DESIGNING HTTP URls AND REST INTERFACES
David Zuelke
Founder
Lead Developer
THE OLDEN DAYS

Before REST Was *En Vogue*
along came
dis is srs SEO bsns
and said
NEIN NEIN
NEIN NEIN
DAS IST
VERBOTEN
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
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
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 JSON BACK
GET /products/ HTTP/1.1
Host: acme.com
Accept: application/xml

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

<?xml version="1.0" encoding="utf-8"?>
<products xmlns="urn:com.acme.products" xmlns:xl="http://www.w3.org/1999/xlink">
  <product id="1234" xl:type="simple" xl:href="http://acme.com/products/1234">
    <name>Red Stapler</name>
    <price currency="EUR">3.14</price>
  </product>
</products>
no hypermedia formats yet in those examples!
I will show that in a few minutes
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

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

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

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

- http://www.acme.com/products/1234
THE NEXT LEVEL
Time To Throw CRUD Into The Mix
COLLECTION OPERATIONS

  - GET to **retrieve** a list of products
  - POST to **create** a new product
    - returns
      - 201 Created
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 MEDIA TYPE

GET /products/1234 HTTP/1.1
Host: acme.com
Accept: application/vnd.acmecorpshop+xml

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"?>
<product 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>
XML is really good for hypermedia formats
(hyperlinks, namespaced attributes, re-use of formats, \ldots)
JSON is more difficult
(no hyperlinks, no namespaces, no element attributes)
XML VERSUS JSON

```xml
<?xml version="1.0" encoding="utf-8"?>
<product 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>
```

```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
without hypermedia, your HTTP interface is not RESTful
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
and how it interoperates with the Web of today
it’s completely seamless
all thanks to the polymorphism of URLs
the “soft transitions” you can achieve with link relations
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
don’t use sessions, logins or cookies to maintain state
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/1/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/1/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
• 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/1/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://twitter.com/statuses/1234 would be different resources!

• Q: What about /1/ 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*
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?
THANK YOU!

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