FrOSCon
Free and Open Source Software Conference

How to deal with **REST** in practice?

FrOSCon, Bonn - Germany
25-26 August 2012

Joshua Thijssen / Netherlands

Freelance consultant, developer and trainer @ NoxLogic / Techademy

Development in PHP, Python, Perl, C, Java....

Blog: http://adayinthelifeof.nl

Email: jthijssen@noxlogic.nl
Twitter: @jaytaph

REST

Representational State Transfer

REST

Representational State Transfer

It's not hard:

REST

Representational State Transfer

It's not hard: like poker, or chess

@Todo:

- Quick REST recap
- Common “myths”
- Common “mistakes”
- How to do stuff

Quick REST recap

Are you doing REST?

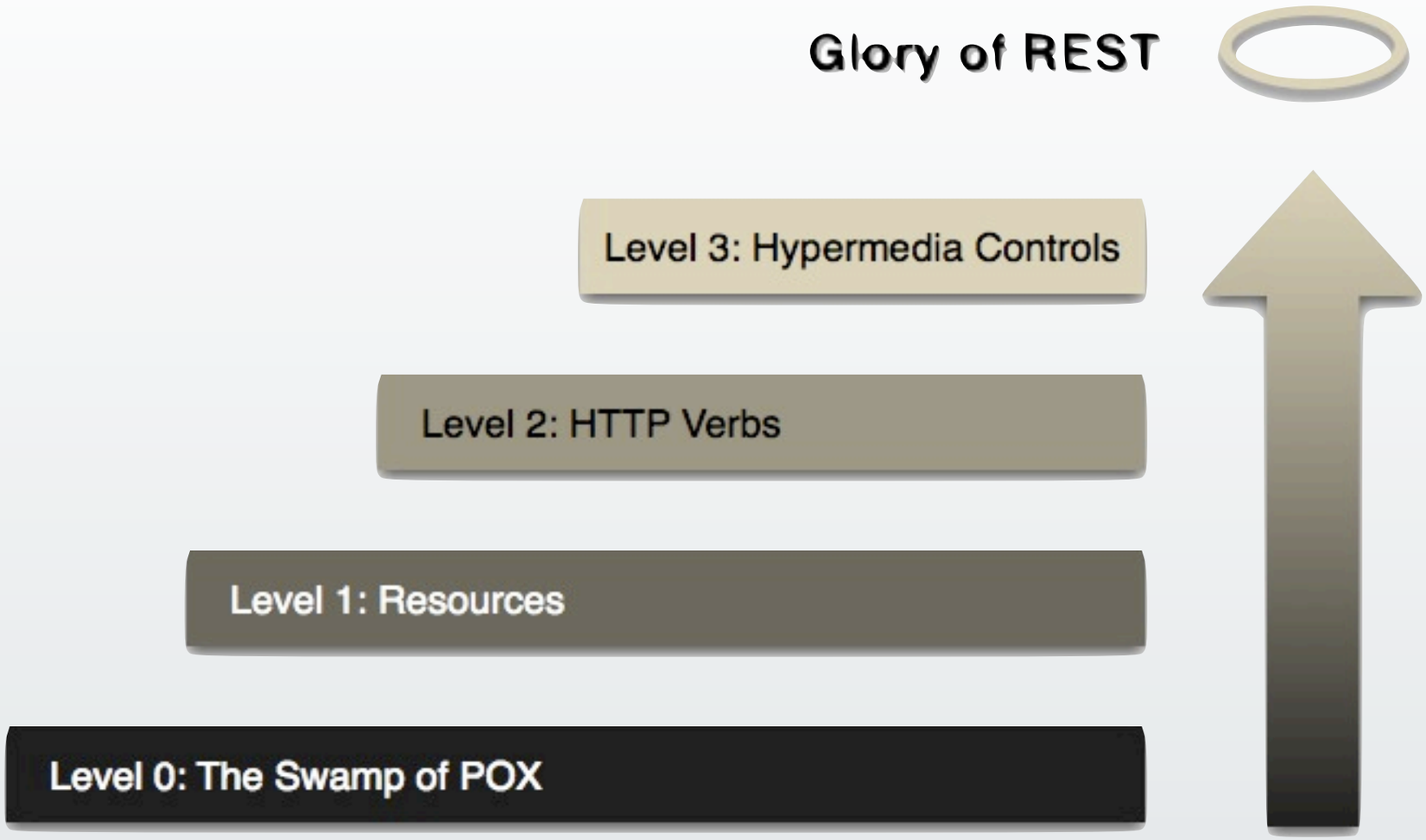
Restful constraints

- Client / Server
- Stateless
- Cacheable
- Layered system
- Code on demand (optional)
- Uniform interface

Uniform interface

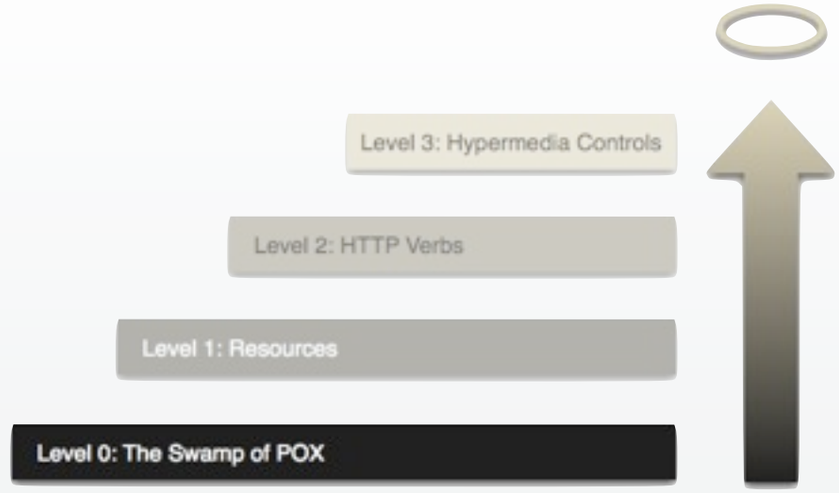
- Identification through representations of resources
- Manipulation through representations
- Self-descriptive messages
- Hypermedia as the engine of application state (HATEOAS)

