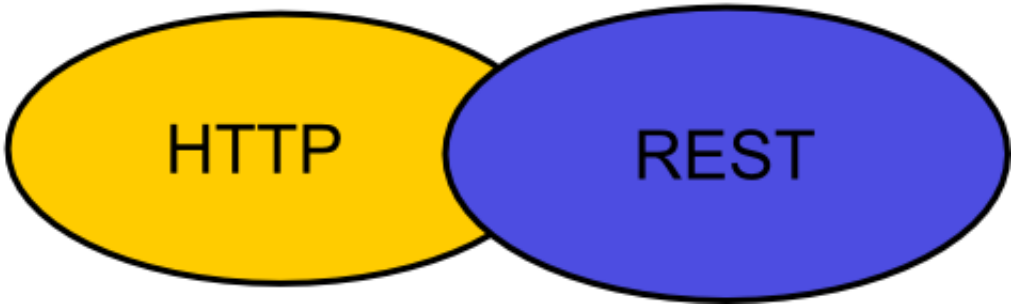
REST versus HTTP

A

B



C

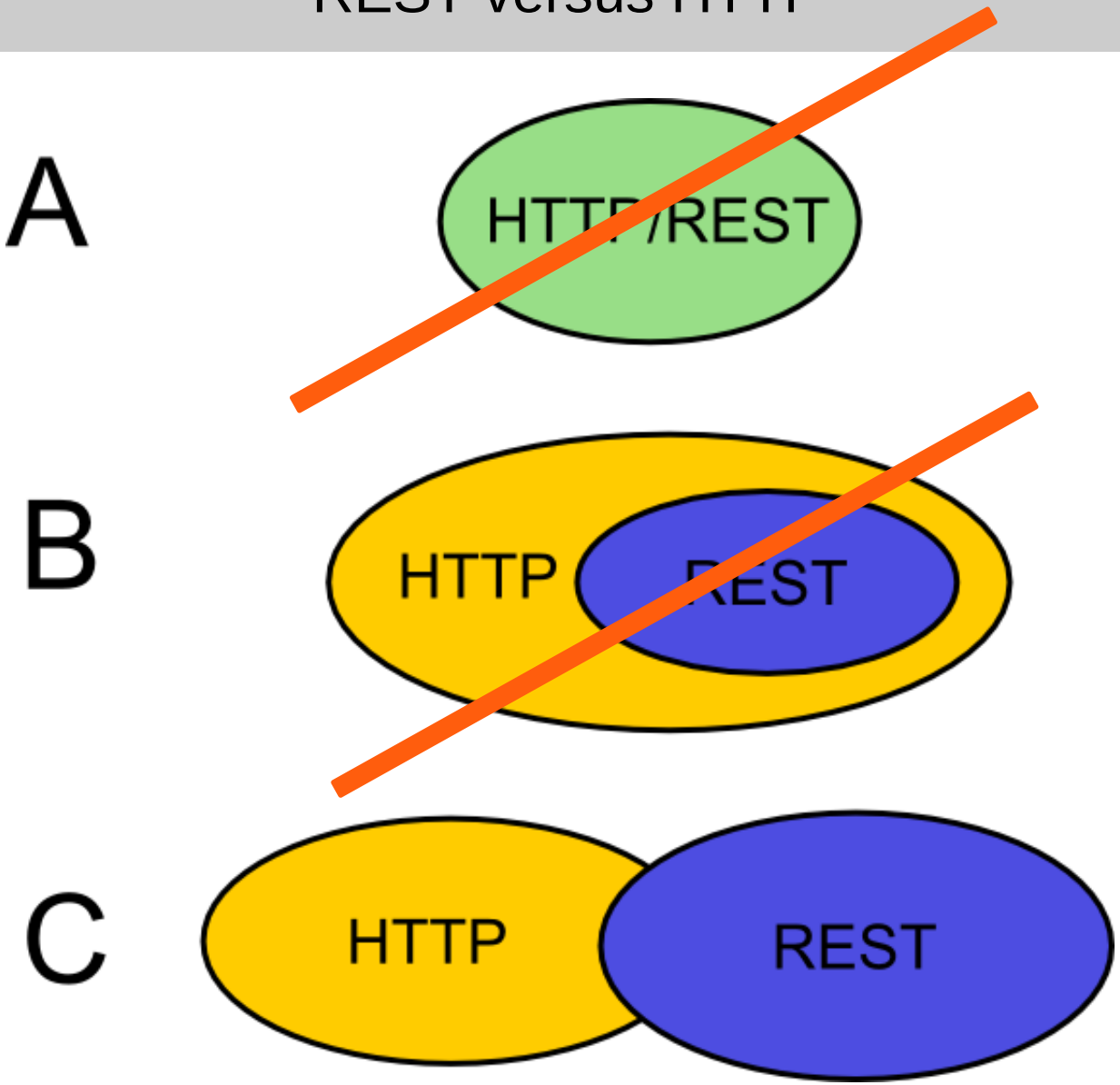


Answer

Last chapter (6/6)

6.3 “REST Applied to HTTP”

REST versus HTTP



Fact A

REST is independent of HTTP

(But they are commonly used together)

Quote

“A REST API should not be dependent on any single communication protocol”

Roy Tomas Fielding

Roy said it in his blog!

http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

What is REST?

REST is a collection of architectural principles

What is REST?

REST is NOT a standard

Standards everywhere

- HTTP 1.0 (RFC 2616)
- SSH (RFC 4251)
- FTP (RFC 959)
- HTML (REC-html401-19991224)
- CSS (REC-CSS2-20080411)
- IMAP (RFC 3051)

