Pragmatic REST
Building RESTful Web APIs

Subbu Allamaraju
Yahoo! Inc  ||  http://subbu.org
About

- Tech Yahoo!
  - Developing standards, patterns and practices for HTTP web APIs

- Past
  - At BEA

- A “Convert”
Disclaimer

All the opinions I express here are mine and do not necessarily represent those of my present or past employers.
Agenda

REST – The Architecture

REST vs WS – Pros and Cons

Building HTTP APIs RESTfully
Smell Test

“We offer SOAP and REST end points”

“Our technology supports REST-style URLs”

“REST is not fit for machine-machine interactions”
QUIZ
Are these RESTful?

http://example.org/persons?start=10&count=100
http://example.org/person/123.xml
http://example.org/person/123/address/4
http://example.org/movie/Gone_With_the_Wind
http://example.org/movie/Terminator/reviews
Is this RESTful?

GET /services/rest/?
    method=flickr.photos.setPerms&photo_id=2691065403&
    is_public=1&is_friend=0&is_family=0&perm_comment=1&
    perm_addmeta=1 HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8

<rsp stat="ok">
    <photoid>2691065403</photoid>
</rsp>
Is this RESTful?

POST /services/rest/?
   method=flickr.photos.getRecent&api_key=... HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8

<rsp stat="ok">
   <photos page="1" pages="10" perpage="100">
      <photo id="2947640330" owner="15150729@N07"
         secret="..." server="3060" farm="4" title="..."
         ispublic="1" isfriend="0" isfamily="0" />
   ...
   </photos>
</rsp>
Is this RESTful?

GET /photos?filterBy=recent HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8

<rsp stat="ok">
  <photos page="1" pages="10" perpage="100"
         total="1000">
    <photo id="2947640330" owner="15150729@N07"
           secret="…" server="3060" farm="4" title="…"
           ispublic="1" isfriend="0" isfamily="0" />

    …
  </photos>
</rsp>
Is this RESTful?

GET /photos?filterBy=recent&api_key=blah
Host: example.org

200 OK
Content-Type: application/xml;charset=UTF-8

<rsp stat="fail">
  <err code="96" msg="Invalid signature"/>
</rsp>
REST
REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; hypermedia as the engine of application state.

- Roy Fielding, 2000
In other words

1. Resources and URIs
2. Representations
3. Uniform interface
4. HATEOAS
But no

- Schemas
- Description languages
- Code generation
- Registries
REST/HTTP EXPLAINED
Tenet 1: Resources

- A thing with an identity
  - A blog entry, a person, a friend, an address

- Resources are named via URIs
  - http://example.org/blog/what-is-rest
  - http://example.org/person/subbu
  - http://example.org/person/subbu/friends
  - http://example.org/person/subbu/address/home

- URIs are stable
  - URIs are names
  - Names don’t change often
Identifying Resources

- We will come back to this later
Assigning URIs

Think primary keys

<table>
<thead>
<tr>
<th>A person</th>
<th><a href="http://example.org/person/123">http://example.org/person/123</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>An address of a person</td>
<td><a href="http://example.org/person/123/address/4">http://example.org/person/123/address/4</a></td>
</tr>
<tr>
<td>A movie</td>
<td><a href="http://example.org/movie/Gone_With_the_Wind">http://example.org/movie/Gone_With_the_Wind</a></td>
</tr>
<tr>
<td>Reviews of a movie</td>
<td><a href="http://example.org/movie/Terminator/reviews">http://example.org/movie/Terminator/reviews</a></td>
</tr>
<tr>
<td>A group of people</td>
<td><a href="http://example.org/persons?start=10&amp;count=100">http://example.org/persons?start=10&amp;count=100</a></td>
</tr>
</tbody>
</table>
URIS are Generally Opaque

Clean URIs don’t generally matter

http://example.org/76asfg2g

Okay as long as clients don’t have to parse and understand these

URI Opacity – We will come back to this later
URI Design Considerations

Opaque to client apps –
Transparent to client developers

Hierarchical path segments
A good way to organize resources

http://example.org/movie/Terminator/reviews
URI Considerations

Reserve query params for optional inputs

(A convention, not a rule)

http://example.org/movies

findBy {latest,director,studio}
fromYear Year
toYear Year
location Postal code, or city
...