Are you mature enough?
(according to Richardson)



http://martinfowler.com/articles/richardsonMaturityModel.html

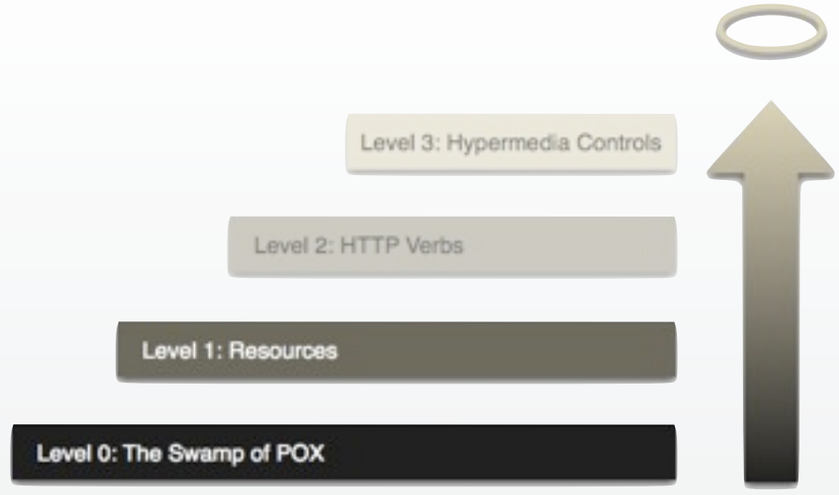
Level 0: Plain Old XML



- HTTP is tunnel protocol
- POST to single URL (or worse: GET)
- SOAP / XML-RPC

http://martinfowler.com/articles/richardsonMaturityModel.html

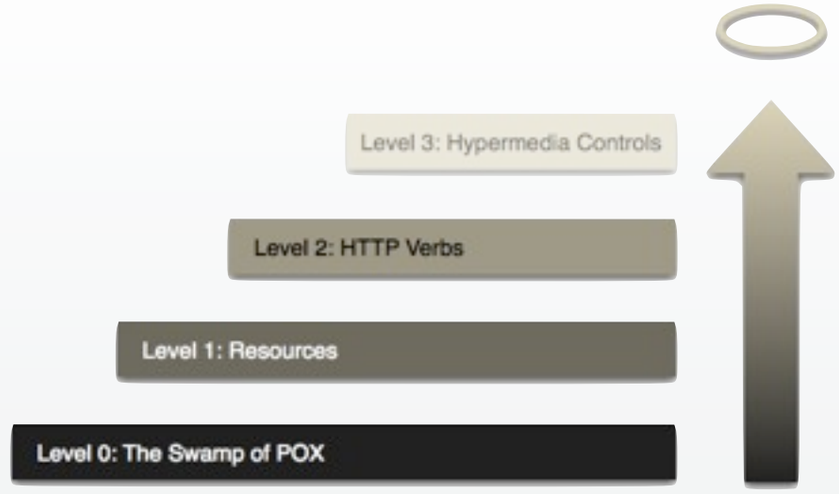
Level 1: Resources



- Entities are resources
- /user/jthijssen instead of /users
- /user/jthijssen/talks

http://martinfowler.com/articles/richardsonMaturityModel.html

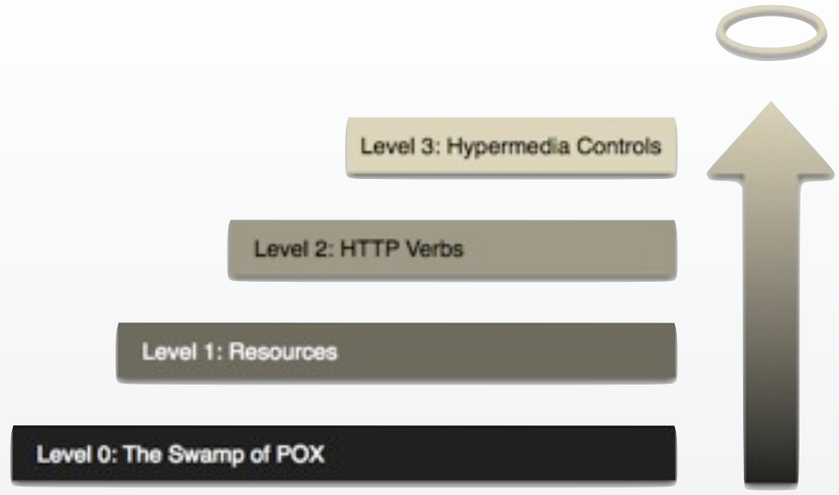
Level 2: HTTP Verbs



- POST or PUT for creations
- GET for retrievals
- POST or PUT for updates
- DELETE for deletions
- PATCH for partial updates

http://martinfowler.com/articles/richardsonMaturityModel.html

Level 3: Hypermedia controls



- HATEOAS
- Hypermedia as the engine of application state
- Using links to detect your next states
- One bookmark link to rule them all

http://martinfowler.com/articles/richardsonMaturityModel.html

Let’s go to DisneyLand!

