RESTful Web Services

Stefan Marr

Agenda

- What is REST?
- The Bookmark Example
- Principles of REST Web Service Design
- Semantic of HTTP/1.1 Operations
- SOAP vs. REST Style
- Assets and Drawbacks
- Security
- Public Web Services

What is REST?

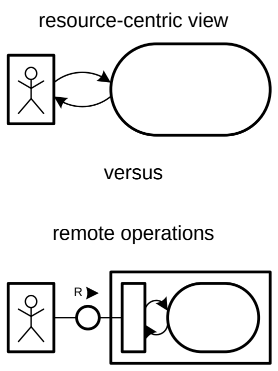
- Representational State Transfer
- Term coined by Roy Fielding [REST00]
  - Resource (item of interest)
  - Representation of resource at URI places client in a state
  - Hyperlinks are transitions between states
  - Services itself should be stateless

"The name 'Representational State Transfer' is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."

Roy Fielding

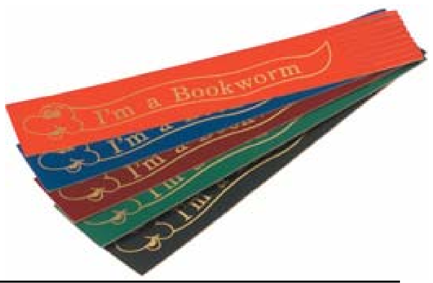
What is REST?

- Architectural style, not a standard
- "The Way the Web works", made it successful
  - Largest distributed application ever created
  - Stateless client/server architecture
  - Small set of operations applied to all resources
    - GET, POST, PUT, DELETE
  - Shared set of media-types
  - Being explicit
  - Identify persistent resources with URIs
    - Including artifact resources and application states
    - Allows bookmarking, link-sharing



The Bookmark Example

- Example for designing a web service the REST way
- Four basic questions
  - What are the resources?
  - What are the representations?
  - What methods are supported at each resource?
  - What status codes could be returned?
- What are the resources?
  - Should be nouns, not verbs
  - A single bookmark
  - Collections of bookmarks
  - Lists of keywords
  - A user profile



Principles of REST Web Service Design

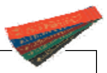
1. Identify all conceptual entities to be exposed as services
   - single bookmark, bookmark collection, keyword list
2. Create a URL to each resource
   - http://www.example.org/testuser/bookmark/45
   - Avoid RPC style and using verbs
     - /bookmarks/getBookmark?user=testuser&id=45
3. Categorize resources according to available operations
   - GET, PUT, POST, DELETE
4. Resources accessible via HTTP GET should be side-effect free

Principles of REST Web Service Design

5. No representation should be an island
   - Put hyperlinks into resource representations
   - Enables clients to obtain related or additional information.
6. Design to reveal data gradually
   - Not everything in a single response document
   - Provide hyperlinks to obtain more details
7. Specify the format of response data using a schema (DTD, XSD, ...)
   - For POST and PUT services provide a request specification
8. Describe how to invoke services using a WSDL, or an HTML document

The Bookmark Example

- Define URI space for the resources



URI	Type of resource	Description
[user]/bookmark/[id]	bookmark	A single bookmark for "user"
[user]/bookmark/newest	bookmark	The most recent bookmark for "user"
[user]/bookmarks/	bookmark collection	The 20 most recent bookmarks for "user"
[user]/bookmarks/tags/[tag]/	bookmark collection	The 20 most recent bookmarks for "user" that were filed in the category "tag"
[user]/keywords/	keyword list	A list of all the "tags" a user has ever used
[user]/profile	user profile	Information about a user
all/bookmarks/	bookmark collection	The 20 most recent bookmarks in the system

The Bookmark Example - What are the representations?

- Define a representation
- XBEL - The XML Bookmark Exchange Language

```
<?xml version="1.0"?>
<!DOCTYPE xbel PUBLIC
"+//IDN python.org//DTD XML Bookmark Exchange Language 1.0//EN//XML"
   "http://www.python.org/topics/xml/dtds/xbel-1.0.dtd">
<xbel version="1.0">
<bookmark href="http://www.xml.com/pub/a/2005/02/09/xml-http-request.html">
   <title>Very Dynamic Web Interfaces</title>
   <desc>Using XMLHttpRequest to build dynamic web interfaces.</desc>
   <info>
        <metadata owner="http://example.com/docu/xbel/baseuri">
        http://example.com/testuser/bookmark/23
        </metadata>
        <metadata owner="http://example.com/docu/xbel/tags">
        <tags><tag>xml</tag><tag>AJAX</tag></tags>
        </metadata>
   </info>
</bookmark>
</xbel>
```

Semantic of HTTP/1.1 Operations

- GET
  - “retrieve whatever information (in the form of an entity) is identified by the Request-URI”
  - Retrieve representation, shouldn’t result in data modification
- POST
  - “request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI”
    - Annotation of existing resources, extending a database through an append operation
    - Posting a message to a bulletin board, or group of articles
    - Providing a block of data (e.g. form data) to a data-handling process
  - Used to change state at the server in a loosely coupled way
- PUT
  - “requests that the enclosed entity be stored under the supplied Request-URI”, create/put a new resource
  - Used to set some piece of state on the server
- DELETE
  - “requests that the origin server deletes the resource identified by the Request-URI”

