DESIGNING HTTP URLS AND REST INTERFACES





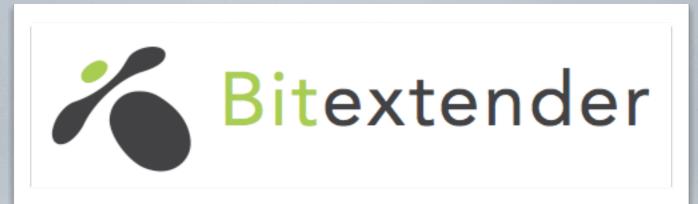
David Zuelke





http://en.wikipedia.org/wiki/File:München_Panorama.JPG

Founder



Lead Developer



<u>@dzuelke</u>

THE OLDEN DAYS

Before REST Was En Vogue

http://www.acme.com/index.php?action=zomg&page=lol

along came



and said

THROUGH.

at least if they were



so we had to change this

http://www.acme.com/zomg/lol

and then things got out of control

because nobody really had a clue

http://acme.com/videos/latest/hamburgers

http://acme.com/search/lolcats/pictures/yes/1/200

oh dear...

ALONG CAME ROY FIELDING

And Gave Us REST

that was awesome

because everyone could say



I haz REST nao

when in fact

they bloody didn't

REST

What Does That Even Mean?

REpresentational State Transfer

- A URL identifies a Resource
- Resources have a hierarchy
 - so you know that something with additional slashes is a subordinate resource
- Methods perform operations on resources
- The operation is implicit and **not** part of the URL
- · A hypermedia format is used to represent the data
- · Link relations are used to navigate a service

and most importantly

a web page is not a resource

it is a representation of a resource

GETTING JSON BACK

```
GET /products/ HTTP/1.1
Host: acme.com
Accept: application/json
```

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Allow: GET, POST

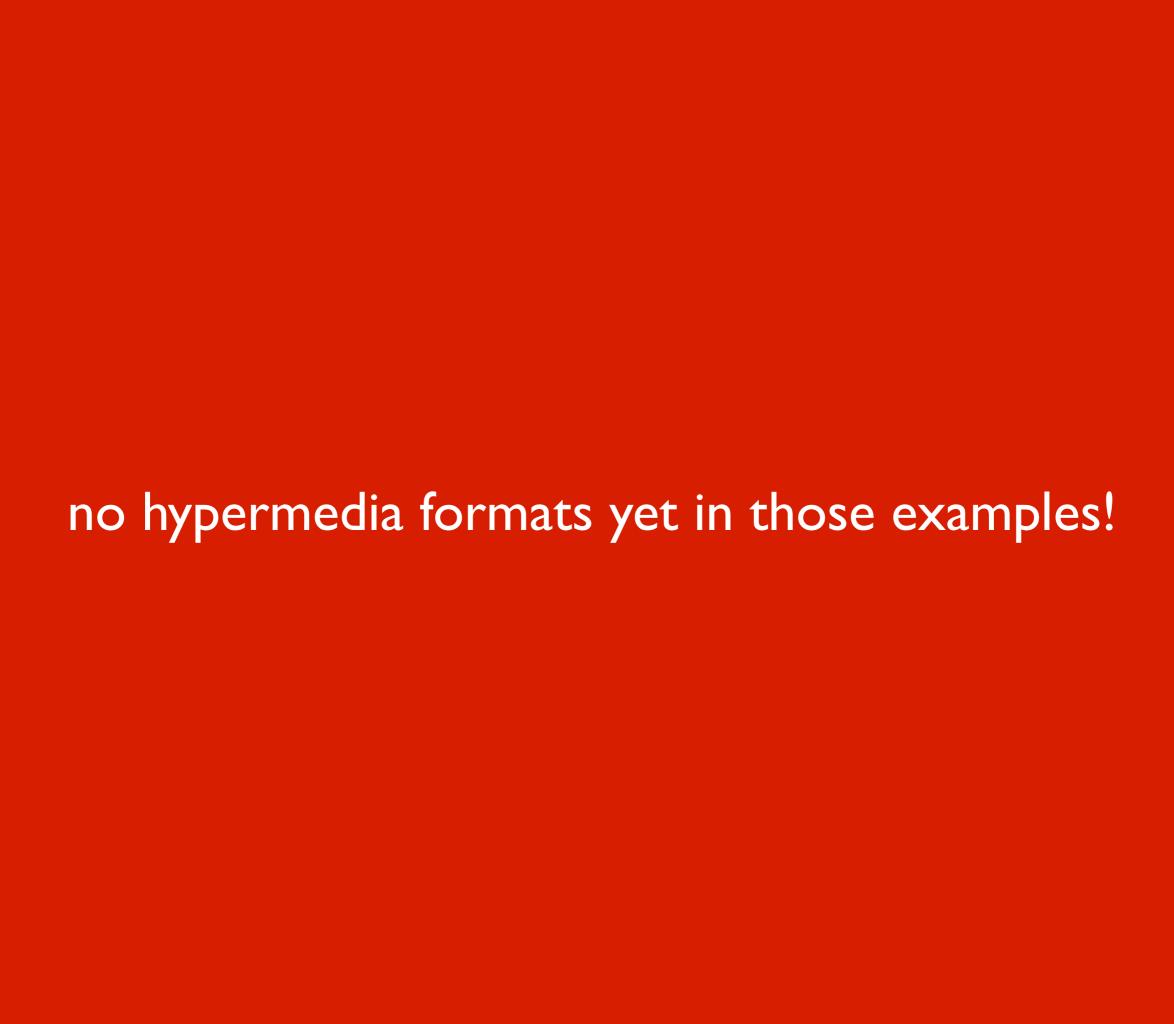
[
 id: 1234,
 name: "Red Stapler",
 price: 3.14,
 location: "http://acme.com/products/1234"
 }
]