Let's go to DisneyLand!

Wait, wut?

Are you (still) doing REST?

Common “myths”

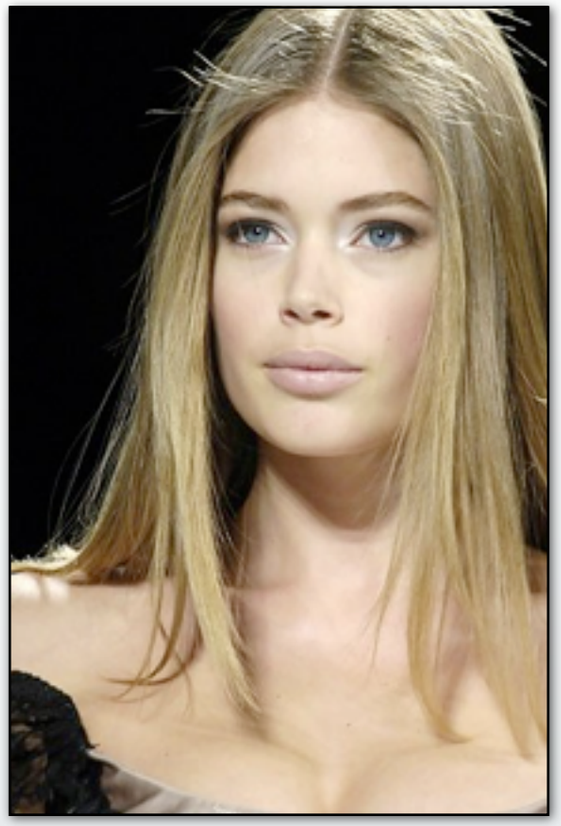
REST == HTTP

REST == CRUD

URL's are important

REST scales

COOKIES are EVIL



I've never have eaten a cookie in my life...

Gimme moar cookies! Om nom nom!

Application state

vs

Resource state

Resource state

- User has got multiple addresses
- Entity X/Y/Z is set to 42

Resource state

- User has got multiple addresses
- Entity X/Y/Z is set to 42

Resource state never changes through GET
(or other safe) methods!

Application state

- Which "stage" is the user in the checkout process?
- Which page is the user currently browsing
- Is the user currently logged in?
- Depends....

"Per client" state should be saved by the client,
not on the server.

"Per client" state should be saved by the client,
not on the server.

It's called Representational STATE TRANSFER

Common “mistakes”

PUT or POST?

PUT or POST?

PUT when the resource URI is known

PUT /user/jthijssen/talk/123

PUT or POST?

PUT when the resource URI is known

PUT /user/jthijssen/talk/123

POST when it's not (server decides)

POST /user/jthijssen/talks

PUT = idempotent, POST is not!

PUT = idempotent, POST is not!

PUT /user/jthijssen/talk/123
PUT /user/jthijssen/talk/123
PUT /user/jthijssen/talk/123

PUT = idempotent, POST is not!

PUT /user/jthijssen/talk/123
PUT /user/jthijssen/talk/123
PUT /user/jthijssen/talk/123

POST /user/jthijssen/talks
POST /user/jthijssen/talks
POST /user/jthijssen/talks

PUT = idempotent, POST is not!

PUT /user/jthijssen/talk/123
PUT /user/jthijssen/talk/123
PUT /user/jthijssen/talk/123

POST /user/jthijssen/talks
POST /user/jthijssen/talks
POST /user/jthijssen/talks

POST is the worst option for caching / scalability,
but use it if you don't know what to do.

If you need to “construct” an URI, you are doing it wrong

If you need to “construct” an URI, you are doing it wrong

```
url = MAIN_URL + “/” + user_id + “/talks”;
rc = HTTP.post(url, data);
```

If you need to “construct” an URI, you are doing it wrong

```
url = MAIN_URL + “/” + user_id + “/talks”;
rc = HTTP.post(url, data);
```

```
url = data.link_rel(“talks”, user_id);
rc = HTTP.post(url, data);
```

If you need to “construct” an URI, you are doing it wrong

```
url = MAIN_URL + “/” + user_id + “/talks”;
rc = HTTP.post(url, data);
```

```
url = data.link_rel(“talks”, user_id);
rc = HTTP.post(url, data);
```

Uri can change: server, uri, protocol, port etc..