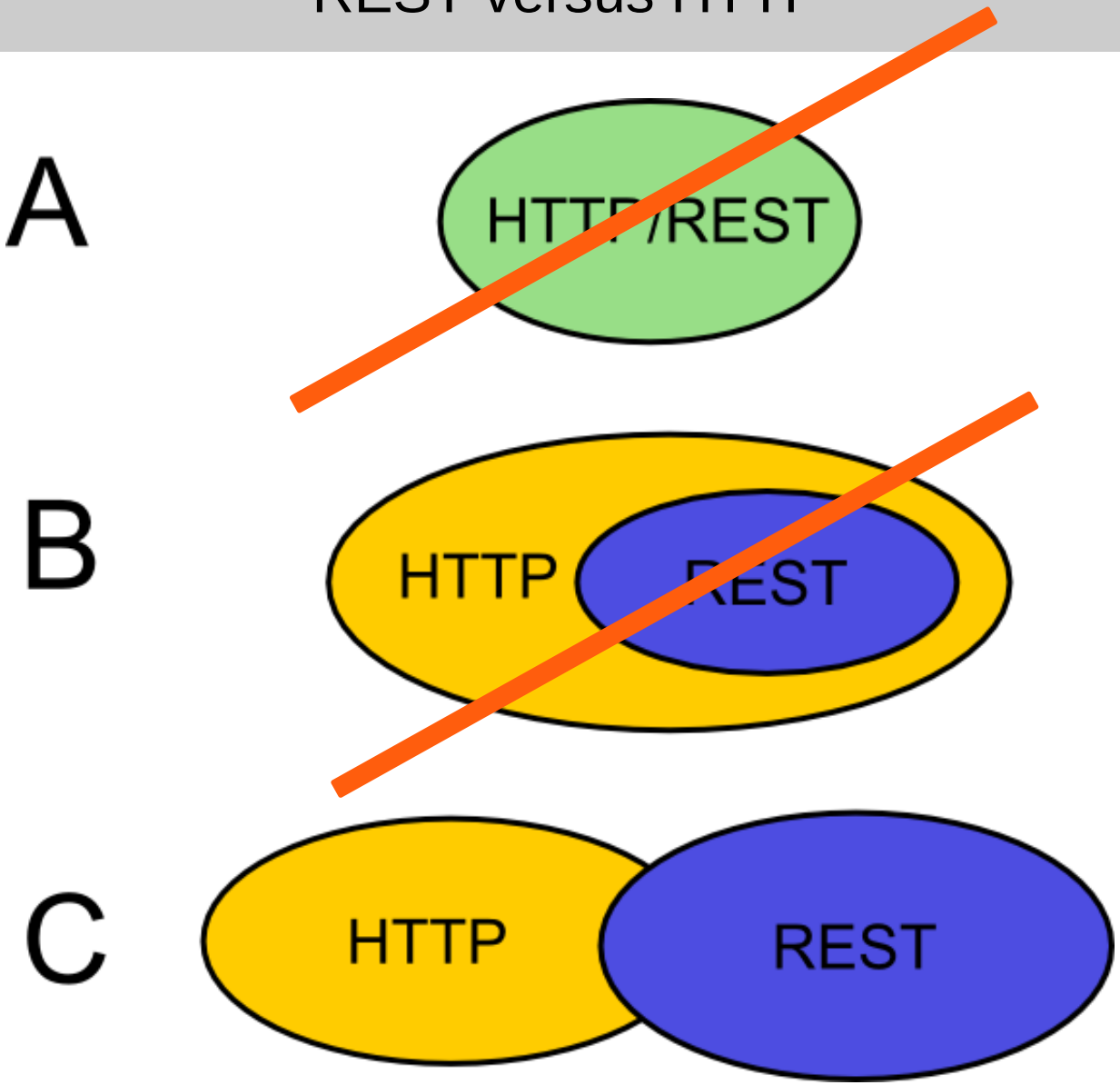
Fact B

There is NO official Specification for REST.

There is no reference implementation for REST

What is REST

REST versus HTTP



WHAT is REST

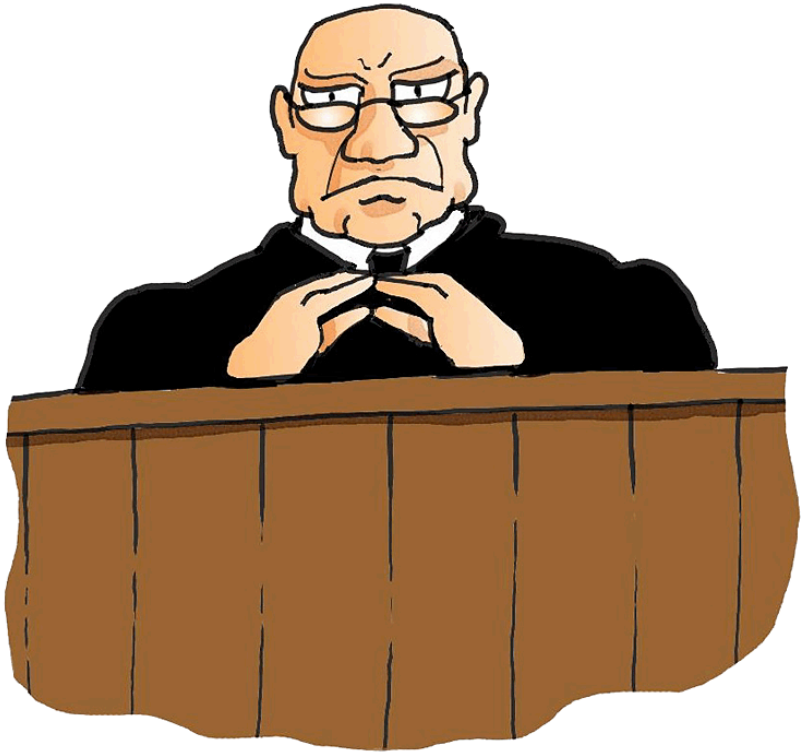
Quiz 2

How many times REST in mentioned in the HTTP 1.1 Spec?

Answer

0 times

Common misconception regarding REST



My service is REST because…
….”X” reason

X = something that is defined by the HTTP SPEC

REJECTED

Your service is NOT REST

And I will show you why….

A REST application

- Client – Server
- Stateless
- Cacheable
- Layered system
- Code on demand
- Uniform interface

Roy Tomas Fielding

Two quotes of Roy, that I really like.

Quote 1

“A REST API should be entered with no prior knowledge beyond the initial URI (bookmark) and set of standardized media types”

Roy Fielding

Quote 2

- “A REST API must not define fixed resource names or hierarchies (an obvious coupling of client and server)”

Roy Fielding

Uniform interface

- “A REST API should be entered with no prior knowledge beyond the initial URI (bookmark) and set of standardized media types”

- “A REST API must not define fixed resource names or hierarchies (an obvious coupling of client and server)”

Flight booking example

This is obviously not REST

- http://myservice/searchFlight
- http://myservice/bookFlight
- http://myservice/cancelFlight

This is obviously not REST

- http://myservice/searchFlight
- http://myservice/bookFlight
- http://myservice/cancelFlight

You have 3 URLS

This is not REST as well

- http://myservice/flights/search
- http://myservice/flights/book
- http://myservice/flights/cancel

This is not REST as well

- http://myservice/flights/search
- http://myservice/flights/book
- http://myservice/flights/cancel

You still have 3 URLS

This is REST

- http://myservice/flights

- Response is: { “search”: “http://myservice/flights/search”, “book”: “http://myservice/flights/book”, “cancel”, “http://myservice/flights/cancel” }

This is REST

- http://myservice/flights

- Response is: { “search”: “http://myservice/flights/search”, “book”: “http://myservice/flights/book”, “cancel”, “http://myservice/flights/cancel” }

You now have one URL

This is not REST as well

- http://myservice/flights
- http://myservice/tickets
- http://myservice/customers

This is not REST as well

- http://myservice/flights
- http://myservice/tickets
- http://myservice/customers

You still have 3 URLS

This is REST

- http://myservice/

- Response is: { “flights”: “http://myservice/flights”, “tickets”: “http://myservice/tickets”, “customers”, “http://myservice/customers” }

This is REST

- http://myservice/

- Response is: { “flights”: “http://myservice/flights”, “tickets”: “http://myservice/tickets”, “customers”, “http://myservice/customers” }

You now have one URL

Naming does not matter

- http://myservice/flights?id=1234
- http://myservice/flights/1234
- http://myservice/flights/v2/1234
- http://v2.myservice/flights

You still have multiple URLS

Naming does NOT matter!