GETTING XML BACK

GET /products/ HTTP/1.1

Host: acme.com

Accept: application/xml



I will show that in a few minutes

AND FINALLY, HTML

```
GET /products/ HTTP/1.1
Host: acme.com
Accept: application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,*/*;q=0.5
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_8; en-us) AppleWebKit...
```

A FEW EXAMPLES

Let's Start With Proper URL Design

BAD URLS

- http://www.acme.com/product/
- http://www.acme.com/product/filter/cats/desc
- http://www.acme.com/product/1234 WTF
- http://www.acme.com/photos/product/1234
- http://www.acme.com/photos/product/1234/new) sausage 17?

new what?

• http://www.acme.com/photos/product/1234/5678

G()()1) URIS

- http://www.acme.com/products/
 http://www.acme.com/products/
 filtering is a query
- http://www.acme.com/products/?filter=cats&sort=desc
- http://www.acme.com/products/1234
 http://www.acme.com/products/1234/photos/

 - http://www.acme.com/products/1234/photos/?sort=latest
 - http://www.acme.com/products/1234/photos/5678

THE NEXT LEVEL

Time To Throw CRUD Into The Mix

COLLECTION OPERATIONS

- http://www.acme.com/products/
 - GET to retrieve a list of products
 - POST to create a new product
 - returns
 - 201 Created
 - Location: http://www.acme.com/products/1235

ITEM OPERATIONS

- http://www.acme.com/products/1234
 - GET to retrieve
 - PUT to update
 - DELETE to, you guessed it, delete

(bonus points if you spotted the CRUD there)

HATEOAS

The Missing Piece in the Puzzle

ONE LAST PIECE IS MISSING

- How does a client know what to do with resources?
- How do you go to the "next" operation?
- What are the URLs for creating subordinate resources?
- Where is the contract for the service?

HYPERMEDIA AS THE ENGINE OF APPLICATION STATE

- Use links to allow clients to discover locations and operations
- · Link relations are used to express the possible options
- · Clients do not need to know URLs, so they can change
- The entire application workflow is abstracted, thus changeable
- The hypermedia type itself can be versioned if necessary
- No breaking of clients if the implementation is updated!

XHTML and Atom are Hypermedia formats

Or you roll your own...

