Designing Read/Write Resource-Oriented Services

Kenneth M. Anderson
University of Colorado, Boulder
CSCI 7818 — Lecture 9 — 10/22/2008

Credit Where Credit is Due

- Portions of this lecture are derived from material in “RESTful Web Services” by Leonard Richardson & Sam Ruby. As such, they are Copyright 2007 by O’Reilly

Agenda

- Chapter 6: Read/Write Resource Oriented Services
- Discussion of some of the Web-based discussions that I referenced on the class website
- Also presentations on Ruby on Rails and Roy Fielding's dissertation

Last Time: Read-Only Services

- Discussed the following ROA design process
  - Figure out the data set
  - Split the data set into resources
  - For each resource
    - Name the resource with a URI
    - Expose a subset of the uniform interface
    - Design the representation accepted from the client
    - Design the representation served to the client
    - Integrate the resource with other resources using links and forms
    - Consider the typical course of events: what's supposed to happen?>
    - Consider error conditions: what might go wrong?

Last Time: URIs

- Path Variables (for discoverable resources)
  - /{planet}/[{scoping-information/][{place-name}]
- Query Variables (for algorithmic resources)
  - http://maps.example.com/Earth?show=Springfield
- Use “/” to model hierarchy (or containment) in resources
- Use “;” or “,” in URIs when dealing with non-hierarchical scoping information
  - Use “;” when order is not important
    - http://mixer.example.com/color-blends/red;blue
  - Use “,” when order is important
    - http://map.example.com/Earth/{lat},{long}

Read/Write Resource-Oriented Services

- Same process, but now we examine full range of uniform interface operations
  - Build matrix with resource types as rows, and operations as columns
    - Indicate what operations apply to which types
    - provide example URIs and discussion of what will happen
      - especially in the case of POST and PUT
        - PUT: create or modify resource
        - POST: append content to existing resource OR append child resource to parent resource (blog entries)
  - Two questions to help
    - Will clients be creating new resources of this type?
    - Who's in charge of determining the new resource's URI? Client or Server? If the former, then PUT. If the latter, then POST.

New Issues: Authentication and Authorization

- Now that we are allowing a client to change stuff on our server, we need
  - Authentication: problem of tying a request to a user
  - Authorization: problem of determining which requests to let through for a given user
- HTTP provides mechanisms to enable this (HTTP Basic/Digest) and other web services roll their own (Amazon's public/private key on subset of request)
- Another Issue: Privacy
  - Can't transmit "private information" in the clear; need to use HTTPS
- Another Issue: Trust
  - How do you trust your client software to do the right thing?
    - Especially in today's environment with malware becoming harder and harder to discern

Coming Up Next

- Chapter 10: ROA versus Big Web Services
    - Need more volunteers for presentations for lecture 10!
- Will start Web 2.0 portion of the course at lecture 11