REST: Intro, Patterns & Anti-Patterns

Stefan Tilkov | innoQ | stefan.tilkov@innoq.com

What is REST?

3
Definitions

1

REST: An Architectural Style

One of a number of “architectural styles”

... described by Roy Fielding in his dissertation

... defined via a set of *constraints* that have to be met

... architectural principles underlying HTTP, defined *a posteriori*

... with the Web as one particular instance

See: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

2

REST: The Web Used Correctly

A system or application architecture

... that uses HTTP, URI and other Web standards “correctly”

... is “on” the Web, not tunneled through it

... also called “WOA”, “ROA”, “RESTful HTTP”

3

REST: XML without SOAP

Send plain XML (w/o a SOAP Envelope) via HTTP

... violating the Web as much as WS-*

... preferably use GET to invoke methods

... or tunnel everything through POST

... commonly called “POX”

Only **option 1 is the right one**
(because Roy said so)

But we'll go with option 2
(and equate "REST" with "RESTful HTTP usage")

and avoid option 3 like the plague

REST Explained in 5 Easy Steps

1. Give Every “Thing” an ID

```
http://example.com/customers/1234

http://example.com/orders/2007/10/776654

http://example.com/products/4554

http://example.com/processes/sal-increase-234
```

2. Link Things To Each Other

```
<order self='http://example.com/orders/1234'>
  <amount>23</amount>
  <product ref='http://example.com/products/4554' />
  <customer ref='http://example.com/customers/1234' />
</order>
```

3. Use Standard Methods

GET	Retrieve information, possibly cached
PUT	Update or create with known ID
POST	Create or append sub-resource
DELETE	(Logically) remove

4. Allow for Multiple “Representations”

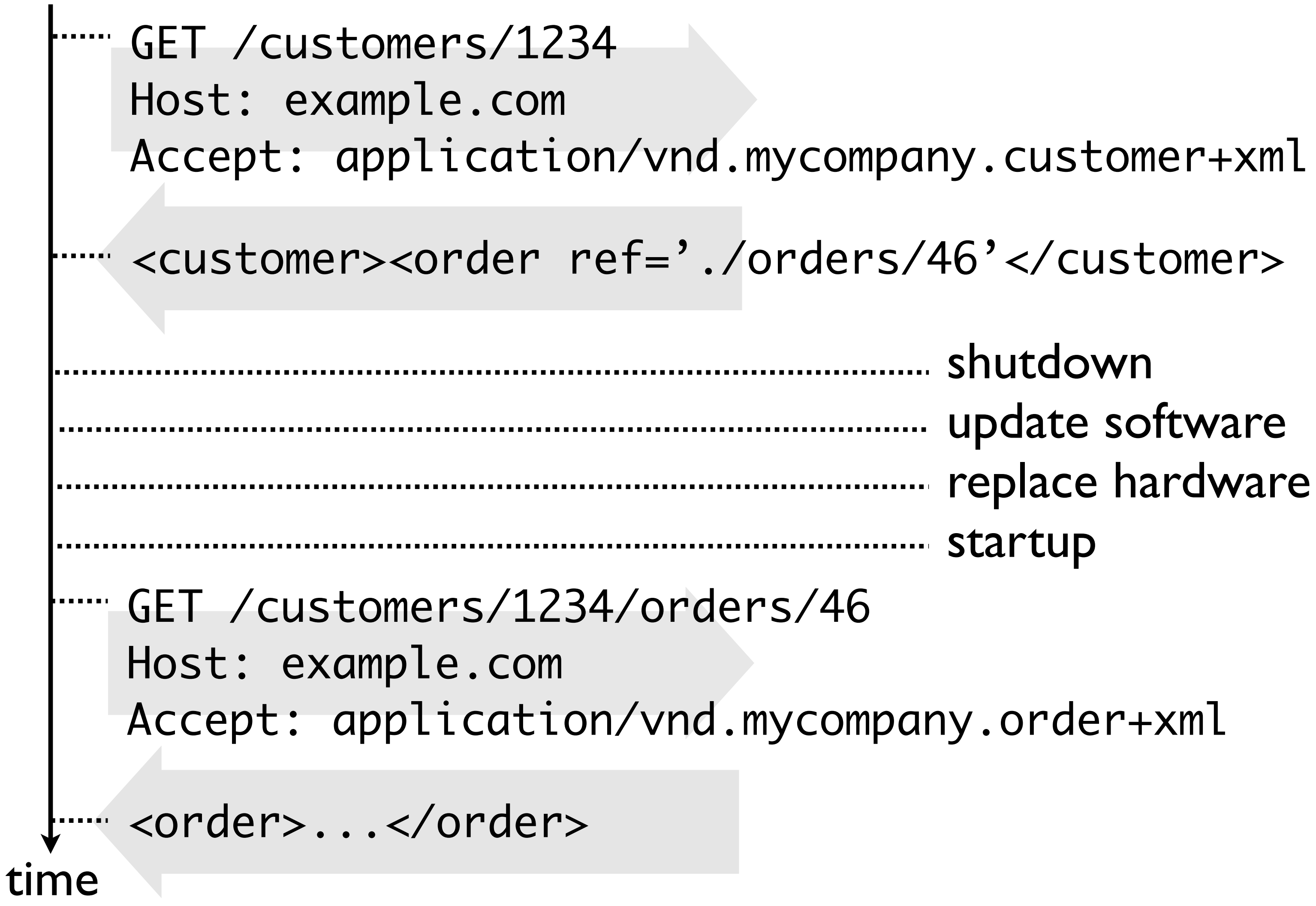
```
GET /customers/1234
Host: example.com
Accept: application/vnd.mycompany.customer+xml
```

```
<customer>...</customer>
```

```
GET /customers/1234
Host: example.com
Accept: text/x-vcard
```

```
begin:vcard
...
end:vcard
```

5. Communicate Statelessly



```
GET /customers/1234
Host: example.com
Accept: application/vnd.mycompany.customer+xml
```

```
<customer><order ref='./orders/46'</customer>
```

```
GET /customers/1234/orders/46
Host: example.com
Accept: application/vnd.mycompany.order+xml
```

```
<order>...</order>
```

What’s cool about REST?

generic

```
interface Resource {
     Resource(URI u)
     Response get()
     Response post(Request r)
     Response put(Request r)
     Response delete()
}
```

Any HTTP client
(Firefox, IE, curl, wget)

Any HTTP server

Caches

Proxies

Google, Yahoo!, MSN

```
class CustomerCollection : Resource {
     ...
     Response post(Request r) {
          id = createCustomer(r)
          return new Response(201, r)
     }
     ...
}
```

Anything that knows
your app

specific

generic

```
interface Resource {
     ...
}
```

Anything that understands HTTP

```
class AtomFeed : Resource {
     AtomFeed get()
     post(Entry e)
     ...
}
```