A CUSTOM MEDIATYPE

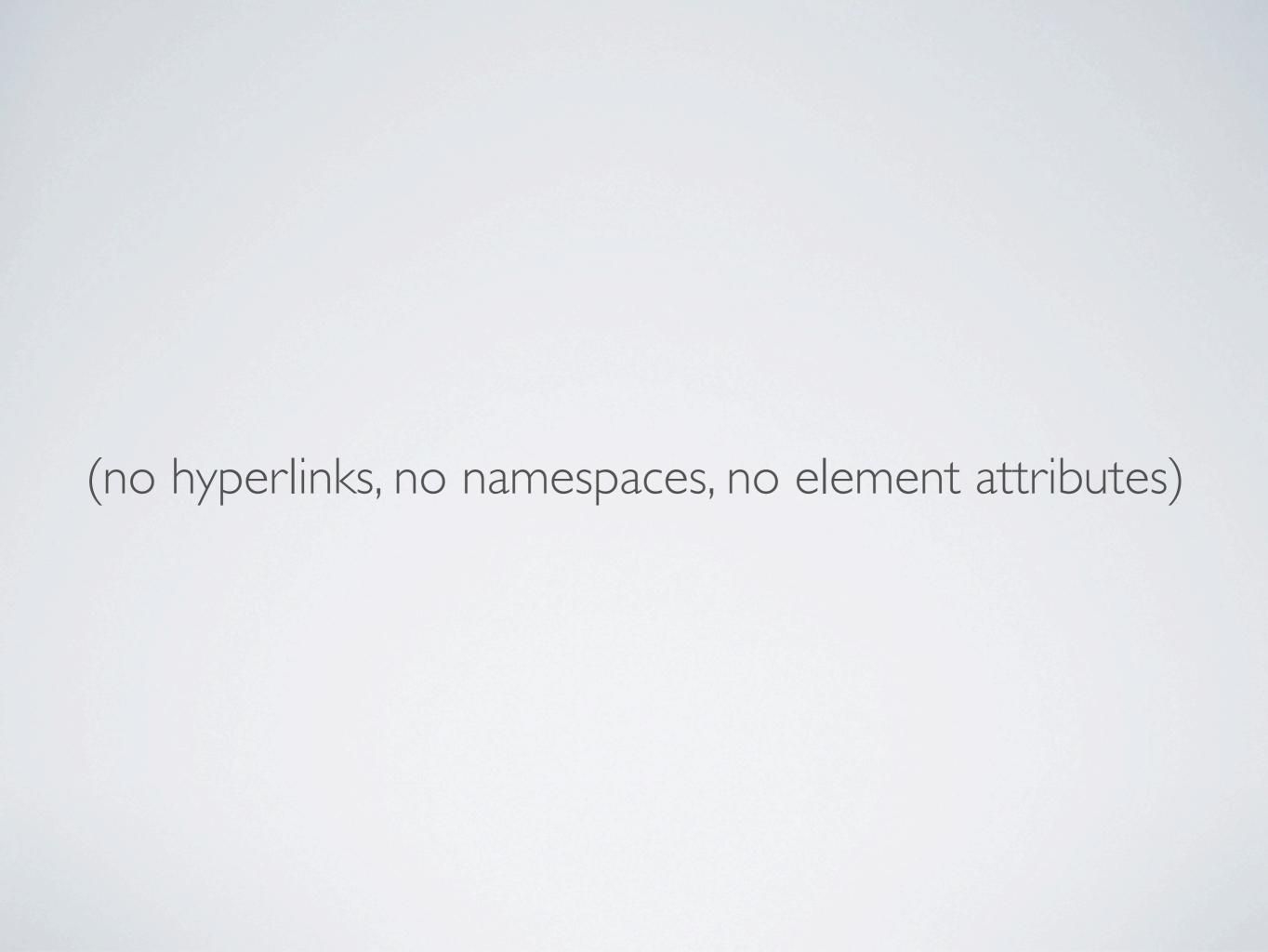
```
GET /products/1234 HTTP/1.1
Host: acme.com
Accept: application/vnd.acmecorpshop+xml
                                                            re-use Atom tor
                                                              link relations
HTTP/1.1 200 OK
Content-Type: application/vnd.acmecorpshop+xml; charset=utf-8
Allow: GET, PUT, DELETE
<?xml version="1.0" encoding="utf-8"?>
 duct xmlns="urn:com.acme.prods" xmlns:atom="http://www.w3.org/2005/xlink">
  <id>1234</id>
  <name>Red Stapler</name>
  <price currency "EUR">3.14</price>
  <atom:link(rel="payment) type="application/vnd.acmecorpshop+xml"
             href "http://acme.com/products/1234/payment"/>
 </product>
```

meaning defined in Atom standard!