…in a well defined REST API

Salesforce REST API

List Available REST API Versions

Use the Versions resource to list summary information about each REST API version currently available

Example usage

```
curl http://na1.salesforce.com/services/data/
```

Example request body

none required

Example JSON response body

```
[
    {
        "version" : "20.0",
        "label" : "Winter '11",
        "url" : "/services/data/v20.0"
    },
    {
        "version" : "21.0",
        "label" : "Spring '11",
        "url" : "/services/data/v21.0"
    },
    ...
    {
        "version" : "26.0",
        "label" : "Winter '13",
        "url" : "/services/data/v26.0"
    }
]
```

Single URL

SalesForce REST API

List Available REST Resources

Use the Resources by Version resource to list the resources available for th

Example

```
curl https://na1.salesforce.com/services/data/v26.0/
```

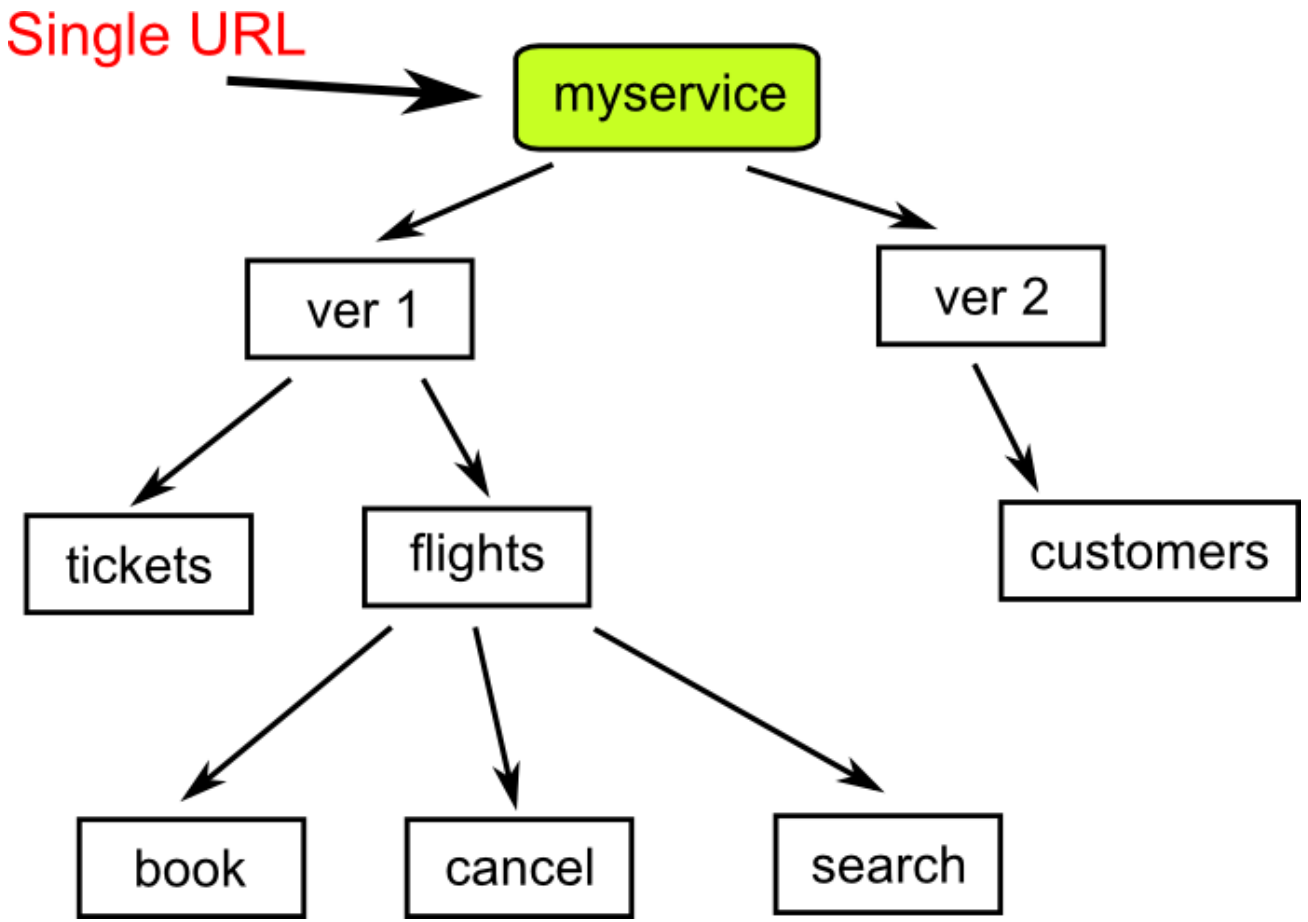
Example request body

none required

Example JSON response body

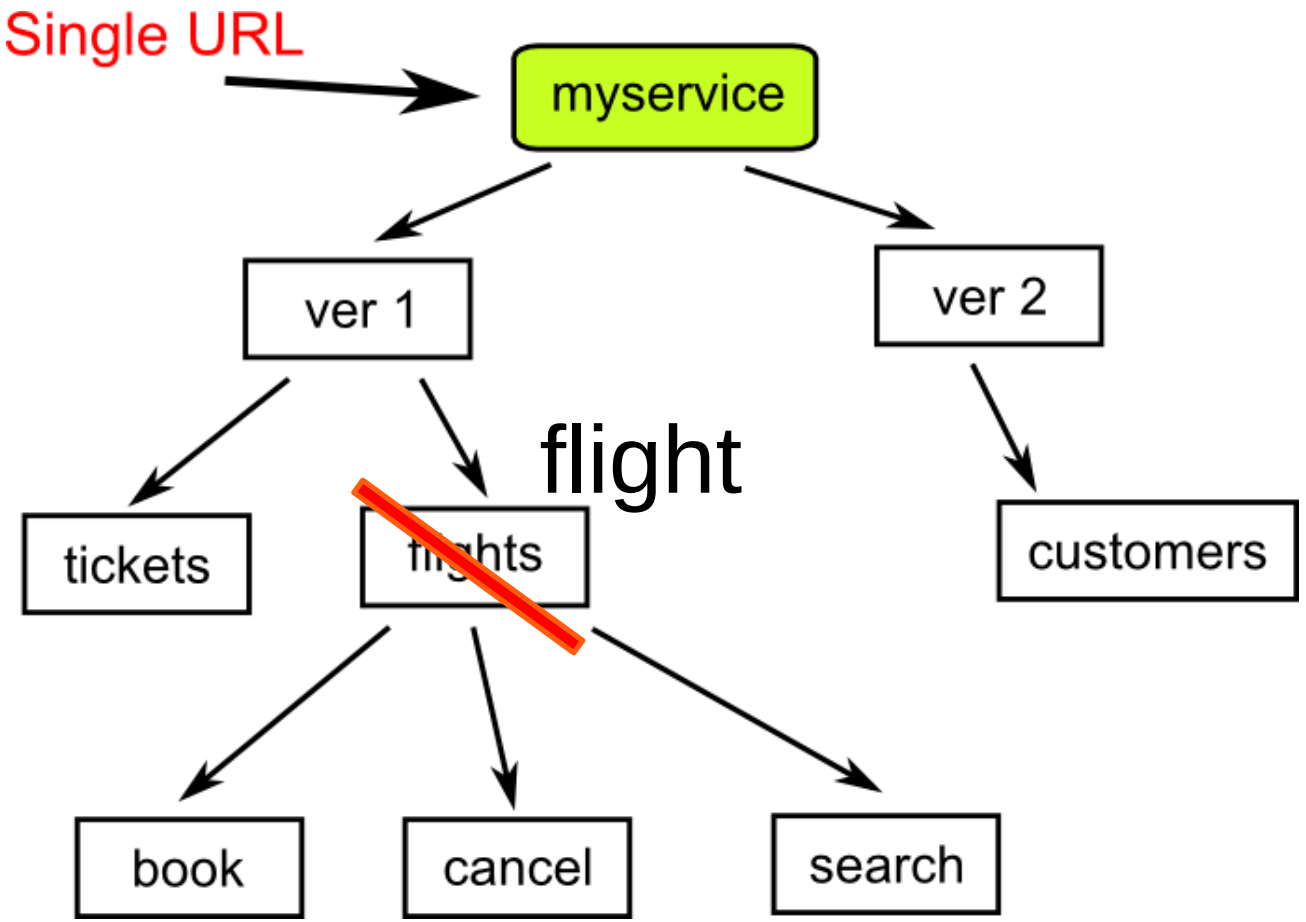
```
{
    "sobjects" : "/services/data/v26.0/sobjects",
    "licensing" : "/services/data/v26.0/licensing",
    "connect" : "/services/data/v26.0/connect",
    "search" : "/services/data/v26.0/search",
    "query" : "/services/data/v26.0/query",
    "tooling" : "/services/data/v26.0/tooling",
    "chatter" : "/services/data/v26.0/chatter",
    "recent" : "/services/data/v26.0/recent"
}
```