Any feed reader

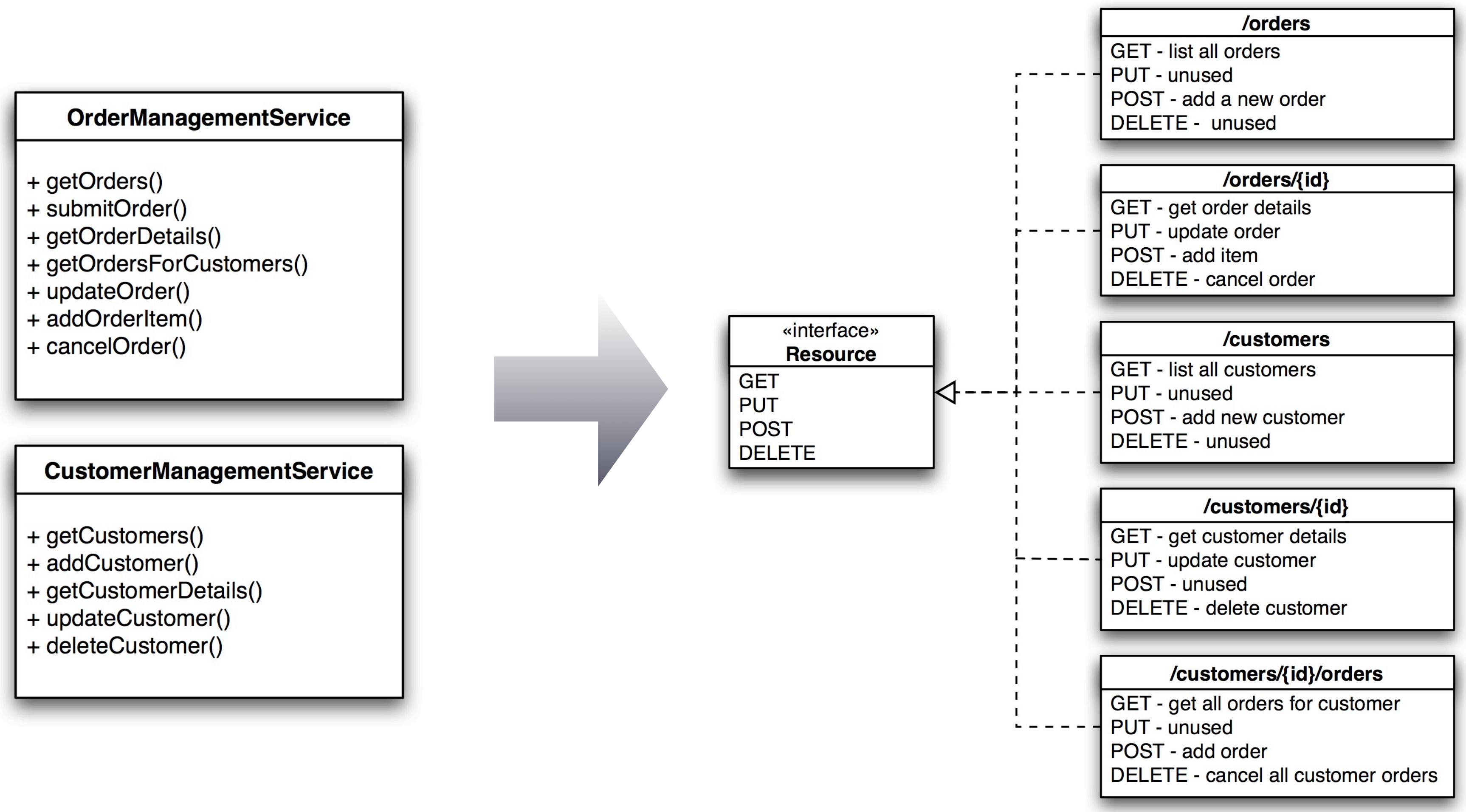
Any AtomPub client

Yahoo! Pipes

```
class CustomerCollection : AtomFeed {
     ...
}
```

Anything that knows your app

specific


OrderManagementService
+ getOrders()
+ submitOrder()
+ getOrderDetails()
+ getOrdersForCustomers()
+ updateOrder()
+ addOrderItem()
+ cancelOrder()
CustomerManagementService
+ getCustomers()
+ addCustomer()
+ getCustomerDetails()
+ updateCustomer()
+ deleteCustomer()
«interface»
Resource
GET
PUT
POST
DELETE
/orders
GET - list all orders
PUT - unused
POST - add a new order
DELETE - unused
/orders/{id}
GET - get order details
PUT - update order
POST - add item
DELETE - cancel order
/customers
GET - list all customers
PUT - unused
POST - add new customer
DELETE - unused
/customers/{id}
GET - get customer details
PUT - update customer
POST - unused
DELETE - delete customer
/customers/{id}/orders
GET - get all orders for customer
PUT - unused
POST - add order
DELETE - cancel all customer orders

Mapping Examples

getFreeTimeSlots(Person)	→GET /people/{id}/timeslots?state=free
rejectApplication(Application)	→POST /rejections↵ <application>http://...</application>↵ <reason>Unsuitable for us!</reason>
performTariffCalculation(Data)	→POST /calculations↵ Data ←Location: http://.../calculations/4711 →GET /calculations/4711 ←Result
shipOrder(ID)	→PUT /orders/0815↵ <status>shipped</status>
shipOrder(ID) [variation]	→POST /shipments↵ Data ←Location: http://.../shipments/4711

REST Anti-Patterns

Tunneling Through GET

```
http://example.com/some-api?method=insert&name=Smith
http://example.com/some-api?method=deleteCustomer&id=13
```

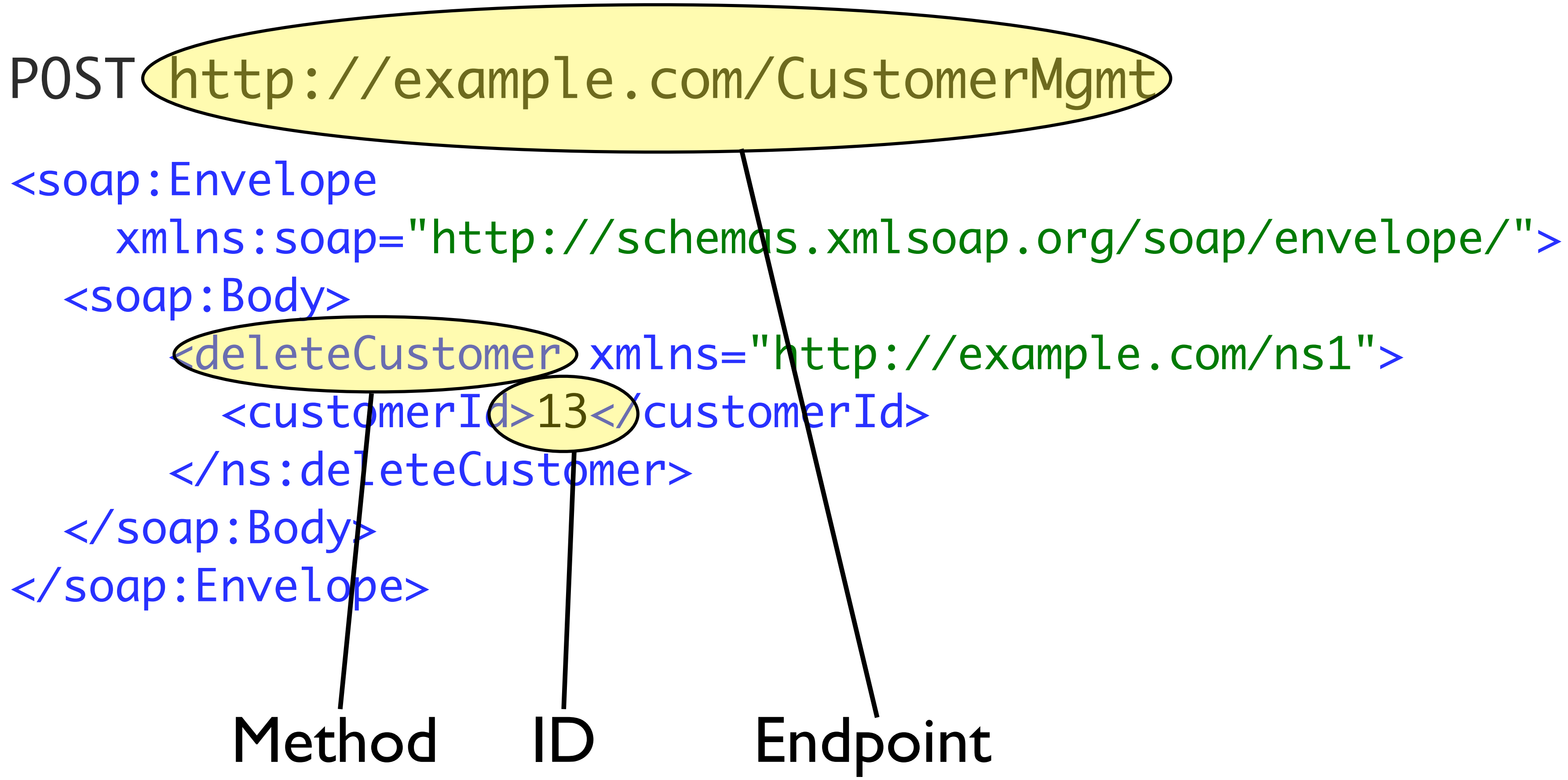
```
http://example.com/some-api?method=findCustomer&id=13
http://example.com/customers/13
```