The Bookmark Example

- What methods are supported at each resource?
  - Decide which methods should be available

- Single bookmarks

Method	Representation	Description
GET	XBEL document for a single bookmark	Get a bookmark [user]/bookmark/[id] [user]/bookmark/newest
PUT	XBEL document for a single bookmark	Update a bookmark at [user]/bookmark/[id] Or insert a bookmark at [user]/bookmark/newest
DELETE	none	Delete a bookmark at [user]/bookmark/[id]

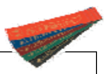
The Bookmark Example

- Bookmark collections

Method	Representation	Description
GET	XBEL document for a bookmark collection	Get a collection of bookmarks [user]/bookmarks/ [user]/bookmarks/tags/[tag]/
POST	XBEL document for a single bookmark	Add bookmark to a collection [user]/bookmarks/

The Bookmark Example

- Keyword lists and user data



Method	Representation	Description
GET	keyword list document	Get a list of all used keywords [user]/keywords/
GET	user profile document	Get the user profile [user]/profile
POST	partial user profile May be multipart/form-data encoded	Update some detail of user profile with an html form [user]/profile

- What status codes could be returned?
  - 201 on successful creation of a bookmark
  - Standard codes
    - 200 OK and 3xx, 4xx, 5xx for redirection, errors, etc.

SOAP vs. REST Style

- Criticism on SOAP from REST point of view
  - Redefines semantic of HTTP operations
  - Doesn’t comply with web architecture
  - POST a SOAP message to a URI
    - POST is meant to add a subordinate to a resource
    - SOAP Requests i.e. a method call
  - Using Status Code 200 for SOAP Errors
- SOAP 1.2
  - Binding of SOAP to HTTP intended to make appropriated use of HTTP as application protocol
  - It is possible to retrieve a application state via GET
  - Correct usage of status codes (200, 4xx, 5xx)
- Pro SOAP
  - Rigid – strong typing, interface contract
  - Rich support, tools for code and modeling

REST Assets and Drawbacks

- Assets
  - Development and testing without complex toolkits
  - Debugging of REST requests with a web browser
  - Requires a basic HTTP client, available in every common language
  - REST services can be easily used by AJAX applications
  - APIs in REST style are more “consumable” then complex APIs
    - Lower learning curve for consumer
    - Everything accessible through universal API
- Drawbacks
  - Only few tools
  - Restrictions for GET length sometimes may be a problem
  - No direct bridge to the OOP world
    - Difference between REST and RPC style may be subtle sometimes, but not necessarily in general case

Security with REST

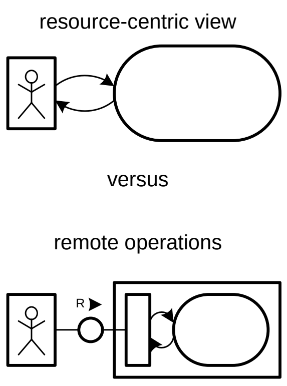
- Firewall
  - Operations based on URIs and HTTP methods
  - Can be filter by firewalls
  - No need to inspect and parse e.g. SOAP
- Server Side
  - Simple ACL based security possible
  - Security and authentication through HTTP(S)
  - Request and response data may be secured by OASIS Web Services Security

Public Web Services

- Amazon
  - http://www.amazon.com/gp/browse.html/ref=sc_fe_l_3/102-5668266-0869764?&node=3487571
  - Complete API in both flavors
- Ebay
  - http://developer.ebay.com/
  - REST API only to access public information
  - Full support only available over SOAP and a proprietary XML API
- Google
  - http://www.google.com/apis/
  - No REST API, only SOAP
- Blogger.com ATOM API
  - http://code.blogger.com/archives/atom-docs.html
  - Blogging API is RESTful

Summary

- REST is
  - A architectural style, no standard
  - "The Way the Web works"
  - Resource-centric
  - Based on mature standards
  - Lightweight, compared to SOAP
- It's another philosophy then OOP
- Currently very few tools to support development (Axis2)
- But actually no tools necessary
- Security advantages over SOAP from network admin point of view



References

- [REST00] Architectural Styles and the Design of Network-based Software Architectures, Roy Thomas Fielding, 2000
  http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- How to create a REST Protocol
  http://www.xml.com/pub/a/2004/12/01/restful-web.html
- Show me the Code – A RESTful Bookmark Web Service
  http://www.xml.com/pub/a/2005/03/02/restful.html
- [XBEL] The XML Bookmark Exchange Language
  http://pyxml.sourceforge.net/topics/xbel/
- REST Wiki
  http://rest.blueoxen.net/cgi-bin/wiki.pl?FrontPage
- HTTP/1.1 RFC 2616 June 1999
- Building Web Services the REST Way
  http://www.xfront.com/REST-Web-Services.html
- RESTful Web Services, John Cowan, 2005
  http://mercury.ccil.org/~cowan/restws.pdf
- REST, Roger L. Costello and Timothy D. Kehoe
  www.xfront.com/REST-full.ppt
- REST, RPC, mountains, molehills, and a retraction (sort of), Jeff Bone, 2001
  http://www.xent.com/pipermail/fork/2001-August/002801.html

RESTful Web Services

Q & A

Stefan Marr