RESTful Design: Intro, Patterns, Anti-Patterns

Stefan Tilkov
CEO & Principal Consultant
stefan.tilkov@innoq.com
http://www.innoq.com/blog/st/
Speaker’s qualifications

- CEO & Principal Consultant, innoQ (Germany/Switzerland-based consultancy)
- Strong Background in Web Services (the SOAP/WSDL kind)
- Lead SOA Editor, InfoQ
- Long-time REST advocate
- JSR 311 (Java API for RESTful Web Services) EG member
- Frequent speaker on SOA, REST, Rails
What is REST?
3 Definitions
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

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”
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”)

www.devoxx.com
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GET</strong></td>
<td>Retrieve information, possibly cached</td>
</tr>
<tr>
<td><strong>PUT</strong></td>
<td>Update or create with known ID</td>
</tr>
<tr>
<td><strong>POST</strong></td>
<td>Create or append sub-resource</td>
</tr>
<tr>
<td><strong>DELETE</strong></td>
<td>(Logically) remove</td>
</tr>
</tbody>
</table>
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>

shutdown
update software
replace hardware
startup

GET /customers/1234/orders/46
Host: example.com
Accept: application/vnd.mycompany.order+xml
(order>...</order>
What’s cool about REST?
interface Resource {
    Resource(URI u)
    Response get()
    Response post(Request r)
    Response put(Request r)
    Response delete()
}

class CustomerCollection : Resource {
    ...
    Response post(Request r) {
        id = createCustomer(r)
        return new Response(201, r)
    }
    ...
}
OrderManagementService
+ getOrders()
+ submitOrder()
+ getOrderDetails()
+ getOrdersForCustomers()
+ updateOrder()
+ addItem()
+ cancelOrder()

CustomerManagementService
+ getCustomers()
+ addCustomer()
+ getCustomerDetails()
+ updateCustomer()
+ deleteCustomer()
## Mapping Examples

<table>
<thead>
<tr>
<th>Function</th>
<th>URIs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getFreeTimeSlots(Person)</code></td>
<td>→ GET /people/{id}/timeslots?state=free</td>
</tr>
</tbody>
</table>
| `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 |
Atom & AtomPub
Atom Syndication Format

Standardized in: RFC 4287
MIME Type: application/atom+xml
Namespace: http://www.w3.org/2005/Atom

RSS Done Right
Atom Model

Feed
- id
- title
- updated

Entry
- id
- title
- updated

Content

Resource

OutOfLine
InlineText
InlineXHTML
InlineOther
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
  title="Example Feed"
  link rel="alternate" type="text/html" href="http://example.org/"/>
<link rel="self" type="application/atom+xml" href="http://example.org/feeds/23.atom"/>
<updated>2003-12-13T18:30:02Z</updated>
<author><name>John Doe</name></author>
<id>http://example.org/feeds/23</id>

<entry>
  <title>Atom-Powered Robots Run Amok</title>
  <link href="http://example.org/2003/12/13/atom03"/>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <summary>Some text.</summary>
</entry>

<entry>
  <title>A Second Contrived Example</title>
  <link href="http://example.org/2003/12/13/atom03"/>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <summary>Some text.</summary>
  <content type="xhtml" xml:base="http://example.org/"
    xmlns="http://www.w3.org/1999/xhtml">
    <div>
      <p><i>Very fine text.</i></p>
    </div>
  </content>
</entry>
</feed>
interface Resource {
    ...
}

class AtomFeed : Resource {
    AtomFeed get()
    ...
}

class CustomerCollection : AtomFeed {
    ...
}
Atom Publishing Protocol

Standardized in: RFC 5023

RESTful Collections Handling:
- Discovery, Description,
- Retrieval,
- Creation, Editing, Deletion of Resources
interface Resource {
    ...
}

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

class CustomerCollection : AtomFeed {
    ...
}
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”?
Ignoring Caching
GET /xyz

200 OK
Vary: Accept-Encoding, User-Agent
Cache-Control: max-age=7200
Expires: Tue, 30 Sep 2008 19:30:56 GMT
ETag: 283728
Ignoring Response Codes

1. Did the operation succeed?
   □ Yes

2. Everything created as intended?
   □ Yes
   □ Yes

3. Can we continue?
   □ Yes
   □ 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
Hypermedia Levels

HatEoAS

Connectedness
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
<?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</name>
    <link type="text/html" href="/customers/4711"/>
    ...
  </customer>
</customers>
```
Paging Collection

**Context**  
Result sets are too large to be retrieved at once

**Solution**  
Turn collection into resource,  
Return subset (e.g. 1st 20) on GET  
Use links for next and previous chunks

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>
  <link rel="next" href="/customers/>
  <link rel="next" href="/customers/?page=2"/>
  <customer>
    ...
  </customer>
  ...
</customers>
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
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>
  <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="..."/>
  </entry>
</feed>
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

```
GET /xyz
If-None-Match: 283728
```

304 Not Modified
Externalized Client Cache

**Context**  Simplify client caching implementation

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

```plaintext
GET /xyz
If-None-Match: 283728

304 Not Modified
```
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
http://www.oreilly.com/catalog/9780596529260/
All content and news on InfoQ 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 implementation of a full-fledged Atom Store based on Apache Abdera. The authors have created several extensions to the AtomPub specification, among them Auto-Tagging, Batching, and Aggregate Feeds. By Chris Berry & Bryon Jacob on Sep 26, 2008, Discuss

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 web services. RS, JSR 311. InfoQ had a chance to talk to spec leads Marc Hadley and Paul Sandoz. By Stefan Tilkov on Sep 26, 2008, 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 describing oriented architecture (WOA), to give business and IT leaders the bottom line about the WOA versus SOA debate. By Krishnan on Sep 22, 2008, Discuss

More news about REST >>

Articles about REST

http://www.infoq.com/REST
Thank you!
Any questions?

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