Accidentally RESTful

http://www.markbaker.ca/blog/2005/04/14/accidentally-restful/

Tunneling Through POST

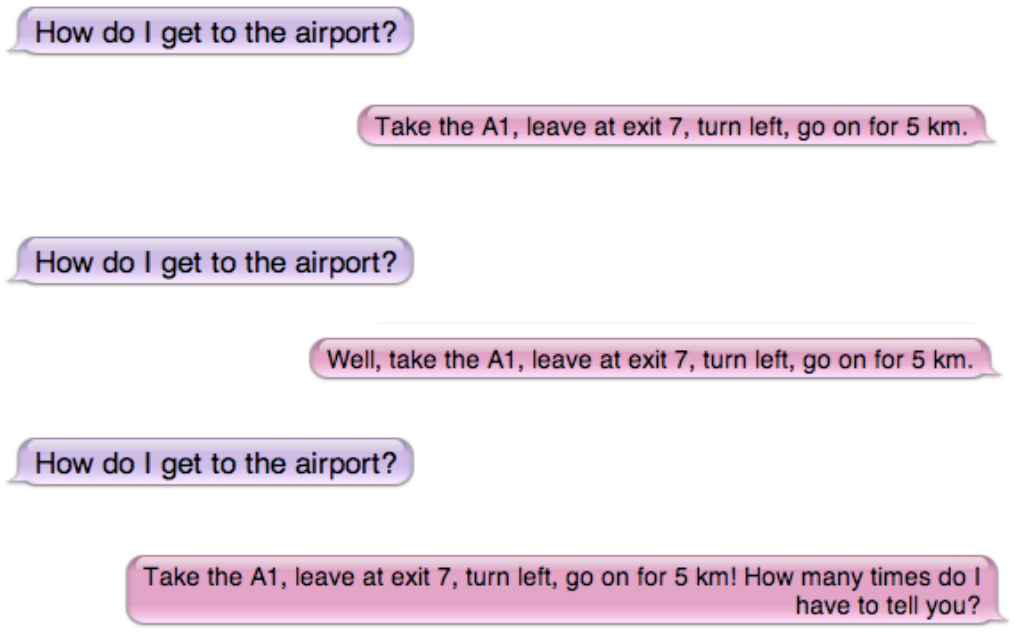
(a.k.a. The SOAP Way)



```
POST http://example.com/CustomerMgmt
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
      <deleteCustomer xmlns="http://example.com/ns1">
        <customerId>13</customerId>
      </ns:deleteCustomer>
  </soap:Body>
</soap:Envelope>
```

“Endpoint”?

DO NOT
ENTER
DEAD
END



How do I get to the airport?

Take the A1, leave at exit 7, turn left, go on for 5 km.

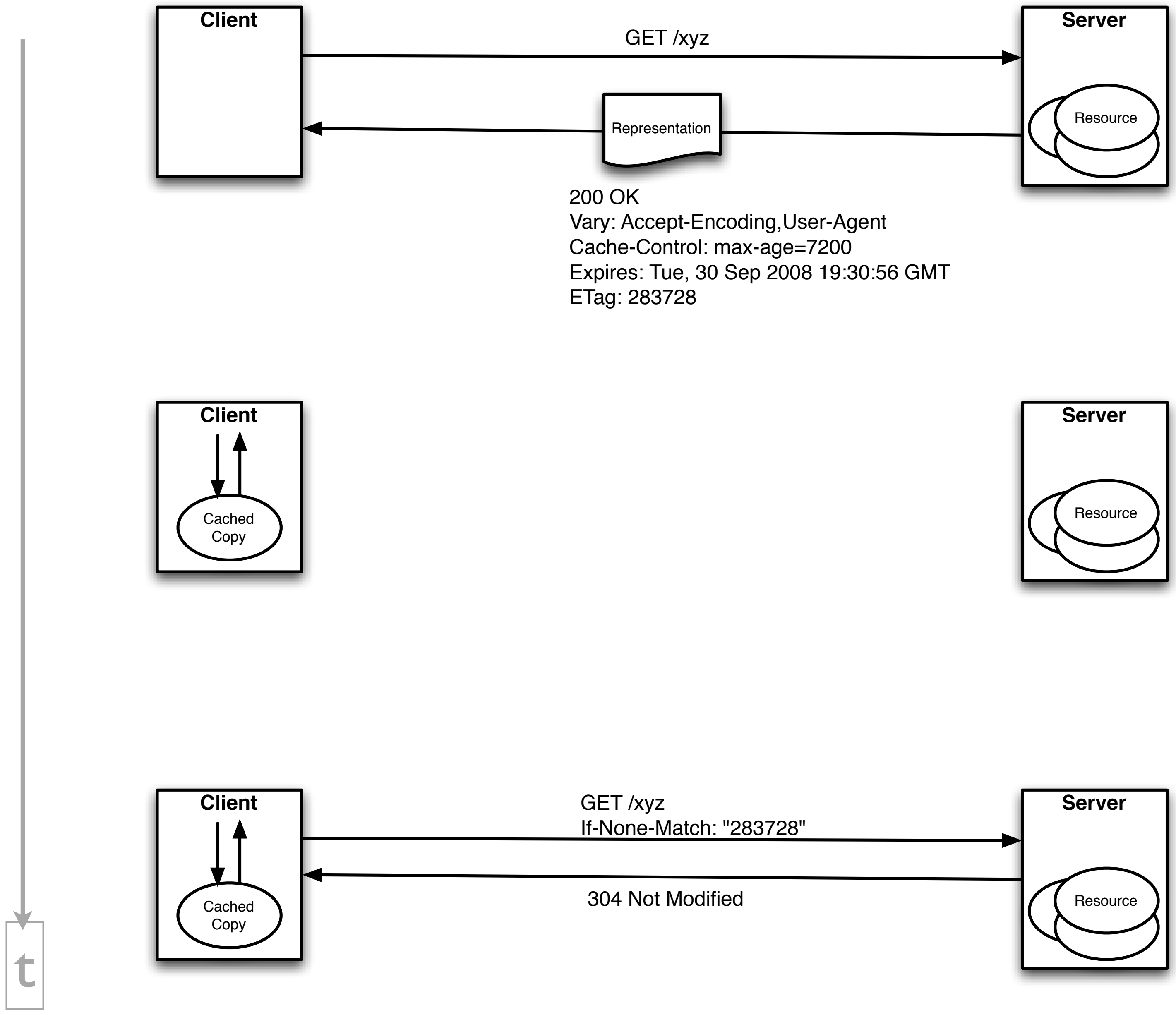
How do I get to the airport?

Well, take the A1, leave at exit 7, turn left, go on for 5 km.