Composite resources

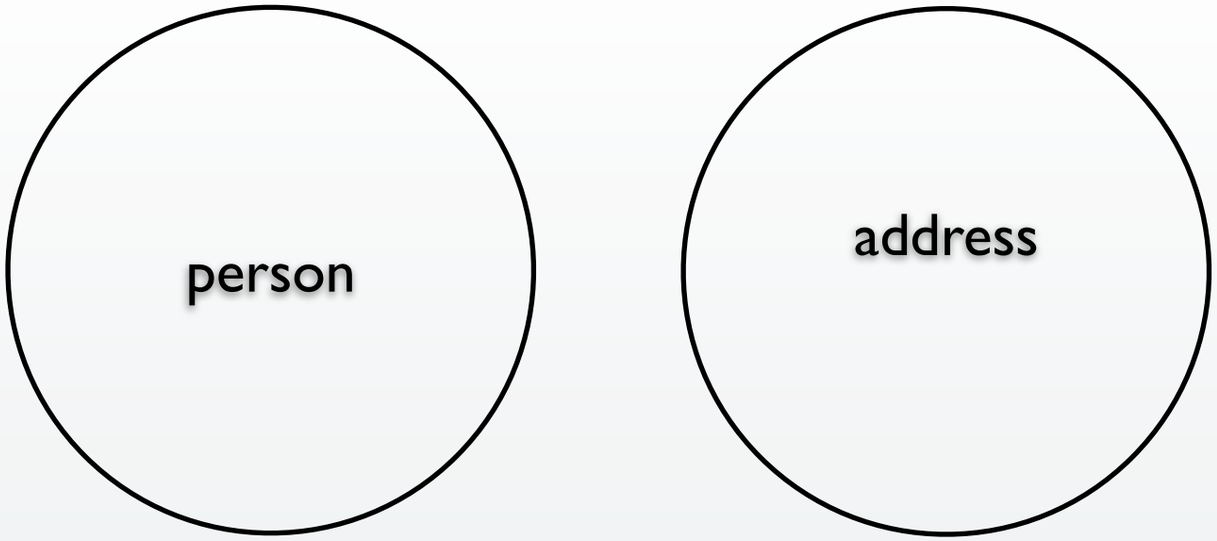
Watch out with visibility

```
<xml version="1.0" encoding="utf-8">
<persons>
  <person>
    <name>Joshua Thijssen</name>
    <phone type="cell">0612345678</phone>
    <link rel="address" href="/users/jthijssen/address">
  </person>
  ...
<persons>
```

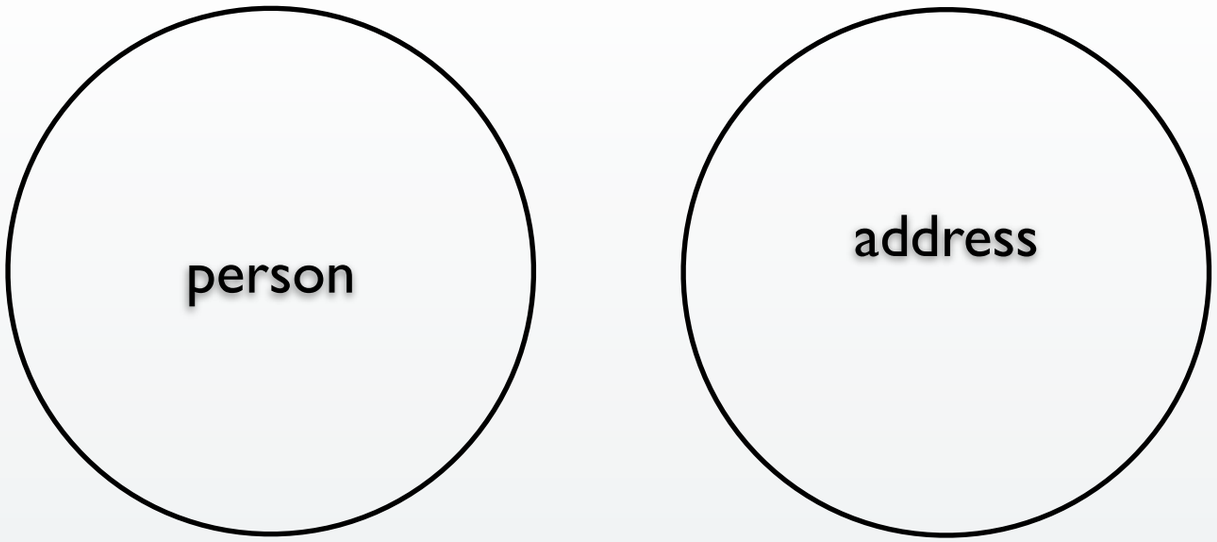
Watch out with visibility

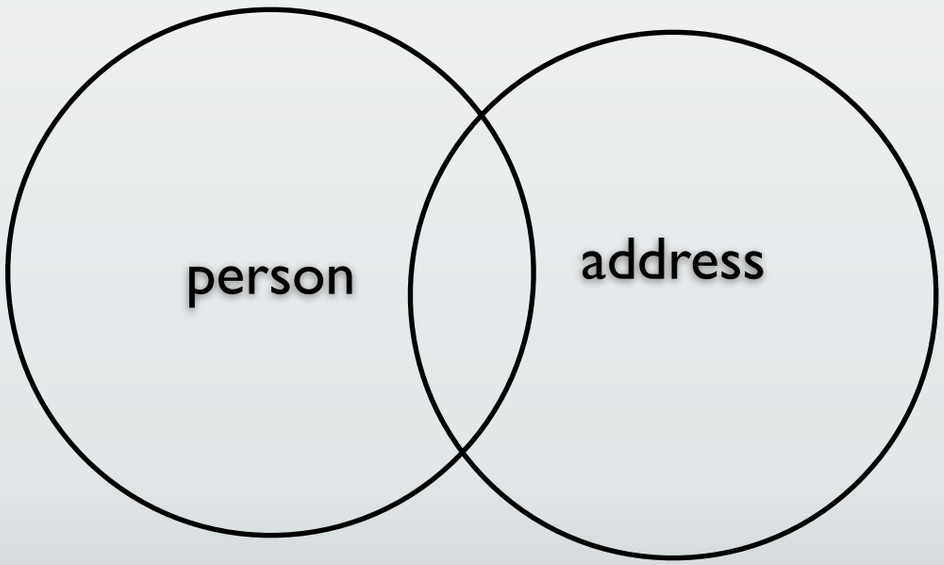
```
<xml version="1.0" encoding="utf-8">
<persons>
  <person>
    <name>Joshua Thijssen</name>
    <phone type="cell">0612345678</phone>
    <address>
      <street>Mainstreet 1234</street>
      <postalcode>1234AB</postalcode>
      <city>Amsterdam</city>
      <country>Netherlands</country>
    </address>
    <link rel="address" href="/users/jthijssen/address">
  </person>
  ...
</persons>
```

