

]project-open[Open Source Enterprise Business Application

]po[RESTful API

Version: 0.2
Date: 12th Mai 2008
Author: Klaus Hofeditz
Klaus.hofeditz@project-open.com



Definition

- Representational State Transfer (REST) is a software architectural style for distributed hypermedia systems like the world wide web. The term originated in a 2000 doctoral dissertation Architectural Styles and the Design of Network- based Software Architectures about the web written by Roy Fielding, one of the principal authors of the HTTP protocol specification, and has quickly passed into widespread use in the networking community.

REST strictly refers to a collection of architectural principles Principled Design of the Modern Web Architecture (described below). The term is also often used to describe any simple interface that uses XML (or YAML, JSON, plain text) over HTTP without an additional messaging layer such as SOAP.

- REST is not a standard. REST is just an architectural style. REST is using the following standards: HTTP, URL, XML/HTML/GIF/JPEG/etc (Resource Representations) text/xml, text/html, image/gif, image/jpeg, etc (MIME Types)

<http://rest.blueoxen.net/cgi-bin/wiki.pl?WhatIsREST>

REST - Principles

- Application state and functionality is divided into resources
Any state retained by the server must be modeled as a resource
- Every resource is uniquely addressable using a universal syntax for use in hypermedia links
- Resources are identified by URIs - each resource should have its own URI
- All resources share a uniform interface for the transfer of state between client and resource, consisting of
 - * A constrained set of well-defined operations
 - * A constrained set of content types, optionally supporting code-on-demand
- A protocol that is: * Client/Server, * Stateless, * Cacheable, * Layered
- GET used for queries, and only for queries, PUT, POST, and DELETE can all change state
- As a matter of style URIs should not reveal the implementation technique used
- Cache: to improve network efficiency responses must be capable of being labeled as cacheable or non-cacheable.
- Specify the format of response data using a schema
- Describe how your services are to be invoked
- The application produces *representations*, not resources.
- A user should never get at the implementation of an object

<http://rest.blueoxen.net/cgi-bin/wiki.pl?WhatIsREST>

http://ajaxpatterns.org/RESTful_Service#GET_used_for_queries.2C_and_only_for_queries

REST – Principles II

- Every request is idempotent -- Basically, this means that resources are designed and implemented so that a client can make the same request more than once without "ill" effects. In particular, a second request for a resource that causes a specific side-effect might only produce the side-effect the first time. Perhaps a few examples would help:
- If the client (end user) makes a request to order a book a second time, the application will send the user a form that enables the user to update the number of copies ordered, rather than automatically ordering another copy.
- If the client attempts to create a "user account" on the server, and tries to do so more than once, then the server will return the user's account information (so that the user can update it) rather than creating a new duplicate account.

<http://rest.blueoxen.net/cgi-bin/wiki.pl?WhatIsREST>

Principles: REST-API Design

- URIs should only contain nouns
- Design to reveal data gradually
- Do it right the first time, you have only one chance!
- When in doubt leave it out (you can always add)
- Be persistent (remove/delete)

- Case: XML is expected, internal error occurs, error code needs to be returned, what representation has thi error code
- The only valid state transfer methods are, GET, PUT,POST, DELETE
GET transfers a representation of the state of a resource from the server to the client, PUT transfers a representation of the state of a resource from the client to the server, POST transfers a representation of the state of a *new* resource from the client to the server. The server's response includes the location of the created resource.DELETE transfers the "deleted" state to a resource.

<http://rest.blueoxen.net/cgi-bin/wiki.pl?BenjaminsRESTTutorial>
<http://rest.blueoxen.net/cgi-bin/wiki.pl?RestFaq>

REST vs. SOAP vs. XML-RPC

- XML-RPC -> requests as XML
- A REST web application requires a different design approach than an RPC (Remote procedure call) application. An RPC application is exposed as one or more network objects, each with an often unique set of functions that can be invoked. Before a client communicates with the application it must have knowledge of the object identity in order to locate it and must also have knowledge of the object type in order to communicate with it.

REST design constrains the aspects of a resource that define its interface (the verbs and content types). This leads to the definition of fewer types on the network than an RPC-based application but more resource identifiers (nouns). REST design seeks to define a set of resources that clients can interact with uniformly, and to provide hyperlinks between resources that clients can navigate without requiring knowledge of the whole resource set. Server-provided forms can also be used in a RESTful environment to describe how clients should construct a URL in order to navigate to a particular resource.