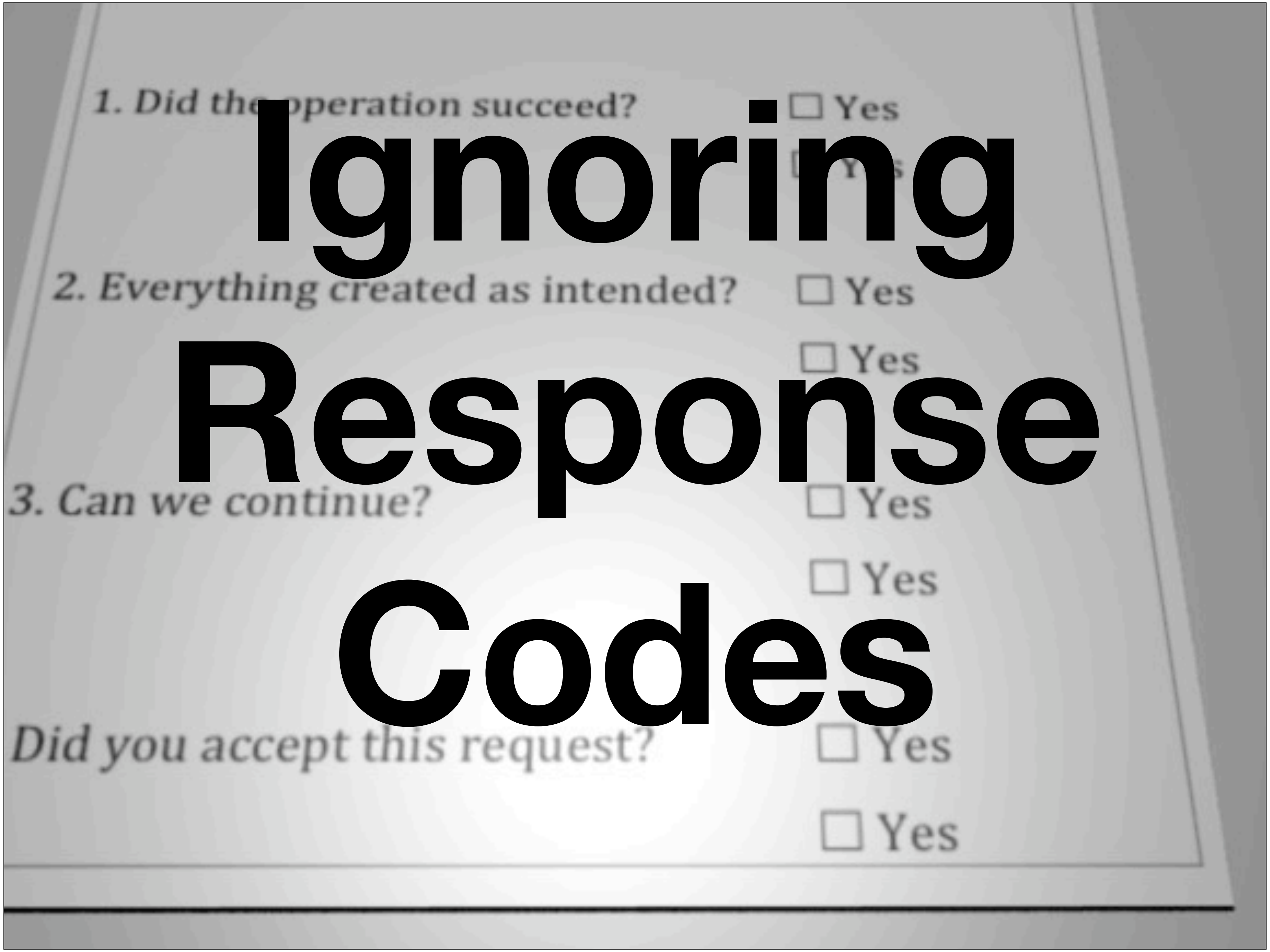
How do I get to the airport?

Take the A1, leave at exit 7, turn left, go on for 5 km! How many times do I have to tell you?

Ignoring Caching


Client
Server
GET /xyz
Representation
Resource
200 OK
Vary: Accept-Encoding,User-Agent
Cache-Control: max-age=7200
Expires: Tue, 30 Sep 2008 19:30:56 GMT
ETag: 283728
Client
Cached Copy
Server
Resource
Client
Cached Copy
Server
GET /xyz
If-None-Match: "283728"
304 Not Modified
Resource
t

Ignoring Response Codes


1. Did the operation succeed? ☐ Yes ☐ Yes
2. Everything created as intended? ☐ Yes ☐ Yes
3. Can we continue? ☐ Yes ☐ Yes
Did you accept this request? ☐ Yes ☐ Yes

100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported

Misusing Cookies

RESTful Cookie Recipe

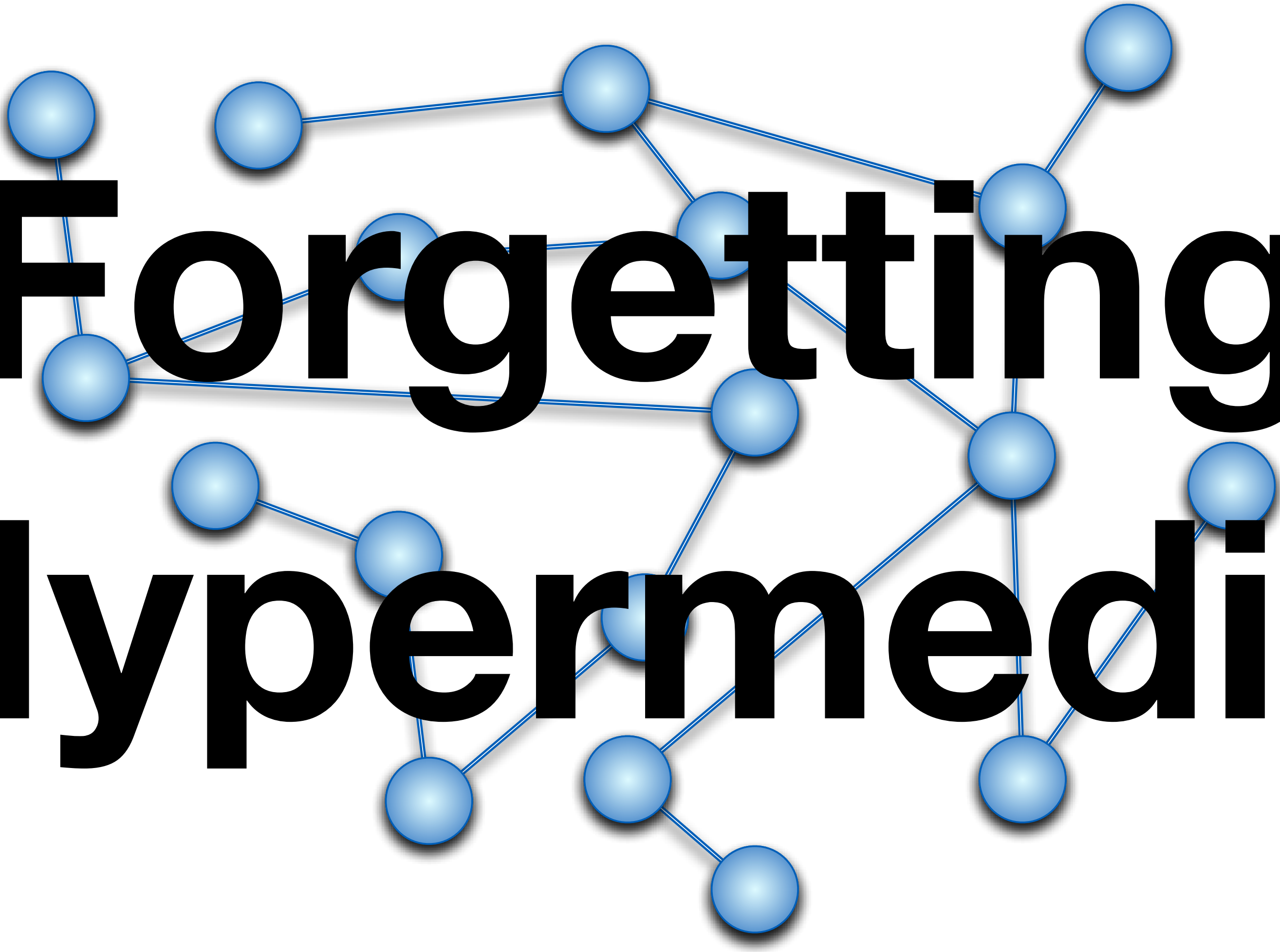
Ingredients:

- 1 server-side secret
- user name/password validation on server (LDAP, DB, ...)