True REST hierarchy



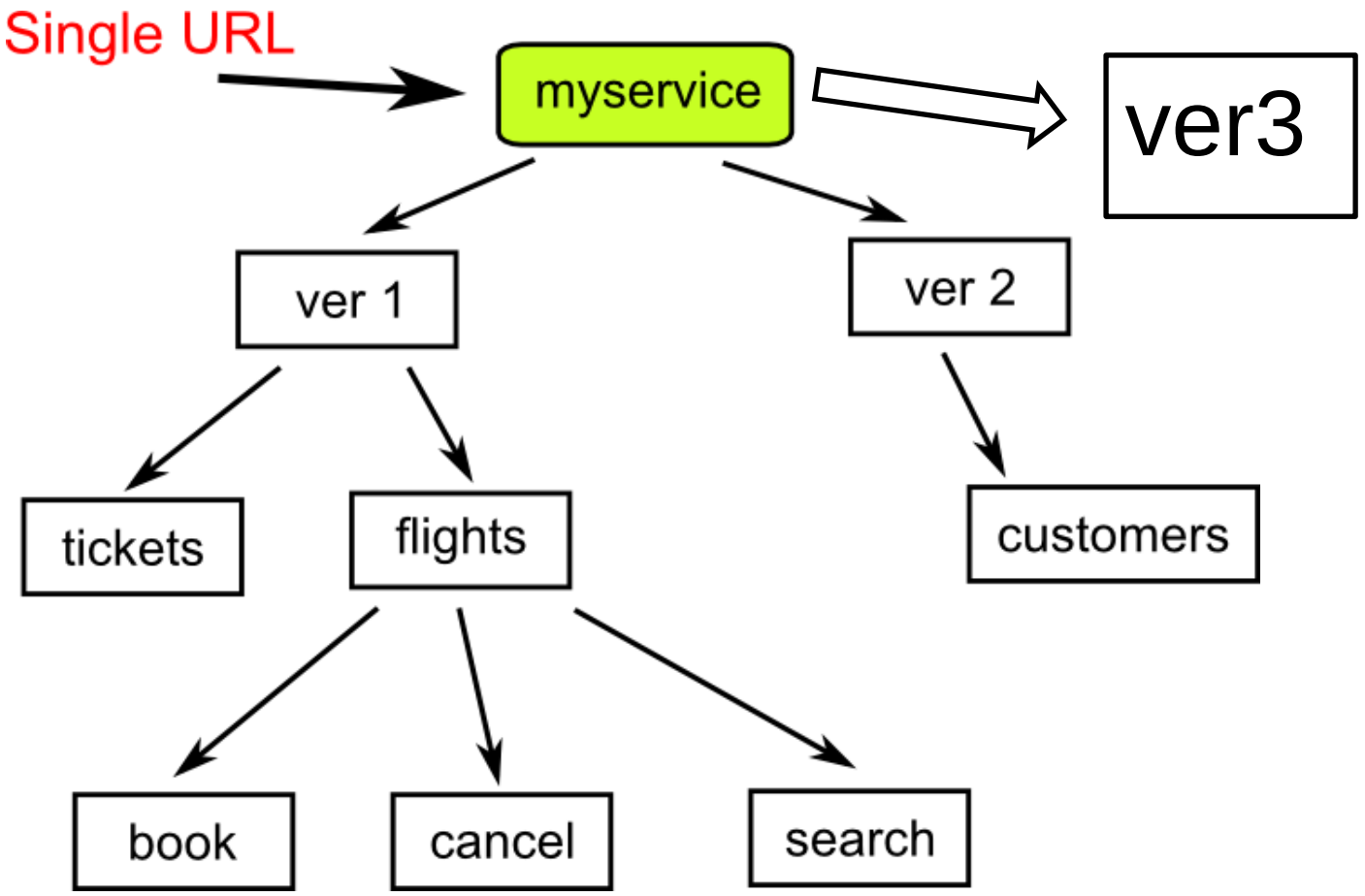
A true REST service is discoverable

True REST hierarchy



A true REST service does not care about naming

True REST hierarchy



A true REST service is extensible

Web endpoints

Surfing the web

1. You go to CNN.com (starting URL)
2. You look at the webpage and decide you want “Entertainment”
3. You click on it.
4. You go to cnn.com/en, or cnn.com/entertainment, or cnn.com/pageid=34 etc.

5. As a human you only remember cnn.com
6. You might bookmark cnn.com/en (but you don’t expect it to be set in stone)

Surfing the web - wrong way

You have as bookmark cnn.com/en, cnn.com/weather, cnn.com/politics, cnn.com/health, cnn.com/news, cnn.com/blogs, cnn.com/sports, cnn.com/tech, cnn.com/world, cnn.com/tv, cnn.com/style, cnn.com/travel, cnn.com/money, cnn.com/search, cnn.com/login, edition.cnn.com, cnn.com/europe, cnn.com/africa, cnn.com/china, cnn.com/asia, cnn.com/us, cnn.com/opinions, cnn.com features

CNN cannot change its page structure because users will get upset.