XML is really good for hypermedia formats



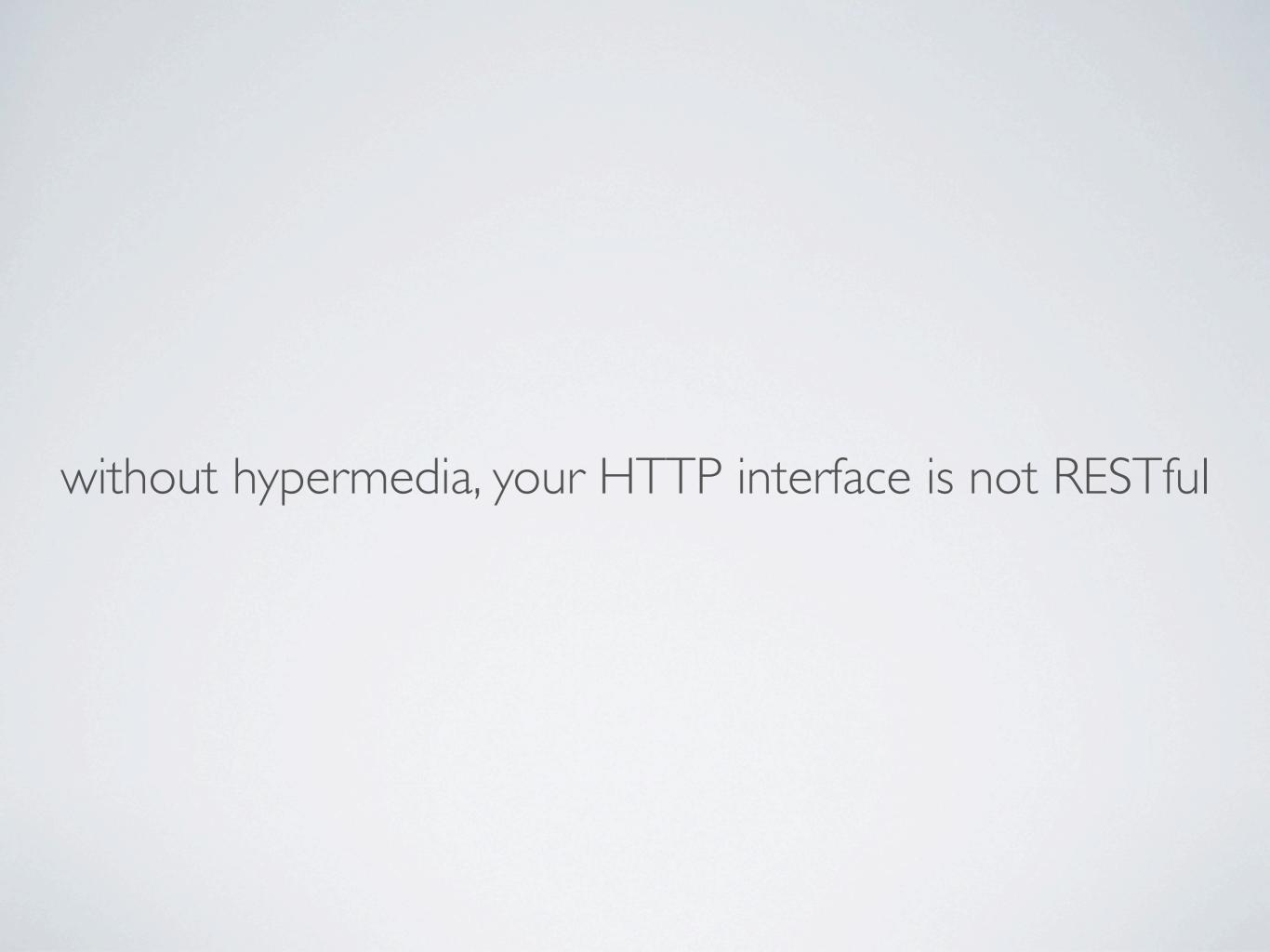
JSON is more difficult



XML VERSUS JSON

```
id: 1234,
name: "Red Stapler",
price: {
   amount: 3.14,
   currency: "EUR"
},
links: [
   {
     rel: "payment",
     type: "application/vnd.acmecorpshop+xml",
     href: "http://acme.com/products/1234/payment"
}
```

