Introducing REST

Dong Liu
Dept. of Computer Science
Univ. of Saskatchewan
dong.liu@usask.ca
http://blogs.usask.ca/dong_notes/
Agenda

- What is REST
- RESTful services
- Comparing RESTful and SOAP-based approaches
- Useful resources
Representational State Transfer

- The term comes from Roy Fielding’s PhD work
- More practical than theoretical

It is an architectural style, not a technology, nor an architecture

- Software architecture = \{Elements, Constraints, Rationale\} *
- An architectural style abstracts elements and constraints from various specific architectures *
- Analogy: designs and design pattern

Deriving REST


RR: replicated repository
$: cache
CS: client-server
LS: layered system
S: stateless
VM: virtual machine
COD: code on demand
U: uniform interface

RR

replicated

on-demand

stateless

reliable

reliable

scalable

scalable

multi-org.

reusable

simple visible

uniform interface

layered

programmable

separated

intermediate processing

shared

extensible

REST

C$SS

CSS

LS

LCS

COD

LCODC$SS

C$SS

$
Uniform interface

- Resources are identified by ONE resource identifier mechanism
- Access methods are the same for all resources
- Resources are manipulated by exchanging representations
- Representations are in self-descriptive messages
- Hypermedia acts as the engine of application state
REST rationale

- Maximize reuse
- Minimize coupling
- Eliminate partial failure condition
- Scale without bound
- Simplify

Generic process to design RESTful services

1. Identify resources
2. Design URIs
3. Expose a subset of the uniform interface
4. Design the representations from and to the client
5. Integrate this resource into existing resources, using hypermedia links and forms
6. Consider typical flows
7. Consider error conditions

## RESTful design of del.icio.us services (1)

The example comes from Leonard Richardson & Sam Ruby "RESTful Web Services," O'Reilly, 2007.

<table>
<thead>
<tr>
<th>Operation</th>
<th>On my service</th>
<th>On del.icio.us</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a user account</td>
<td>POST /users</td>
<td>POST /register (via web site)</td>
</tr>
<tr>
<td>View a user account</td>
<td>GET /users/{username}</td>
<td>GET /users/{username} (via web site)</td>
</tr>
<tr>
<td>Modify a user account</td>
<td>PUT /users/{username}</td>
<td>Various, via web site</td>
</tr>
<tr>
<td>Delete a user account</td>
<td>DELETE /users/{username}</td>
<td>POST /settings/{username}/profile/delete (via web site)</td>
</tr>
</tbody>
</table>

The example comes from Leonard Richardson & Sam Ruby "RESTful Web Services," O'Reilly, 2007.
**RESTful design of del.icio.us services (2)**

<table>
<thead>
<tr>
<th>Operation</th>
<th>On my service</th>
<th>On del.icio.us</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post a bookmark</td>
<td>POST /users/{username}/bookmarks</td>
<td>GET /posts/add</td>
</tr>
<tr>
<td>Fetch a bookmark</td>
<td>GET /users/{username}/bookmarks/{URI-MD5}</td>
<td>GET /posts/get</td>
</tr>
<tr>
<td>Modify a bookmark</td>
<td>PUT /users/{username}/bookmarks/{URI-MD5}</td>
<td>GET /posts/add</td>
</tr>
<tr>
<td>Delete a bookmark</td>
<td>DELETE /users/{username}/bookmarks/{URI-MD5}</td>
<td>GET /posts/delete</td>
</tr>
<tr>
<td>See when the user last posted a bookmark</td>
<td>Use conditional HTTP GET</td>
<td>GET /posts/update</td>
</tr>
<tr>
<td>Fetch a user’s posting history</td>
<td>GET /users/{username}/calendar</td>
<td>GET /posts/dates (your history only)</td>
</tr>
<tr>
<td>Fetch a user’s posting history, filtered by tag</td>
<td>GET /users/{username}/calendar/{tag}</td>
<td>GET /posts/dates with querystring (your history only)</td>
</tr>
<tr>
<td>Operation</td>
<td>On my service</td>
<td>On del.icio.us</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>Fetch a user’s tag vocabulary</td>
<td>GET /users/{username}/tags</td>
<td>GET /tags/get (your tags only)</td>
</tr>
<tr>
<td>Rename a tag</td>
<td>PUT /users/{username}/tags/{tag}</td>
<td>GET /tags/rename</td>
</tr>
<tr>
<td>Fetch the list of a user’s tag bundles</td>
<td>GET /users/{username}/bundles</td>
<td>GET /tags/bundles/all (your bundles only)</td>
</tr>
<tr>
<td>Group tags into a bundle</td>
<td>POST /users/{username}/bundles</td>
<td>GET /tags/bundles/set</td>
</tr>
<tr>
<td>Fetch a bundle</td>
<td>GET /users/{username}/bundles/{bundle}</td>
<td>N/A</td>
</tr>
<tr>
<td>Modify a bundle</td>
<td>PUT /users/{username}/bundles/{bundle}</td>
<td>GET /tags/bundles/set</td>
</tr>
<tr>
<td>Delete a bundle</td>
<td>DELETE /users/{username}/bundles/{bundle}</td>
<td>GET /tags/bundles/delete</td>
</tr>
</tbody>
</table>
WS stack

Web Services Standards Overview

from http://www.innoq.com/resources/ws-standards-poster/

Jan. 8 2008

Dong Liu
Support for the Basic Profile is the baseline for interoperable Web services. Customers should demand that all of their Web services-enabled technology be compliant with the Basic Profile, and that in turn will lay the foundation for Web services to fulfill their promise and provide technology independent interoperability.

Dan Sholler, Gartner
WS-Babel

WS-Addressing

The Web is agreement

Two must-read’s

UNIVERSITY OF CALIFORNIA,
IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

Dissertation Committee:
Professor Richard N. Taylor, Chair
Professor Mark S. Ackerman
Professor David S. Rosenbaum

2000

Web Services for the Real World

Leonard Richardson & Sam Ruby
Foreword by David Heimemeier Hansson
The community

- Yahoo rest-discuss group
- Steve Vinoski
- Paul Downey
- Sam Ruby
- Bill de hÓra
- Joe Gregorio
- Mark Nottingham
- Pete Lacey
- Stefan Tilkov
- Stuart Charlton
- Mark Baker
- ...

Jan. 8 2008  Dong Liu
Thanks