Fact C

If your service has more than one URLs (or needs an SDK) you are NOT REST compliant!

… and your service is just plain HTTP

Roy Tomas Fielding

Part of the HTTP1.0 team

Part of the HTTP1.1 team

Author of “REST” PHD

Your service is not REST

If you argue about naming, your service is probably not REST

Your service is not REST

If your API documentation is 95% definitions and 5% examples, you are probably not REST

Ideal documentation of a REST service

Ideal documentation of a REST service

Your API documentation should be 5% definitions and 95% examples

Quote

“I bet that 95% of self-proclaimed REST APIs are not actually REST”

Quote

“I bet that 95% of self-proclaimed REST APIs are not actually REST”

Kostis Kapelonis

Quote

“I am getting frustrated by the number of people calling any HTTP-based interface a REST API.”

Roy Tomas Fielding

Quote by Roy Tomas Fielding!

Your Service is not REST

If you need more than one URL in your HTTP service you are not REST compliant

Let that sink, for a moment

…because your service is not discoverable

Let that sink, for a moment

Your Service is not REST

…because clients are tied to your hierarchy

Let that sink, for a moment

Your Service is not REST

…because it is impossible to change the names of resources

Let that sink, for a moment

Your Service is not REST

…and because every time you add /remove something all clients need upgrading. (Doh!)

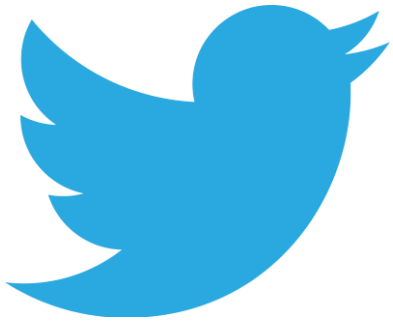
Let that sink, for a moment

The Twitter “REST” API

- GET statuses/mentions_timeline
- GET statuses/user_timeline
- GET statuses/home_timeline
- GET statuses/retweets_of_me
- GET statuses/retweets/:id
- GET statuses/show/:id
- POST statuses/destroy/:id
- POST statuses/update
- POST statuses/retweet/:id
- POST statuses/update_with_media
- GET statuses/oembed
- GET statuses/retweeters/ids
- GET statuses/lookup
- POST media/upload
- GET direct_messages/sent
- GET direct_messages/show
- GET search/tweets
- GET direct_messages
- POST direct_messages/destroy
- POST direct_messages/new
- GET friendships/no_retweets/ids
- GET friends/ids

Resource Information

Response formats	JSON
Requires authentication?	Yes (user context only)
Rate limited?	Yes



Parameters

id
required

The ID of the direct message to delete.

Example Values: `1270516771`

include_entities
optional

The `entities` node will not be included when set to `false`.

Example Values: `false`

OAuth Signature Generator

You have not registered any apps. Head on over to Manage Your Apps to create an app to get started!

Example Request

POST
`https://api.twitter.com/1.1/direct_messages/destroy.json?`

REST??

TRASYS
WE GET IT DONE

Error codes in REST

Error handling in REST applications

Error handling in ~~REST~~ HTTP applications

Error handling in ~~REST~~ HTTP1.1 applications

First Approach

Use HTTP status codes (duh!)

HTTP errors

- 200 OK
- 400 bad request
- 401 Unauthorised
- 404 Not found
- 406 Not acceptable
- 413 Request entity too large
- 500 Internal server error

Error 407

"Proxy authentication required"

Error 415

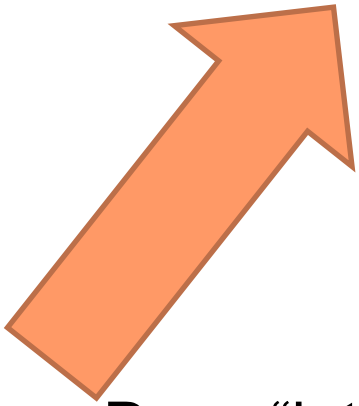
"Unsupported media type"

Error 426

"Upgrade required (TLS)"

Error 500

"Internal Server error"



Does "internal" include my application? My VM?

Error 503

Service Unavailable

“The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. The implication is that this is a temporary condition which will be alleviated after some delay.”

Error 504

“Gateway Timeout”

WTF?

I am a developer. Why should I bother with these error codes?

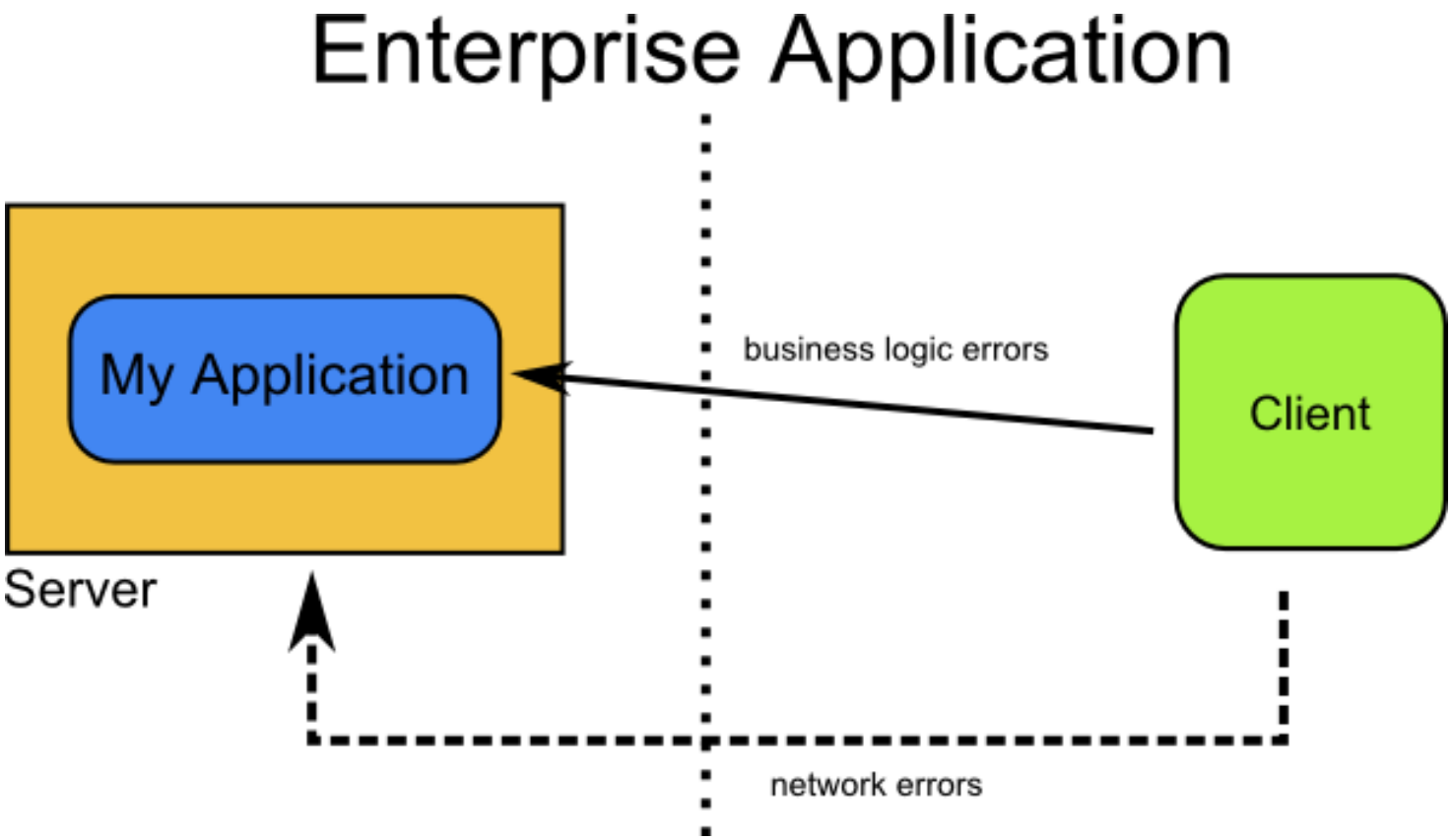
Maybe a system administrator must see those errors..