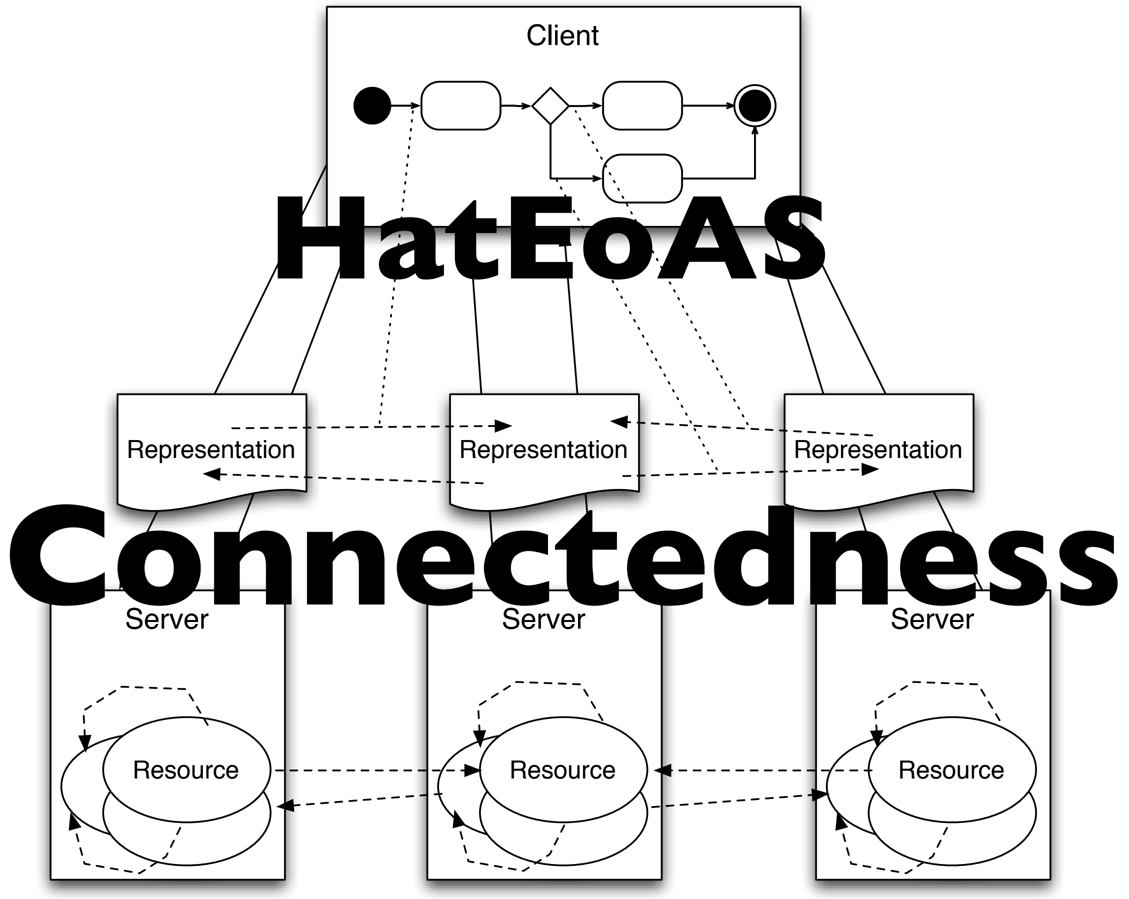
Approach:

- ask user for name and password if no cookie passed
- authenticate user
- create auth token as username + expiry date
- hash(auth token + server secret)
- return cookie as hash + auth_token
- server validates with algorithm on in-memory data

Forgetting Hypermedia



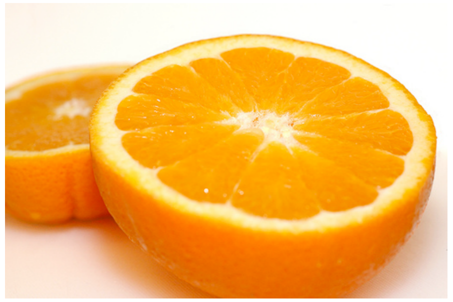
Hypermedia Levels



application/xml
application/vnd.mytype

Thing

Thing



Thing

Ignoring MIME Types

Breaking Self-descriptiveness

REST Patterns

Collection Resource

Context	Related resources are accessed in groups
Solution	Turn collection into resource, Use links to point to contained resources, Include summary information for contained resources

```
GET http://example.com/customers/
```

```
<?xml version="1.0" encoding="utf-8"?>
<customers xmlns="http://example.com/ns/crm">
  <base-uri>http://example.com</base-uri>
  <customer>
    <name>Company A</title>
    <link type="text/html" href="/customers/4711"/>
    ...
  </customer>
```

Read-only View

Context	Need for specialized views on one or more collections or resources
Solution	Create additional read-only list resources, Link to underlying resources

```
http://example.com/customers/
http://example.com/customers/?region=3
http://example.com/customer-addresses/
http://example.com/changes/customers/?limit=10
http://example.com/orders/2008/09/30/1200-1259
```

NOTICE

Stop Worrying
About URI Design

```
http://example.com/orders/2008/09/30/1200-1259
```

```
http://example.com/AD273AFCCB78898ADEEFCC22
```

Resource Creation

Context	Resources are created concurrently and need unique URIs
Solution	POST contents to the collection that will contain the resource Receive 201 response code, (possibly changed) representation and Location header
Alternative	Create UUID on client, PUT content to {server URI}/{UUID}

Notification Polling

Context	Clients need to know about updates to resources
Solution	Define View if needed, Expose as RSS or Atom Feed, Ensure correct cache control headers

Conflict Handling

Context	Protect against concurrent modification (lost update problem)
Solution	Provide ETag and Last-Modified Headers, Include preconditions, Send correct 409/412 response codes for unsafe methods

Named Link

Context	Decouple client processing resource connections
Solution	Define link roles, Build processing for roles, Include links with role as attribute