http://example.org/movies/findBy/
Tenet 2: Representations

Things sent over requests and received over responses

An XML representation of a book

A PNG representation of a map

Data submitted through an HTML form to create a user

A JSON representation of the user created

An HTML view of the same user
Media Types

Like the “type” of an object
Like the “schema” of an XML element

An XML representation of a book: 
application/vnd.example.book+xml

A PNG representation of a map: image/png

HTML form submission: application/x-www-form-urlencoded

A JSON representation of the user created: 
application/vnd.example.user+json

An HTML view of the same user: text/html
Representations are Negotiable

Accept: application/atom+xml; q=1.0, text/html; q=0.1
Accept-Charset: UTF-8

Content-Type: application/atom+xml; charset=UTF-8

Accept-Language: fr; q=1.0, en=0.8

Content-Language: en

Accept-Encoding: gzip, deflate

Content-Encoding: deflate
Varying a Response

Tell intermediaries and clients how you chose a representation

Accept: application/atom+xml;q=1.0,text/htm;q=0.1
Accept-Charset: UTF-8
Accept-Language: fr;q=1.0,en=0.8
Accept-Encoding: gzip, deflate

Content-Type: application/atom+xml;charset=UTF-8
Content-Encoding: deflate
Vary: Accept, Accept-Encoding

Don’t overload URIs
Tenet 3 – Uniform Interface

Network transparency

- A uniform interface to operate on resources
- Uniform interface == A fixed set of operations
- Uniform interface == Generic interface
  - Irrespective of what resource is being operated on
HTTP is a Uniform Interface

A protocol between clients and resources

- **GET**
  - Get a representation
  - Safe and idempotent

- **POST**
  - Like a factory operation
  - Unsafe and non-idempotent

- **PUT**
  - Create or update a resource
  - Unsafe but idempotent

- **DELETE**
  - Delete a resource
  - Unsafe but idempotent
A Generic Application Protocol

- CRUD on resources
- Content negotiation
- Caching
- Optimistic concurrency
Is CRUD Crude?

- Yes, may be – depends on how you identify resources
Domain Nouns

- Nouns in your application domain
- NYT Movie Reviews API
  - Movies, Reviews, Critics, Critic’s picks, Reviews by reviewer
- Netflix API
  - Catalog, Users, Rentals, Rental History, Rental Queue, Reviews, Recommendations, etc.
Composites

Listen to Client Developers

A map with traffic directions + weather alerts + and construction data

A user profile with 5 contacts + favorite colors + 10 latest updates

- A group of other resources
- Generally read-only
Tasks and Processes

Clear the CRUD

- Transfer $100 from A to B
- An order processing workflow
- Hiring an employee

- Spawn several resources and have their own lifecycle
Finding Resources

Domain Nouns

Composites

Tasks & Processes
HTTP Caching

GET /photo/2691065403/comments

  200 OK
  Last-Modified: Thu, 24 Jul 2008 16:25:14 GMT
  ETag: 584219-2bb-80758e80
  Cache-Control: max-age=300

GET /photo/2691065403/comments
  If-Modified-Since: Thu, 24 Jul 2008 16:25:14 GMT
  If-None-Match: 584219-2bb-80758e80

  304 Not Modified

For DB stored data, maintain version IDs and time stamps
Optimistic Concurrency

GET /photo/2691065403/comment/1

200 OK
Last-Modified: Thu, 24 Jul 2008 16:25:14 GMT
ETag: 584219-2bb-80758e80
Cache-Control: max-age=300

PUT /photo/2691065403/comment/1
If-Unmodified-Since: Thu, 24 Jul 2008 16:25:14 GMT
If-Match: 584219-2bb-80758e80

412 Precondition Failed
Tenet 4: HATEOAS

“Hypermedia as the engine of application state”
Web APIs without Hypermedia

- All URIs *prepublished*
- Clients *solely* rely on documentation
- Clients *create* URIs from scratch
- Representations are like *POJOs*
Hypermedia for Web APIs

```
<account>
  <link href="http://example.org/transaction/1"
       rel="self"/>
  <link href="http://example.org/account/1/history"
       rel="http://example.org/rels/history"/>
  <link href="http://example.org/customer/aZff13"
       rel="http://example.org/rels/customer"/>
...
</account>
```


Or define your own.
Hypermedia for Web APIs

Representations reflect app state

```xml
<customer>
  <link href="http://example.org/customer/aZff13" rel="self"/>
  <link href="http://example.org/invite?z09sa3k" rel="http://example.org/rels/verify"/>
  ...
</profile>
```