The HTTP error codes

- 1xx informational
- 2xx success
- 3xx Redirection
- 4xx Client Error
- 5xx Server error

Where is Application error?

Another view of error handling



How to distinguish these two kinds of errors?

How would you map these?

- Flight delayed
- Flight overbooked
- Flight full
- Flight canceled
- Flight rescheduled
- Only business seats free

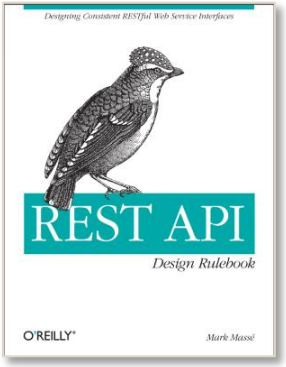
HTTP error codes are confusing

Rule: 302 ("Found") should not be used

The intended semantics of the 302 response code have been misunderstood by programmers and incorrectly implemented in programs since version 1.0 of the HTTP protocol.† The confusion centers on whether it is appropriate for a client to always automatically issue a follow-up GET request to the URI in response's Location header, regardless of the original request's method. For the record, the intent of 302 is that this automatic redirect behavior only applies if the client's original request used either the GET or HEAD method.

To clear things up, HTTP 1.1 introduced status codes 303 ("See Other") and 307 ("Temporary Redirect"), either of which should be used instead of 302.

REST API Design
Rulebook O'Reilly 2011

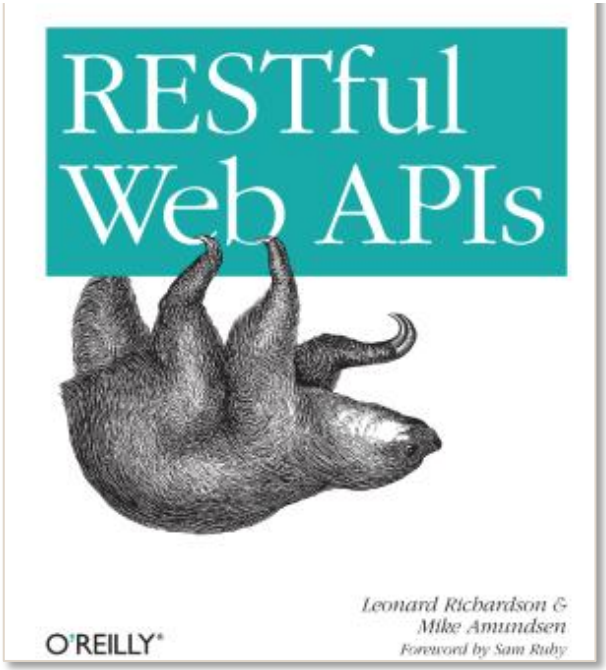


HTTP error codes are confusing

That said, some of the HTTP status codes are completely useless. Some are useful only in very limited situations, and some are only distinguishable from one another by careful hairsplitting. To someone used to the World Wide Web (that's all of us), the variety of status codes can be bewildering.

"Useless, Hairsplitting, bewildering"

RESTful Web APIs
O'Reilly 2013



HTTP error codes are confusing

Four Status Codes: The Bare Minimum

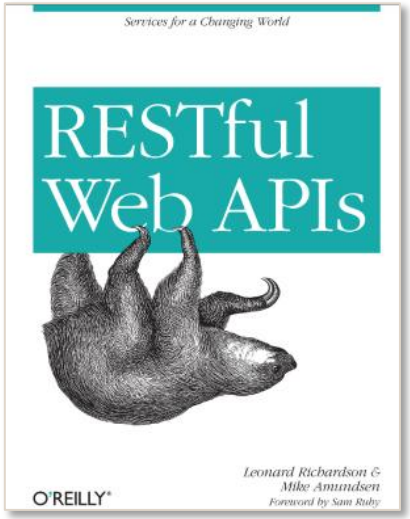
Before going through the big list of status codes, I want to list just four that I consider the bare minimum for APIs. There's one code from each family (apart from 1xx, which you can more or less ignore):

200 (`OK`)
: Everything's fine. The document in the entity-body, if any, is a representation of some resource.

301 (`Moved Permanently`)
: Sent when the client triggers a state transition that moves a resource from one URL to another. After the move, requests to the old URL will also result in a 301 status code.

400 (`Bad Request`)
: There's a problem on the client side. The document in the entity-body, if any, is an error message. Hopefully the client can understand the error message and use it to fix the problem.

500 (`Internal Server Error`)
: There's a problem on the server side. The document in the entity-body, if any, is an error message. The error message probably won't do much good, since the client can't fix a server problem.



How would you map these?

- 500 -Flight delayed
- 500 - Flight overbooked
- 500 - Flight full
- 500 - Flight canceled
- 500 - Flight rescheduled
- 500 - Only business seats free

Companies have “extended” http codes

440 Login Timeout (Microsoft)

A Microsoft extension. Indicates that your session has expired.[20]

444 No Response (Nginx)

Used in Nginx logs to indicate that the server has returned no information to the client and closed the connection (useful as a deterrent for malware).

449 Retry With (Microsoft)

A Microsoft extension. The request should be retried after performing the appropriate action.[21]

450 Blocked by Windows Parental Controls (Microsoft)

A Microsoft extension. This error is given when Windows Parental Controls are turned on and is blocking access to the given webpage.[22]

451 Unavailable For Legal Reasons (Internet draft)

Defined in the internet draft "A New HTTP Status Code for Legally-restricted Resources".[23] Intended to be used when resource access is denied for le
blocked access. A reference to the 1953 dystopian novel *Fahrenheit 451*, where books are outlawed.[24]

451 Redirect (Microsoft)

Used in Exchange ActiveSync if there either is a more efficient server to use or the server cannot access the users' mailbox.[25]
The client is supposed to re-run the HTTP Autodiscovery protocol to find a better suited server.[26]

494 Request Header Too Large (Nginx)

Nginx internal code similar to 431 but it was introduced earlier in version 0.9.4 (on January 21, 2011).[27][*original research?*]

495 Cert Error (Nginx)

Nginx internal code used when SSL client certificate error occurred to distinguish it from 4XX in a log and an error page redirection.