and hey



that's totally fine and sometimes even the only way to do it

(e.g. CouchDB or S3 are never going to be RESTful)

but don't you dare call it a RESTful interface

YOU MIGHT BE WONDERING

Why Exactly Is This Awesome?

because it scales

not just terms of performance

but also in how you can extend and evolve it



it's completely seamless

all thanks to the polymorphism of URLs



and all the features HTTP has to offer*

*: if you're using REST over HTTP

HTTP GOODIES

- Content Negotiation
- Redirection
- Authentication
- Transport Layer Security
- Caching
- Load Balancing

but remember this



TWITTERS "REST" API, DISSECTED

Let's Look At The Status Methods

STATUSES/SHOW

- GET http://api.twitter.com/1/statuses/show/id.format
- Problems:
 - Operation ("show") included in the URL
 - Status ID not a child of the "statuses" collection
- Better: GET http://twitter.com/statuses/id with Accept header

STATUSES/UPDATE

- POST http://api.twitter.com/l/statuses/update.format
- Problems:
 - Operation ("update") included in the URL
 - Uses the authenticated user implicitly
- Better: POST http://twitter.com/users/id/statuses/

STATUSES/DESTROY

- POST http://api.twitter.com/l/statuses/destroy/id.format
- Problems:
 - Operation ("destroy") included in the URL like it's 1997
 - Odd, illogical hierarchy again
 - Allows both "POST" and "DELETE" as verbs
- Better: DELETE http://twitter.com/statuses/id

STATUSES/RETWEETS

- GET http://api.twitter.com/1/statuses/retweets/id.format
- Problems:
 - Hierarchy is wrong
- Better: GET http://twitter.com/statuses/id/retweets/

STATUSES/RETWEET

- PUT http://api.twitter.com/l/statuses/retweet/id.format
- Problems:
 - · "retweets" collection exists, but is not used here
 - As usual, the action is in the URL ("make retweet" is RPC-y)
 - Allows both "PUT" and "POST" as verbs
- Better: POST http://twitter.com/statuses/id/retweets/

SUMMARY

- http://twitter.com/statuses/
 - POST to create a new tweet
- http://twitter.com/statuses/12345
 - DELETE deletes, PUT could be used for updates
- http://twitter.com/statuses/12345/retweets
 - POST creates a new retweet

HOSTS AND VERSIONING

- Q: Why not http://api.twitter.com/?
 - A: Because http://api.twitter.com/statuses/1234 and http://api.twitter.com/statuses/1234 would be different resources!
- Q: What about / I/ or /2/ for versioning?
 - A: Again, different resources. Instead, use the media type: application/vnd.com.twitter.api.v1+xml or application/vnd.com.twitter.api+xml;ver=2

FURTHER READING

- Ryan Tomayko
 How I Explained REST to my Wife
 http://tomayko.com/writings/rest-to-my-wife
- Jim Webber, Savas Parastatidis & Ian Robinson How to GET a Cup of Coffee http://www.infoq.com/articles/webber-rest-workflow
- Roy Thomas Fielding
 Architectural Styles and the Design of Network-based Software
 Architectures
 http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

UPCOMING EVENTS

REST Fest

September 17th & 18th in Greenville, SC Just \$50 for the unconference and a full-day workshop by Mike Amundsen on the 17th

International PHP Conference

October 11th - 14th in Mainz, Germany Full-day tutorial "HTTP for the REST of us", presented by Ben Ramsey and yours truly on October 14 The End

Questions?

THANKYOU!

This was a presentation by @dzuelke
Send me an e-mail!
david.zuelke@bitextender.com