REST: SOA without Contracts?

Stefan Tilkov | innoQ | stefan.tilkov@innoq.com
What is REST?
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”
RESTful HTTP 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>...

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)
    }
    ...
}
# Mapping Examples

<table>
<thead>
<tr>
<th>Function</th>
<th>API Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>getFreeTimeSlots(Person)</td>
<td>GET /people/{id}/timeslots?state=free</td>
</tr>
<tr>
<td>rejectApplication(Application)</td>
<td>POST /rejections()</td>
</tr>
<tr>
<td></td>
<td>(\langle application\rangle http://...\langle/ application\rangle)() (\langle reason\rangle Unsuitable for us!\langle/ reason\rangle)</td>
</tr>
<tr>
<td>performTariffCalculation(Data)</td>
<td>POST /contracts()</td>
</tr>
<tr>
<td></td>
<td>Data (\langle Location: http://.../contracts/4711\rangle) (\langle Result\rangle)</td>
</tr>
<tr>
<td>shipOrder(ID)</td>
<td>PUT /orders/0815()</td>
</tr>
<tr>
<td></td>
<td>(\langle status\rangle shipped\langle/ status\rangle)</td>
</tr>
<tr>
<td>shipOrder(ID) [variation]</td>
<td>POST /shipments()</td>
</tr>
<tr>
<td></td>
<td>Data (\langle Location: http://.../shipments/4711\rangle) (\langle Result\rangle)</td>
</tr>
</tbody>
</table>
Description
The SOAP/WSDL Problem

Each application is different
Each application requires its own protocol
Need to learn a new API every single time
WSDL as formal approach for syntax only
Separation of application and metadata
## Anatomy of a WSDL File

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>XML Schema</td>
</tr>
<tr>
<td>2%</td>
<td>Message Definitions</td>
</tr>
<tr>
<td>5%</td>
<td>Operation Names, Input, Output</td>
</tr>
<tr>
<td>10%</td>
<td>Meaningless Legacy</td>
</tr>
<tr>
<td>3%</td>
<td>Address Info</td>
</tr>
</tbody>
</table>
### SOAP/WSDL

- **XML Schema**
- **Message Definitions**
- **Operation Names, Input, Output**
- **Meaningless Legacy**
- **Address Info**

- **“Informal” Documentation (Word, PDF, HTML, ...)**

### RESTful HTTP

- **XML Schema**
- **GET, PUT, POST, DELETE**

- **URIs**

- **“Informal” Documentation (Word, PDF, HTML, ...)**
RESTful HTTP Approach

Data  Operations  Identity
Data

media types
content negotiation
standard formats
XML Schema & Co.
Operations

minimal set of methods
standardized semantics
uniformity
general applicability
Identity

standardized IDs

cross-application usage

“dereferencability”

ID longevity
“RESTful” Formalisms

WSDL 2.0: Supposedly Usable for REST

- XML-focused and operation-centric
- No content negotiation
- No hypermedia Support

WADL (Web Application Description Language), https://wadl.dev.java.net/

- As RESTful as external metadata can be
- Use cases still doubtful
WADL Example

<resources base="http://api.search.yahoo.com/NewsSearchService/V1/">
  <resource path="newsSearch">
    <method name="GET" id="search">
      <request>
        <param name="appid" type="xsd:string" style="query" required="true"/>
        <param name="query" type="xsd:string" style="query" required="true"/>
        <param name="type" style="query" default="all">
          <option value="all"/>
          <option value="any"/>
          <option value="phrase"/>
        </param>
        <param name="results" style="query" type="xsd:int" default="10"/>
        <param name="start" style="query" type="xsd:int" default="1"/>
        <param name="sort" style="query" default="rank">
          <option value="rank"/>
          <option value="date"/>
        </param>
        <param name="language" style="query" type="xsd:string"/>
      </request>
      <response>
        <representation mediaType="application/xml" element="yn:ResultSet"/>
        <fault status="400" mediaType="application/xml" element="ya:Error"/>
      </response>
    </method>
  </resource>
</resources>
Conclusion(s)
1. External metadata is a problem, not a solution
2. Data, operation and identity semantics can be separated
3. The Web is more than you think it is
If You Want to Know More
http://www.innoq.com/resources/REST
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. By Stefan Tilkov on Sep 22, 2008, Discuss

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 web 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 >>
Thank you!
Any questions?

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