*e.g. a one time link*
HATEOAS – Consequences

- URIs are given to clients
  - Or clients use known algorithms, or URI templates
  - Don’t have to pre-publish all URIs
- URIs can be context and state sensitive
- URIs remain opaque
  - Less coupling
URI Templates

Use when client needs to supply inputs

A URI to fetch all movies with title containing a keyword:

http://example.org/movies?contains={keyword}
Describing Web APIs

Hypermedia and media types to reduce the need for description languages such as WADL
Describing Web APIs

No silver bullet

Publish a few root level URIs

OPTIONS to discover verbs

Media type specifications

Links with known relations to discover new contextual URIs
EXAMPLE: ACCOUNT TRANSFER
A client app would like to transfer $100 from one bank account to another.
Resources and URIs

Bank account: http://example.org/account/{id}

Transfers collection: http://example.org/transfers
Account transfer: http://example.org/transfer/{id}

Status: http://example.org/status/...

Perhaps upon authentication or through a previous search

Link returned upon account transfer

Prepublished, or linked from an account representation

How do client apps find these URIs?
Representations

application/vnd.example.account+xml
application/vnd.example.transfer+xml
application/vnd.example.status+xml

Describe each media type
Link Relations

"self": To self-link to each resource
"edit": Link to create a new transfer request
"http://example.org/rels/source": Source
"http://example.org/rels/target": Target
"http://example.org/rels/status": Status
Create a Transfer

POST /transfers HTTP/1.1
Host: example.org
Content-Type: application/vnd.example.transfer+xml

<transfer>
    <link href="http://example.org/account/1" rel="http://example.org/rels/source"/>
    <link href="http://example.org/account/2" rel="http://example.org/rels/target"/>
    <currency>USD</currency>
    <amount>100.00</amount>
    <note>Testing transfer</note>
</transfer>
Response

HTTP/1.1 201 Created
Location: http://example.org/transfer/1
Content-Type: application/vnd.example.transfer+xml

<transfer>
    <link href="http://example.org/transfer/1" rel="self"/>
    <link href="http://example.org/status/1?z09sa3k" rel="http://example.org/rels/status"/>
    <id>org:example:transfer:1</id>
    ...
</transfer>
Get Status

GET /status/1?z09sa3k HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
$status>
</status>
QUIZ - REVISIT
Are these RESTful?

http://example.org/persons?start=10&count=100
http://example.org/person/123.xml
http://example.org/person/123/address/4
http://example.org/movie/Gone_With_the_Wind
http://example.org/movie/Terminator/reviews

There is nothing RESTful or unRESTful.
These are just names of resources.
Is this RESTful?

GET /services/rest/?
method=flickr.photos.setPerms&photo_id=2691065403&
is_public=1&is_friend=0&is_family=0&perm_comment=1&
perm_addmeta=1 HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
Content-Type: text/xml;

<rsp stat="ok">
  <photoid>2691065403</photoid>
</rsp>
Is this RESTful?

Using GET and POST synonymously
GET is idempotent and safe. POST is not.

POST /services/rest/?
   method=flickr.photos.getRecent&api_key=... HTTP/1.1
Host: example.org
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8

<rsp stat="ok">
   <photos page="1" pages="10" perpage="100">
      <photo id="2947640330" owner="15150729@N07"           secret="..." server="3060" farm="4" title="..."           ispublic="1" isfriend="0" isfamily="0" />
      ...
   </photos>
</rsp>
Is this RESTful?

GET /photos?filterBy=recent HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
Content-Type: application/xml;

<rsp stat="ok">
  <photos page="1" pages="10" perpage="100"
total="1000">
    <photo id="2947640330" owner="15150729@N07"
secret="..." server="3060" farm="4" title="..."
ispublic="1" isfriend="0" isfamily="0" />
  ...
</photos>
</rsp>

Use links – not internal IDs
Is this RESTful?

```
GET /photos?filterBy=recent&api_key=blah
Host: example.org

200 OK
Content-Type: application/xml;charset=UTF-8

<rsp stat="fail">
  <err code="96" msg="Invalid signature"/>
</rsp>
```

Hiding errors from intermediaries and client infrastructure.

Use 4xx or 5xx codes
CHALLENGES
Key Challenges

- Modeling resources
  - Not just as data, but linked and context aware
- Respecting the uniform interface
- Programming to hypermedia
- And occasional fights between “that one” and “the other one”.

© Copyright 2008, Yahoo! Inc.
THANKS