Watch out with visibility

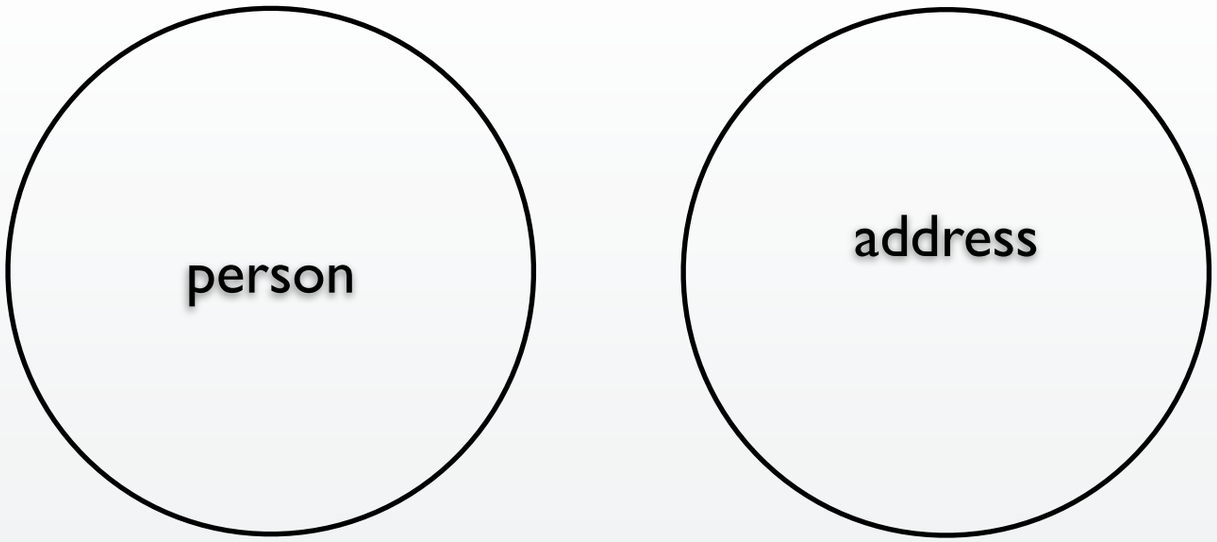


Watch out with visibility

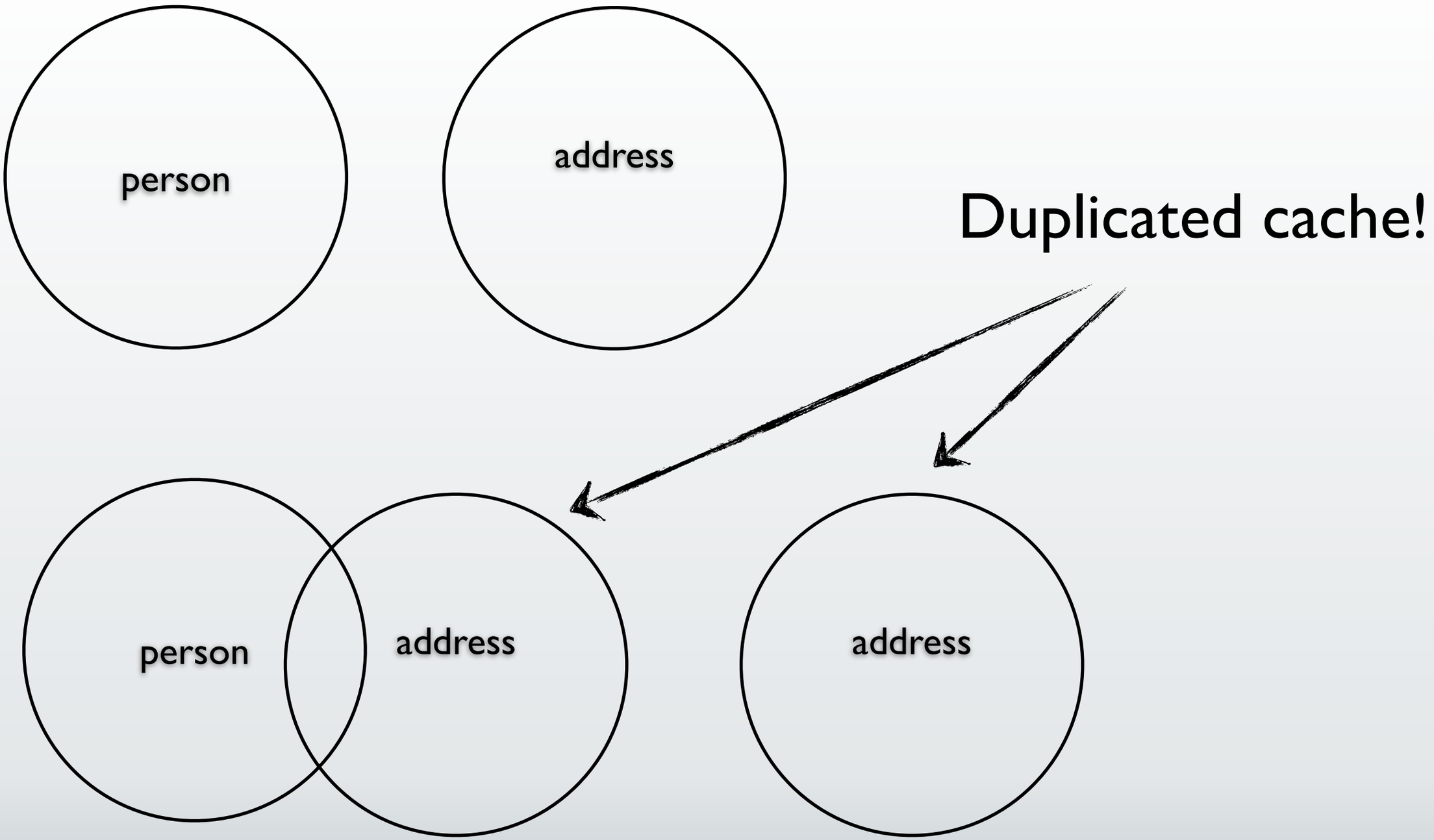




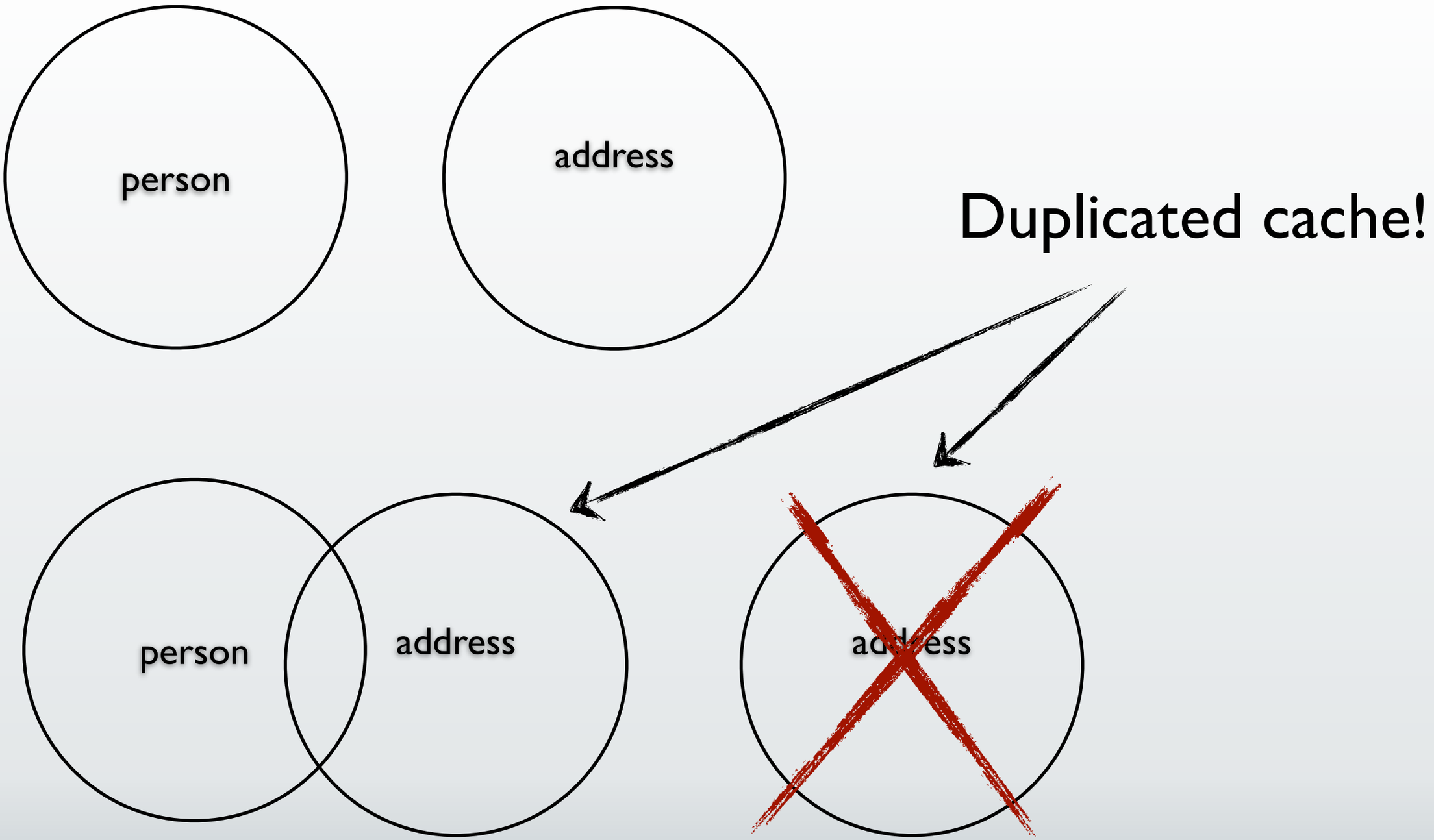
Watch out with visibility



Watch out with visibility



Watch out with visibility



- Degrades resource visibility since they contain overlapping data
- Use caching proxies! (maybe ESI???)
- Each client needs a different composite.

It sounded like a good idea at the time...

- Use HTTP (verbs) wisely

- Use HTTP (verbs) wisely
- Etags / If-(not-)modified

NOX LOGIC

- Use HTTP (verbs) wisely
- Etags / If-(not-)modified
- HTTP codes

NOX LOGIC

- HTTP 1xx : Do I got info for you!

- HTTP 1xx : Do I got info for you!
- HTTP 2xx : We're cool..

- HTTP 1xx : Do I got info for you!
- HTTP 2xx : We're cool..
- HTTP 3xx : Take a look there..