496 No Cert (Nginx)

Nginx internal code used when client didn't provide certificate to distinguish it from 4XX in a log and an error page redirection.

497 HTTP to HTTPS (Nginx)

Nginx internal code used for the plain HTTP requests that are sent to HTTPS port to distinguish it from 4XX in a log and an error page redirection.

498 Token expired/invalid (Esri)

Returned by ArcGIS for Server. A code of 498 indicates an expired or otherwise invalid token.[28]

499 Client Closed Request (Nginx)

Used in Nginx logs to indicate when the connection has been closed by client while the server is still processing its request, making server unable to ser

499 Token required (Esri)

Returned by ArcGIS for Server. A code of 499 indicates that a token is required (if no token was submitted).[28]

Microsoft

Nginx

Esri (GIS)

Another suggestion

Decouple HTTP error codes for your application

Use 200 OK 99%

This means that the client request has reached your application (and network is ok)

Include the error code inside the response

Receiving Error Codes

The following represents a common error response resulting from a failed API request:

```
{
  "error": {
    "message": "Message describing the error",
    "type": "OAuthException",
    "code": 190 ,
    "error_subcode": 460
  }
}
```

Major benefits

- Easy to understand network failures vs development errors (Very important)
- Unlimited codes
- Unlimited ways to return errors (simple string or big object)
- Can clearly define retry-time out and back off policy

Adobe flash application

Adobe Flash intercepted all non-200 responses

Old Twitter REST API

Twitter had a magic “suppress_response_code” parameter to always return 200 OK

Old Facebook REST API

Facebook at some point used to return always 200 OK

Facebook API

The “REST API” is obsolete. (December 2011)

New API is “Graph API”

https://developers.facebook.com/blog/post/616/

Facebook ads error codes

Search Facebook Developers | Docs | Tools | Support | News | Apps | Log In

1487133	Invalid Negative Connections: If you specify negative targeting, you must be the administrator or developer of the objects whose fans you want to specify to exclude. You are not an admin of the following specified connections: {connections}
1487174	Invalid Image Hash: Invalid Image Hash - {hash}
1487194	Permission Error: Either the object you are trying to access is not visible to you or the action you are trying to take is restricted to certain account types.
1487199	Ad targeting does not match targeting of the story: The targeting specified for this ad is not compatible with the story being boosted. Check the privacy and language/country targeting of the story you are trying to sponsor.
1487202	Invalid object - not admin or object not public: The user is not an admin of the object or the object is not publicly accessible.
1487211	Invalid URL For Creative Destination: Creative must have a valid destination URL, and if attached object is page, destination must match page.
1487225	Adgroup Creation Limited By Daily Spend: The number of adgroups you can create in a given period of time has a limit determined by your daily spend level. Higher spend levels allow creation of more adgroups. Increase your daily spend limit or create fewer ads per time period.
1487244	Campaign Update Failed: Campaign {campaign_id}: {reason}
1487246	Campaign Creation Failed: {reason}
1487256	Targeting declined due to policy: Invalid ads targeting. The targeting spec was declined due to policy restrictions.
1487283	Not Allowed To Use View Tags: Only some partners are allowed to use view tags. Please verify that you are using an approved account.
1487301	Custom Audience Unavailable: The custom audience you're trying to use hasn't been shared with your ad account. Please create or choose a different custom audience, or ask the owner of the custom audience to let you use it.

Hybrid approach

Use both http codes and extra business codes

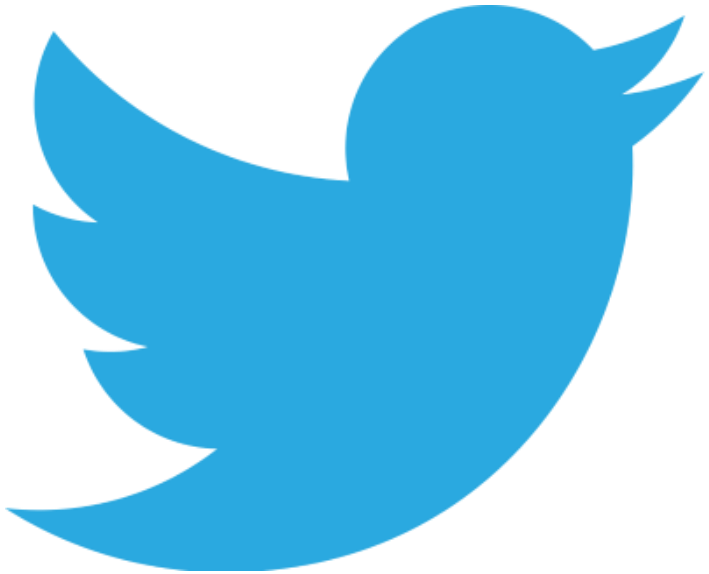
Twitter Rest API error codes

```
{"errors":[{"message":"Sorry, that page does not
exist","code":34}]}
```

Error Codes

In addition to descriptive error text, error messages contain machine-parseable codes. While the text for an error message may change, the codes will stay the same. The following table describes the codes which may appear when working with the API:

Code	Text	Description
32	Could not authenticate you	Your call could not be completed as dialed.
34	Sorry, that page does not exist	Corresponds with an HTTP 404 - the specified resource was not found.
64	Your account is suspended and is not permitted to access this feature	Corresponds with an HTTP 403 — the access token being used belongs to a suspended user and they can't complete the action you're trying to take
68	The Twitter REST API v1 is no longer active. Please migrate to API v1.1. https://dev.twitter.com/rest/public	Corresponds to a HTTP request to a retired v1-era URL.
88	Rate limit exceeded	The request limit for this resource has been reached for the current rate limit window



Paypal does it in another way

API

Paypal has a hybrid approach (both types of error codes)

Hybrid Approach by Paypal

Errors

PayPal uses standard HTTP status codes when returning errors. Additionally, we provide details about errors in the body of the response in the following format:

```
{
  "name": "{ERROR NAME)",
  "message": "{Error description}",
  "information_link": "{Link to error documentation}",
  "details": "[Additional details about the error]"
}
```

Error List

The following is a list of errors related to the REST API. We provide corrective action where available.

INTERNAL_SERVICE_ERROR

An internal service error has occurred

Resend the request at another time. If this error continues, contact PayPal Merchant Technical Support.

VALIDATION_ERROR

Invalid request

There was a validation issue with your request - see details

HTTP Status Codes

- `200` - Request OK
- `201` - Resource created
- `401` - Unauthorized request
- `402` - Failed request
- `403` - Forbidden
- `404` - Resource was not found
- `50x` - PayPal server error

Actual Error codes

Amazon does it in another way

API

Amazon has also a hybrid approach (both types of error codes)

Hybrid Approach by Amazon

Amazon Simple Storage Service
API Reference (API Version 2006-03-01)

Search: Documentation - This Guide

The following table lists Amazon S3 error codes.