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">dive into mark</title>
  <updated>2005-07-31T12:29:29Z</updated>
  <id>tag:example.org,2003:3</id>
  <link rel="alternate" type="text/html" hreflang="en" href="http://example.org/"/>
  <link rel="self" type="application/atom+xml" href="http://example.org/feed.atom"/>
  <entry>
    <title>Atom draft-07 snapshot</title>
    <link rel="alternate" type="text/html" href="http://example.org/2005/04/02/atom"/>
    <link rel="enclosure" type="audio/mpeg" length="1337" href="..."/>
    ...
```

Saved Search

Context	Complex query input with mostly stable result or "unsafe" query
Solution	POST search criteria, Receive result URI in Location header, GET result (w/ cache control headers)

Conneg Extensions

Context	Support linking to specific representation formats, increase testability
Solution	Provide generic resource with content negotiation, Provide distinct resources for one or more representations mapped by extension

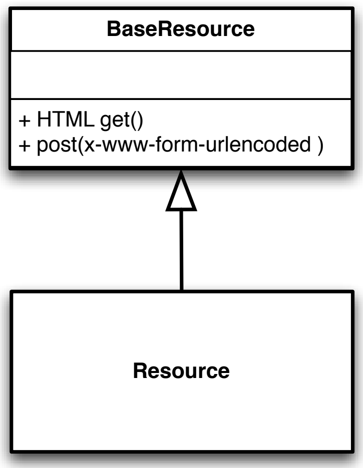
```
GET http://example.com/customer/4711
GET http://example.com/customer/4711.xml
GET http://example.com/customer/4711.html
```

PUT/DELETE Tunneling

Context	Firewalls or other tooling does not support or blocks PUT and DELETE
Solution	Use POST to tunnel PUT and DELETE, Encode "true" verb in HTTP header or hidden HTML form field

Canonical Representation

Context	Ensure lowest common denominator of processing
Solution	Provide default HTML presentation for reading Enable www-form-data for simple processing Provide HTML for queries



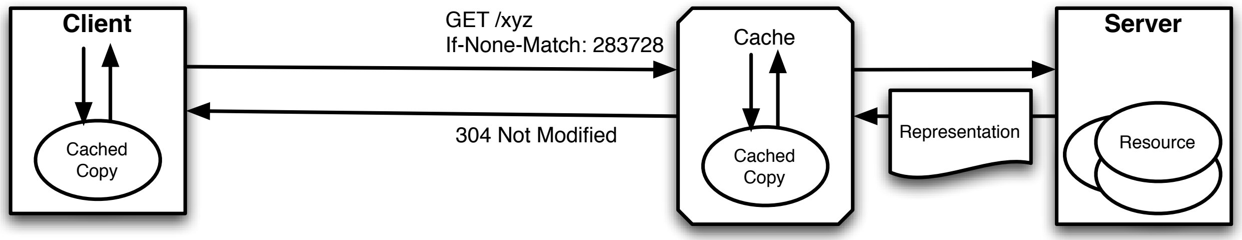
Deep ETags

Context	Reduce computation load on server
Solution	Include ETag for resource presentations returned from server, Implement fast ETag checking w/o full representation computation, Return appropriate 304 response code