- HTTP 1xx : Do I got info for you!
- HTTP 2xx : We're cool..
- HTTP 3xx : Take a look there..
- HTTP 4xx : your bad!

- HTTP 1xx : Do I got info for you!
- HTTP 2xx : We're cool..
- HTTP 3xx : Take a look there..
- HTTP 4xx : your bad!
- HTTP 5xx : my bad!

- Sometimes hard:
  - 405 Method not allowed
  - 501 Not implemented
  - Who is to blame? (tip: blame client!)

Don’t return OK when it’s not:

```
HTTP/1.1 200 Ok

<xml>
  <errorcode>my-superduper-code-nobody-understands<errorcode>
  <error>This action is forbidden</error>
</xml>
```

- Use a hypermedia format (xhtml / atom)

- Use a hypermedia format (xhtml / atom)
- JSON is NOT a hypermedia format

- Use a hypermedia format (xhtml / atom)
- JSON is NOT a hypermedia format
- JSON-LD http://json-ld.org/

- Use a hypermedia format (xhtml / atom)
- JSON is NOT a hypermedia format
- JSON-LD http://json-ld.org/
- HAL http://stateless.co/hal_specification.html

How to do stuff

How do I login into my API?

You don't

Client

Cookie
PHPSESSID: 1234ABCD

Server

PHPSESSID: 1234ABCD
LoggedIn: true
User: 52
IsAdmin: false

Client

Server

Cookie
Loggedin: true
User: 52
IsAdmin:false

Client

Server

Cookie
Loggedin: true
User: 52
IsAdmin: true

Client

Server

Cookie:
0xEncryptedData

- Authenticate / Authorize per request.
- Caching is possible, just don't rely on it.
- If you need state, make sure it's resource state, not session/application state.

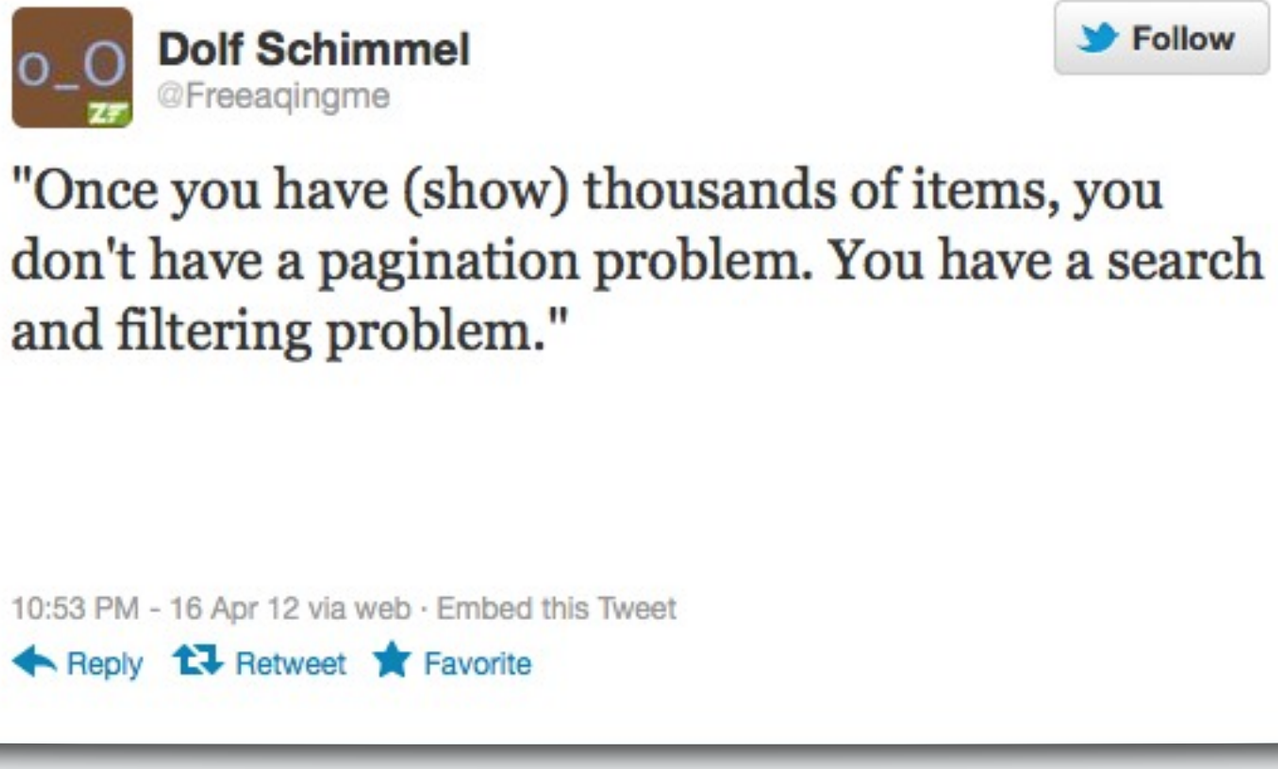
Querying for data

```
/entities/search?
what=restaurants&type=italian&postalcode=1234AB&radius=5000&sor
t=rating&order=desc
```

```
/entities/search?
what=restaurants&type=italian&postalcode=1234AB&radius=5000&sor
t=rating&order=desc
```

```
/restaurants/italian/top10?postalcode=1234AB&radius=5000
```