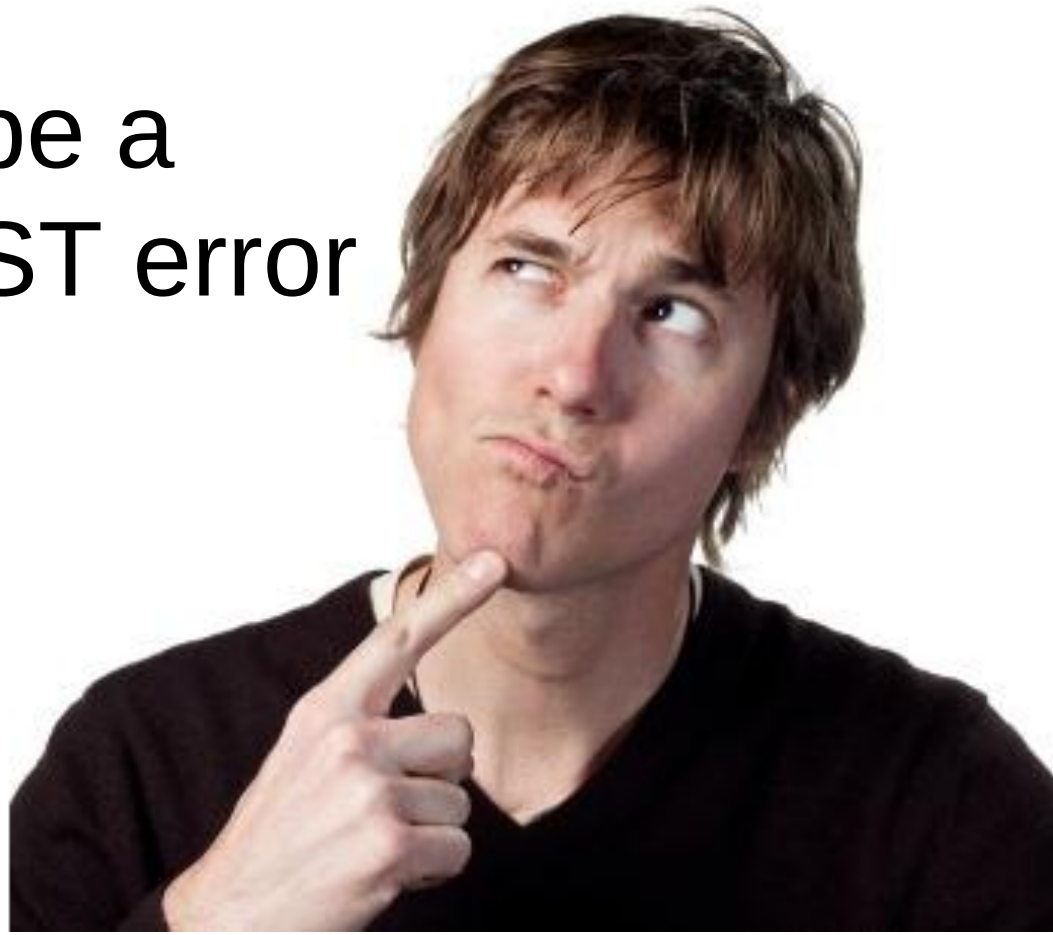
Error Code	Description	HTTP Status Code
AccessDenied	Access Denied	403 Forbidden
AccountProblem	There is a problem with your AWS account that prevents the operation from completing successfully. Please use Contact Us.	403 Forbidden
AmbiguousGrantByEmailAddress	The e-mail address you provided is associated with more than one account.	400 Bad Request
BadDigest	The Content-MD5 you specified did not match what we received.	400 Bad Request
BucketAlreadyExists	The requested bucket name is not available. The bucket namespace is shared by all users of the system. Please select a different name and try again.	409 Conflict
BucketAlreadyOwnedByYou	Your previous request to create the named bucket succeeded and you already own it.	409 Conflict
BucketNotEmpty	The bucket you tried to delete is not empty.	409 Conflict
CredentialsNotSupported	This request does not support credentials.	400 Bad Request
CrossLocationLoggingProhibited	Cross location logging not allowed. Buckets in one geographic location cannot log information to a bucket in another location.	403 Forbidden
EntityTooSmall	Your proposed upload is smaller than the minimum allowed object size.	400 Bad Request
EntityTooLarge	Your proposed upload exceeds the maximum allowed object size.	400 Bad Request
ExpiredToken	The provided token has expired.	400 Bad Request
IllegalVersioningConfigurationException	Indicates that the Versioning configuration specified in the request is invalid.	400 Bad Request
IncompleteBody	You did not provide the number of bytes specified by the Content-Length HTTP header	400 Bad Request

Business error

HTTP Error

There is no standard way

Shouldn’t there be a standard for REST error handling?



There is no standard way

JSend

- **What?** - Put simply, JSend is a specification that lays down some rules for how ⇨ JSON responses from web servers should be formatted. JSend focuses on application-level (as opposed to protocol- or transport-level) messaging which makes it ideal for use in ⇨ REST-style applications and APIs.
- **Why?** - There are lots of web services out there providing JSON data, and each has its own way of formatting responses. Also, developers writing for JavaScript? front-ends continually re-invent the wheel on communicating data from their servers. While there are many common patterns for structuring this data, there is no consistency in things like naming or types of responses. Also, this helps promote happiness and unity between backend developers and frontend designers, as everyone can come to expect a common approach to interacting with one another.
- **Hold on now, aren't there already specs for this kind of thing?** - Well... no. While there are a few handy specifications for dealing with JSON data, most notably ⇨ Douglas Crockford's ⇨ JSONRequest proposal, there's nothing to address the problems of general application-level messaging. More on this later.
- **(Why) Should I care?** - If you're a library or framework developer, this gives you a consistent format which your users are more likely to already be familiar with, which means they'll already know how to consume and interact with your code. If you're a web app developer, you won't have to think about how to structure the JSON data in your application, and you'll have existing reference implementations to get you up and running quickly.
- **Discuss** - ⇨ Mailing list

http://labs.omniti.com/labs/jsend

3 Points to Take Away

Point 1

REST is independent from HTTP. HTTP is independent from REST. (but they can be used together)

Point 2

A true REST API has only one URL. No other URL is known in advance.

Point 3

Use custom error codes for your business.

Make HTTP error codes secondary…

Thank you

- http://www.lornajane.net/posts/2013/five-clues-that-your-api-isnt-restful
- http://blog.theamazingrando.com/posts/2009/your-web-service-might-not-be-restful-if.html
- http://vvv.tobiassjosten.net/development/your-api-is-not-restful/
- http://www.infoq.com/articles/web-api-rest

http://trasys.be

http://codepipes.com

http://twitter/codepipes

http://manning.com/kapelonis

Backup slides

About You

Hands up

Have you written code that consumes a REST service?

Hands up

Have you implemented yourself a REST service?

Hands up

Have you implemented yourself an HTTP service?

About You

WTF?

Hands up

Do you know the term RESTifarian?