Externalized Server Cache

Context Simplify server caching implementation

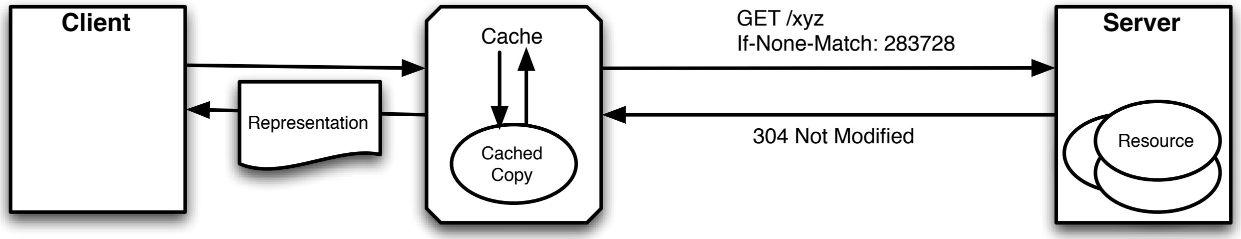
Solution Get rid of server cache implementation,
Produce cache-control headers/ETags/
Last-Modified,
Implement Deep ETags,
Add caching intermediary



Externalized Client Cache

Context Simplify client caching implementation

Solution Get rid of client cache implementation,
Add client caching intermediary

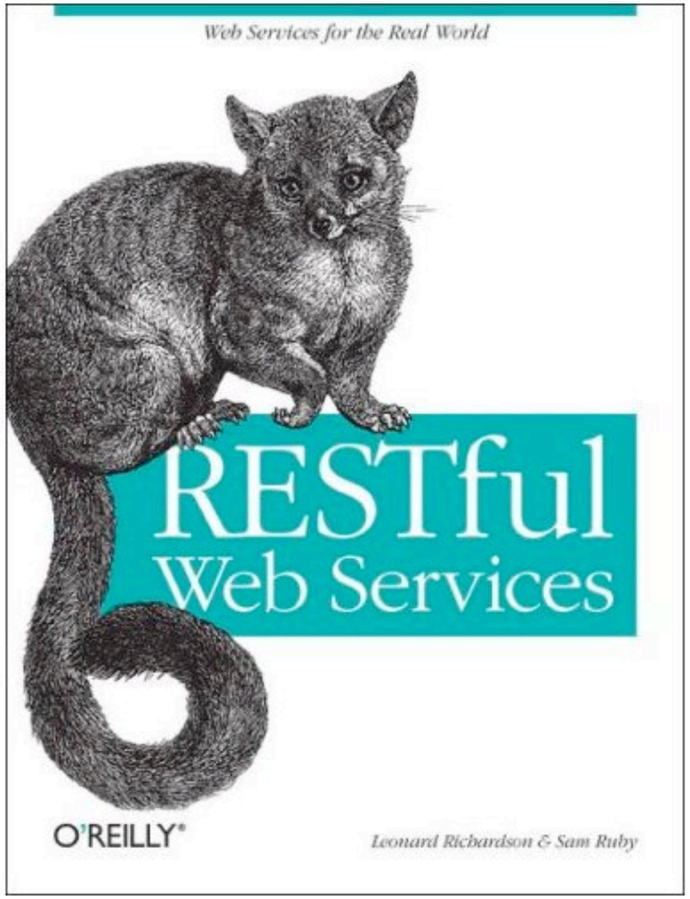


Transaction

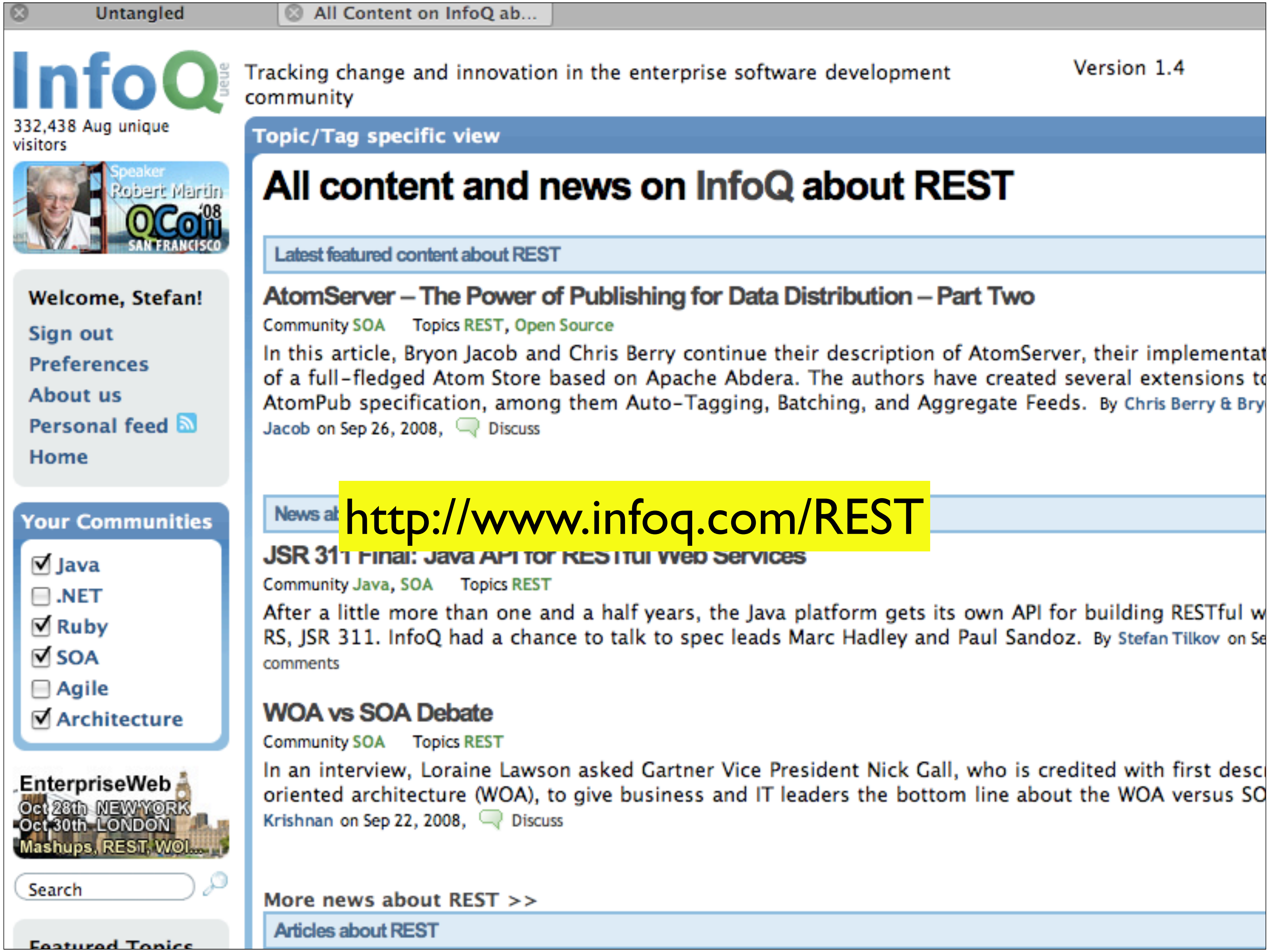
Context	Several resources have to be modified in a single request
Solution	Turn transaction into resource, Modify transaction resource itself, possibly in multiple steps Finally PUT to transaction to commit all changes

If You Want to Know More

http://www.innoq.com/resources/REST


Web Services for the Real World
RESTful Web Services
O'REILLY
Leonard Richardson & Sam Ruby

http://www.oreilly.com/catalog/9780596529260/



Tracking change and innovation in the enterprise software development community

Version 1.4

332,438 Aug unique visitors

Topic/Tag specific view

All content and news on InfoQ about REST

Welcome, Stefan!

Sign out

Preferences

About us

Personal feed

Home

Latest featured content about REST

AtomServer – The Power of Publishing for Data Distribution – Part Two

Community SOA Topics REST, Open Source

In this article, Bryon Jacob and Chris Berry continue their description of AtomServer, their implementat of a full-fledged Atom Store based on Apache Abdera. The authors have created several extensions to AtomPub specification, among them Auto-Tagging, Batching, and Aggregate Feeds. By Chris Berry & Bry Jacob on Sep 26, 2008, Discuss