Pagination


o_O
Dolf Schimmel
@Freeaqingme
Follow
"Once you have (show) thousands of items, you don't have a pagination problem. You have a search and filtering problem."
10:53 PM - 16 Apr 12 via web · Embed this Tweet
Reply
Retweet
Favorite

```
/restaurants?page=5
```

```
/restaurants?page=635
```

```
/restaurants/top1000?page=5
```

```
<xml version="1.0" encoding="utf-8">
<restaurants>
  <restaurant>...</restaurant>
  <restaurant>...</restaurant>

  <link rel="first" href="/restaurants/top10?page=1">
  <link rel="self" href="/restaurants/top10?page=5">
  <link rel="previous" href="/restaurants/top10?page=4">
  <link rel="next" href="/restaurants/top10?page=6">
  <link rel="last" href="/restaurants/top10?page=25">
</restaurants>
```

Applying hierarchy

Not everything implies an hierarchy:

```
/directions/brussels/amsterdam
```

Parameters, order does matter:

```
/directions/brussels,amsterdam
```

```
/directions/amsterdam,brussels
```

```
/directions?from=brussels&to=amsterdam
```

```
/directions?from=amsterdam&to=brussels
```

Parameters, order does not matter:

```
/distance/amsterdam;brussels
```

```
/distance/brussels;amsterdam
```

One cache system:

```
HTTP/1.1 303 See other
Location: /distance/amsterdam;brussels
```

Asynchronous updates

Synchronous updates

```
POST /blogs HTTP/1.1
Content-type: application/vnd.myblog.article+xml ; version=1.0

<?xml version="1.0" encoding="UTF-8" ?>
<article>
  <title>My blogpost</title>
  <author>John Doe</author>
  <content>This is the content for my blog article</content>
</article>
```

Synchronous updates

```
POST /blogs HTTP/1.1
Content-type: application/vnd.myblog.article+xml ; version=1.0

<?xml version="1.0" encoding="UTF-8" ?>
<article>
  <title>My blogpost</title>
  <author>John Doe</author>
  <content>This is the content for my blog article</content>
</article>
```

```
HTTP/1.1 201 Created
Location: /blog/20010101-myblogpost
```

Asynchronous updates

```
POST /blogs HTTP/1.1
Content-type: application/vnd.myblog.article+xml ; version=1.0

<?xml version="1.0" encoding="UTF-8" ?>
<article>
  <title>My blogpost</title>
  <author>John Doe</author>
  <content>This is the content for my blog article</content>
</article>
```

Asynchronous updates

```
POST /blogs HTTP/1.1
Content-type: application/vnd.myblog.article+xml ; version=1.0

<?xml version="1.0" encoding="UTF-8" ?>
<article>
  <title>My blogpost</title>
  <author>John Doe</author>
  <content>This is the content for my blog article</content>
</article>
```

```
HTTP/1.1 202 Accepted
Location: /queue/621252
```

Asynchronous updates - waiting in queue

```
GET /queue/621252 HTTP/1.1
```

Asynchronous updates - waiting in queue

```
GET /queue/621252 HTTP/1.1
```

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" ?>
<queue>
  <status>Pending</status>
  <eta>10 minutes</eta>
  <link rel="cancel" href="/queue/621252"/>
</queue>
```

Asynchronous updates - in progress

```
GET /queue/621252 HTTP/1.1
```

Asynchronous updates - in progress

```
GET /queue/621252 HTTP/1.1
```

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" ?>
<queue>
  <status>In progress</status>
  <eta>3 minutes, 25 seconds</eta>
</queue>
```

Asynchronous updates - done

```
GET /queue/621252 HTTP/1.1
```

Asynchronous updates - done

```
GET /queue/621252 HTTP/1.1
```

```
 HTTP/1.1 303 See Other
Location: /blog/20010101-myblogarticle
```

Transactions

Don’t make transactions
through multiple resources.

Breaks the state constraint

(plus the rest of the the internet)

```
POST /account/1234?amount=-100
TransIDX: 55A50611FE
```

```
POST /account/1234?amount=-100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /account/1234?amount=-100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /account/4567?amount=+100
TransIDX: 55A50611FE
```

```
POST /account/1234?amount=-100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /account/4567?amount=+100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /account/1234?amount=-100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /account/4567?amount=+100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /commit
TransIDX: 55A50611FE
```

```
POST /account/1234?amount=-100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /account/4567?amount=+100
TransIDX: 55A50611FE
```

```
HTTP/1.1 202 Accepted
```

```
POST /commit
TransIDX: 55A50611FE
```

Nope! This is state!

Your API is not a RDBMS.

The internet does not need more ACID.

A transaction can been as a resource

Another attempt:

Create a new transaction

```
POST /transactions
201 Created
Location: /transactions/55A50611FE
```

```
POST /transactions/55A50611FE
POST /transactions/55A50611FE/commit
POST /transactions/55A50611FE/rollback
POST /transactions/55A50611FE/snapshot
POST /transactions/55A50611FE/rollback/1
```

But now we are back to XML-RPC or worse..

So, define services:

```
POST /booking
<xml>
  <amount currency=”USD”>10.000.000</amount>
  <from_account>12.34.56.789, my bank</from_account>
  <to_account>X5252P25, Cayman Islands</to_account>
</xml>
```

This stuff is hard!

- If you do REST, don't break the constraints.
- Be realistic about the constraints
- XML-RPC, HTTP-services (even SOAP) are valid for their uses.

NOX LOGIC

- Take into account that you probably are not building the next Twitter API.
- If you follow the REST constraints, at least your API can scale.

Questions?

Thank you

Find me on twitter: @jaytaph

Find me on email: jthijssen@noxlogic.nl

Find me for blogs: www.adayinthelifeof.nl

Find me for development or training: www.noxlogic.nl