Your Communities

☑ Java
☐ .NET
☑ Ruby
☑ SOA
☐ Agile
☑ Architecture

News ab

http://www.infoq.com/REST

JSR 311 Final: Java API for RESTful Web Services

Community Java, SOA Topics REST

After a little more than one and a half years, the Java platform gets its own API for building RESTful w RS, JSR 311. InfoQ had a chance to talk to spec leads Marc Hadley and Paul Sandoz. By Stefan Tilkov on Se comments

WOA vs SOA Debate

Community SOA Topics REST

In an interview, Loraine Lawson asked Gartner Vice President Nick Gall, who is credited with first desc oriented architecture (WOA), to give business and IT leaders the bottom line about the WOA versus SO Krishnan on Sep 22, 2008, Discuss

EnterpriseWeb
Oct 28th NEW YORK
Oct 30th LONDON
Mashups, REST, WOI...

Search

More news about REST >>

Articles about REST

Featured Topics

Thank you!
Any questions?

http://www.innoq.com
http://railsconsulting.de

Stefan Tilkov
http://www.innoq.com/blog/st/

innoQ

Architectural Consulting

SOA	WS-*	REST
MDA	MDSD	MDE
J(2)EE	RoR	.NET

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 21 02 77 162-100
info@innoq.com · www.innoq.com

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

REST ≠ CRUD

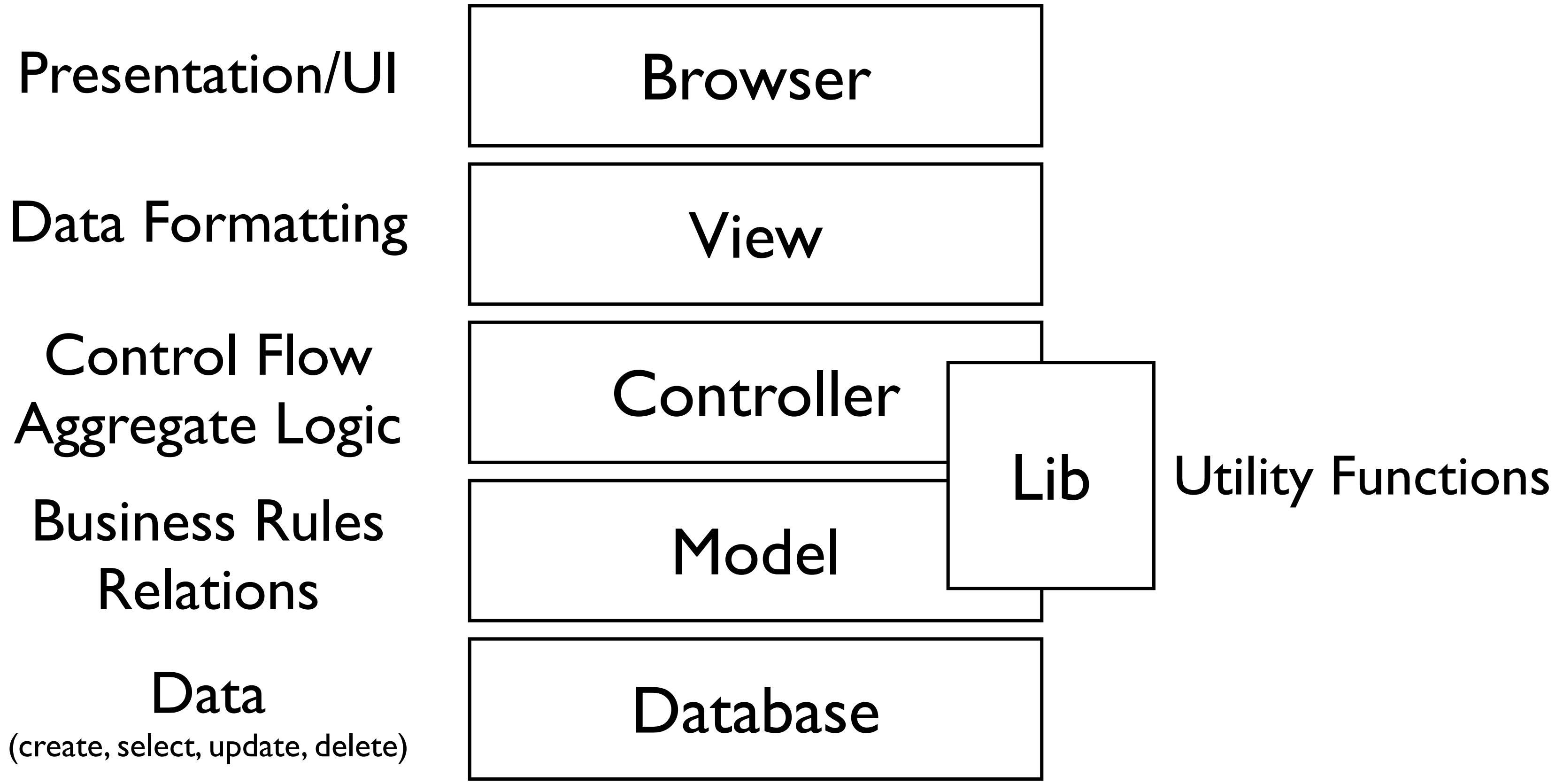
?

Resource ≠ Entity

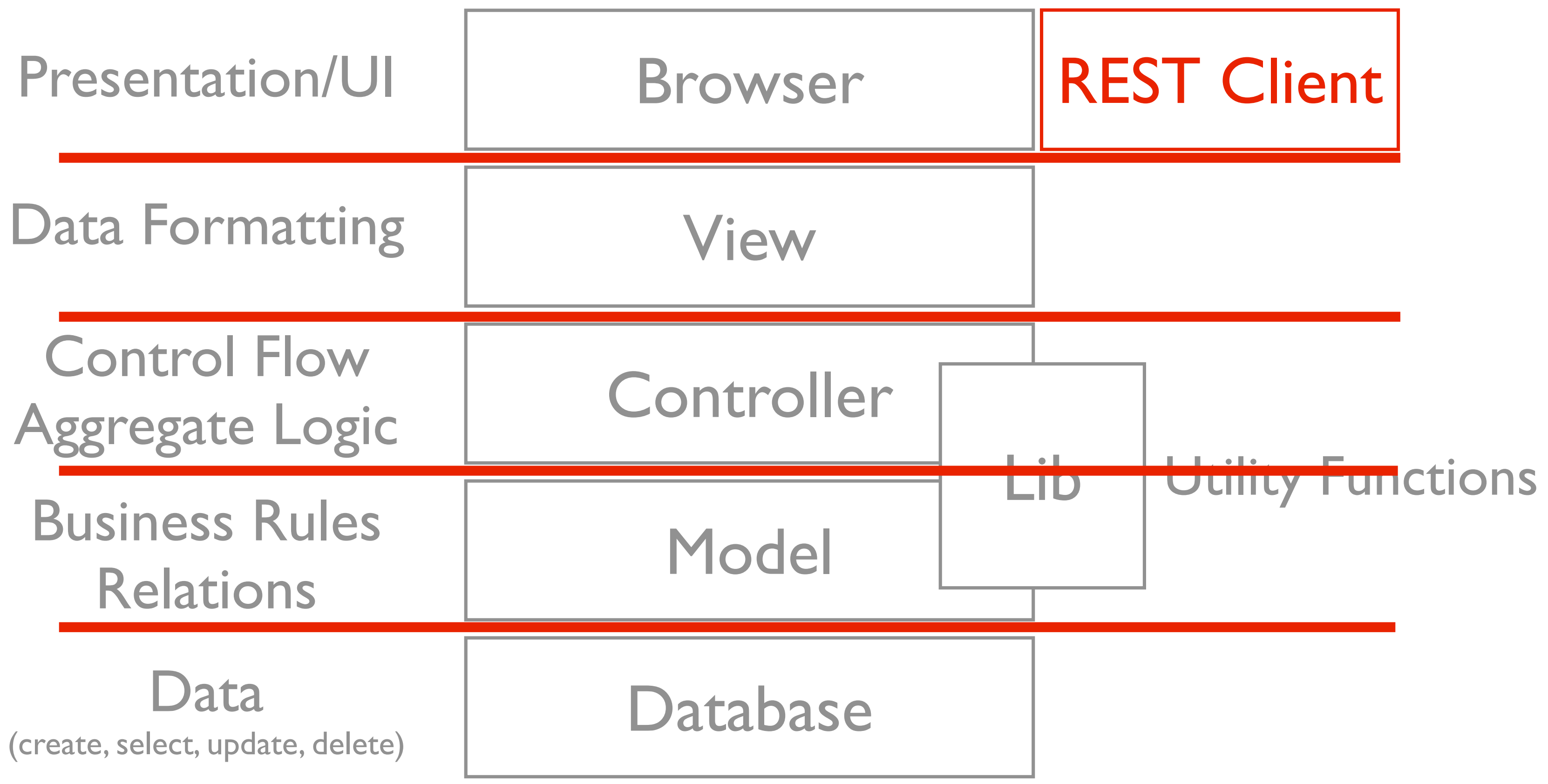
Resource ≈ Model

Resource ≈ Controller

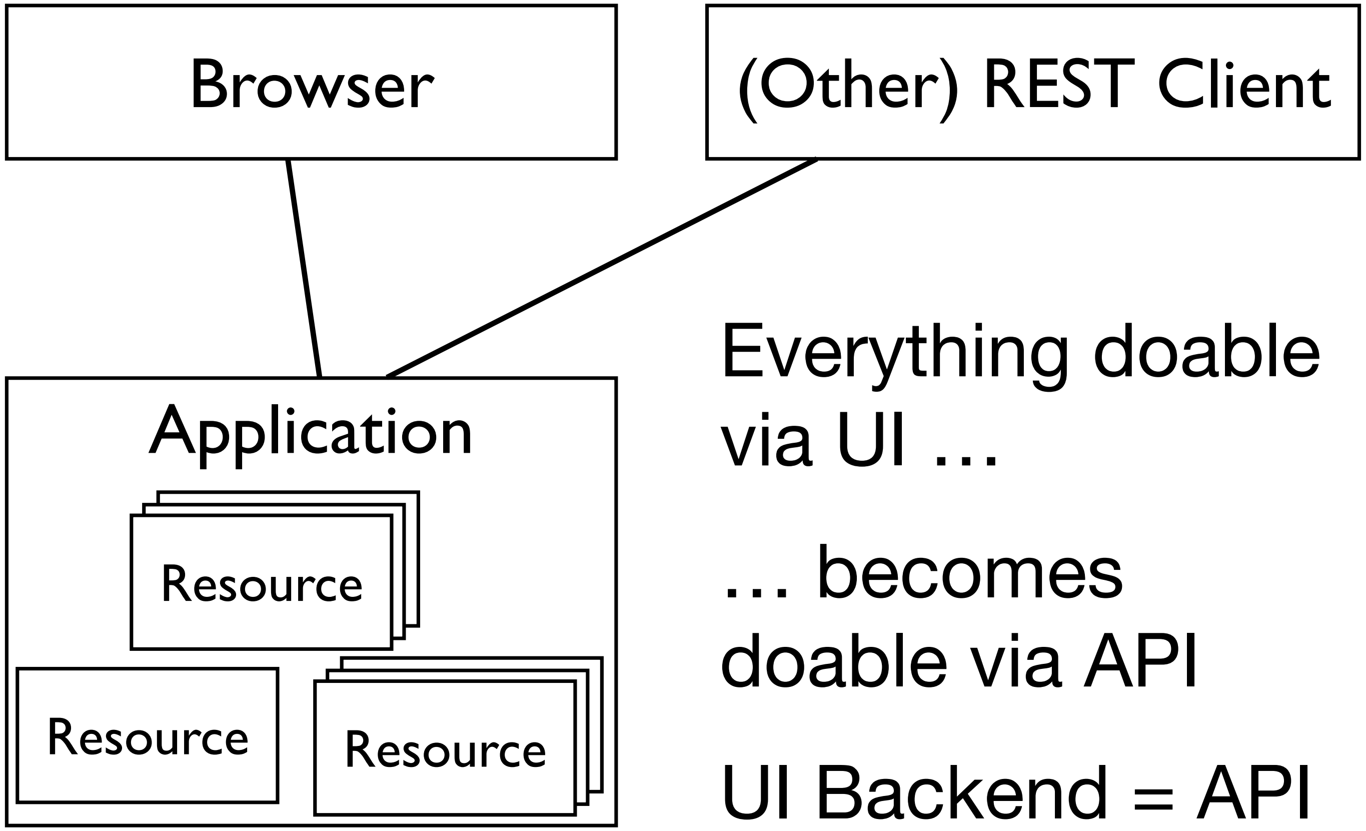
Application Layers



Application Layers & Resources



Single Resource Model



Everything doable via UI …

… becomes doable via API

UI Backend = API

RESTful APIs

RESTful APIs don’t expose low-level details

Same layer – different abstraction

Value through uniformity and hypermedia

Mapping necessity: “Implement” HTTP base interface