REST - General Challenges

- Consistency - how will developers know what to expect?
- Authentication
- Versioning
- Side effects
- Depth of representation
- List – Limits, how many items of a resource should be delivered

Authentication

- Authentication:
“I imagine implementing a hash that works similar to how the session cookies work, where a component of the hash requires re-authenticating after a certain period and it ties in to existing session procs. Since this is between machines, maybe it could use the shortest of the session intervals, SessionRenew (or a combination of SessionRenew and SessionTimeout) in the kernel parameters etc.” (Torben Brosten)
- HTTP Authorization (Basic/Digest)
- "RFC 2104 - Keyed-Hashing for Message Authentication"
(<http://www.ietf.org/rfc/rfc2104.txt>)
- “Users are accustomed to having a log out button that enables them to log out of a site before leaving a public terminal. The HTTP authentication protocols provide no way for the server to request that the client erase the credentials for a realm other than prompting for them again, which causes the browser to pop up the dialog again. Today's browsers do not themselves provide a way to log out of a realm, other than quitting the browser.”

<http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAuthentication.html>
<http://www.artima.com/weblogs/viewpost.jsp?thread=155252>

HTTP authentication workarounds

- “Optional authentication can be achieved by using cookies to indicate whether the user has authenticated, while using HTTP authentication for the actual authentication. A sign in page can be created that provides "Forgot your password" and "Sign Up" links, an explanation of what to type into the browser's dialog (i.e., Artima ID or email and password), and includes a button to press that will actually cause the HTTP authentication dialog to appear. Even better, if JavaScript is enabled on the client, some JavaScript embedded in that sign in page can replace the explanation and button with a user-friendly login form. JavaScript can capture the submit, and do the HTTP authentication itself.”
- “In addition, if JavaScript is enabled on the client, there are even ways to effectively log people off of an HTTP authentication session. For example, if you require that passwords are at least six characters long, then no one can have the password "abc". When you want to log someone out because they've pressed the "Log Out" button, some JavaScript on the client can run and do an HTTP authentication for the realm you want to log the user out of at a special URI with the user's username and the password "abc". This authentication will succeed at that special URI, which now means that the next time the client visits any other URI in that realm, it will attempt to authenticate with the wrong password. You have effectively logged the user out of his or her authentication session for that realm.”

<http://www.artima.com/weblogs/viewpost.jsp?thread=155252>

Implementation Designs & Objectives

Objectives:

- Easy to use (hard to misuse),
- Easy to learn
- Easy to extend
- Sufficiently powerful to satisfy requirements
- Agility trumps completeness

Rules

- URIs should only contain nouns

Clarify:

- Transmission of Error codes – http errors / vs. Ressource errors

REST w/ AOL Server

<http://panoptic.com/aolserver/chat/20080419.html>

IRC [14:31] <justis> Which was: how do I get variables that are represented in a RESTful style URL?
IRC [14:31] <justis> I live in Raleigh, NC.
IRC [14:31] <Dossy> Aha. ns_queryget :)
IRC [14:31] <Dossy> and/or ns_querygetall, depending
IRC [14:32] <justis> Oh, really? ns_queryget? Gotta check that out.
IRC [14:32] <justis> I ended up using ns_urllv
IRC [14:32] <Dossy> ha. yeah, ns_queryget for one value, or ns_querygetall for a Tcl list of values
IRC [14:32] <Dossy> i.e., foo.adp?a=1&b=2&c=3&a=4&a=5
IRC [14:33] <Dossy> [ns_queryget a] should be [list 1]. [ns_querygetall a] should be [list 1 4 5]
IRC [14:33] <Dossy> Parsing ns_urllv "works" too, but ...
IRC [14:33] <justis> Oh, that's not going to do. By "variables in a RESTful" URL, I meant prior to the query string: /album/345/track/3
IRC [14:33] <Dossy> Aha. Yeah, for that, you want to use [ns_conn urllv] - heh.
IRC [14:33] <justis> OK, cool.
IRC [14:34] <Dossy> And probably use ns_register_adp or ns_register_proc for the endpoint URL prefix.
IRC [14:34] <justis> I was contemplating using ns_register_filter and getting it to convert the URL into a more traditional query.
IRC [14:34] <Dossy> Or, yeah - you could use registered filters ... there's several different ways to do it, depending on how you prefer to design.
IRC [14:34] <justis> Yeah, I ended up doing the endpoint URL. Something like /album from the example above.
IRC [14:34] <Dossy> Yup.
IRC [14:34] <Dossy> It's remarkably simple.
IRC [14:35] <justis> I want to explore the filters, but I'm in "get it done" mode, so I went with the low hanging fruit.
IRC [14:35] <Dossy> No "beans," no "web.xml," no "containers," no "scaffolding" or "framework" ... :)
IRC [14:35] <justis> My disappointment was that it made me go through another proc in the middle that was reproducing behavior that was already modeled well at the ns_ level.
IRC [14:35] <justis> ns_register_proc, specifically.
IRC [14:35] <Dossy> Heh.
IRC [14:35] <justis> But "get it done" won out.
IRC [14:36] <Dossy> Sure. You're talking one line of code, heh.
IRC [14:37] <justis> Yeah, I'm starting to think that it could be a very easy way to quickly prototype RESTful applications, which is a philosophy that has also thrown away a lot of the need for all the extra stuff in most frameworks.
IRC [14:39] <justis> The question currently in my head is whether there are any "cloud computing" providers for AOLserver. One of my favorite aspects of ReST is its ability to scale.
IRC [14:42] <justis> Admittedly, AOLserver scales very well for my client. That, with "pound" in front of it makes for a pretty graceful machine.
IRC [14:43] <justis> For my own apps, though, I have no budget for scaling and I like the idea of architecting them for cloud computing from day one.
IRC [14:45] <justis> The apps I want to write, personally, are mostly aimed at social applications that would target users bases with grand scale, like Twitter, Facebook, OpenSocial, etc. Thus, they all run the "risk" of garnering a huge user base in a short period of time.
IRC [14:45] <justis> That "risk", of course, is part of the allure :)
IRC [14:47] <justis> I guess it would be "risk" in the way that PMI speaks of project risk, which can be either positive or negative consequence, and changes the project in a significant way.
IRC [14:49] <justis> So, where do you live? Is it a gorgeous day where you are, too?
IRC [14:50] <justis> It's in the high 70s here in Raleigh.
IRC [14:53] <justis> I've gotta go AFK for a few minutes. I hope to chat with you again soon. It's very nice work, this AOLserver, and I'm curious to learn more about it. As you pointed out, having ns_register_proc as a single line can be really elegant. It's kinda getting me hooked.
IRC [15:05] <Dossy> It's gorgeous here - Butler, NJ - 77F and not a cloud in the sky.
IRC [15:05] <Dossy> Scaling AOLserver is more a function of how you design your application rather than AOLserver's capability itself.
IRC [15:06] <Dossy> If you know how to scale a web application horizontally (no host-local storage, etc.) in any other platform, AOLserver is no different.

Next Steps

- Gather requirements (use cases)
- Modeling the persistent resources (entities that have a typical life expectancy of days or greater), ensure that each instance has a unique URL. When possible, assign meaningful names to resources.
- Define XML Schema for each representation
- Add high level methods which take care of all composite create, update, and delete operations.
 - Give it a GET method that returns XML conforming to the schema.
 - Give it a PUT method that updates the underlying database for every transaction.
 - Give it a DELETE method for removing relevant data from the database.

http://www.prescod.net/rest/Steps_to_extreme.html

URI identifiers:

Considerations when building Identifiers:

- Browser: back-button handling and bookmarking;
- WebServer: Caching, Compilation, etc.
- Network routers and caches all along the way; robot applications such as those crawling the web on behalf of search engines; personal web agents which trawl through the web on behalf of individuals
- Ideally, a service should be intuitive and *self-documenting*

Options:

- CGI-style body: id=995&bluebaggers=150&redlegs=60
- Command-qualified URL:
`http://example.com/matchMaintenance/command=newMatch`

]po[Identifier nouns:

Principal Identifiers:

- <http://project-open.dnsalias.com/rest/projects>
- <http://project-open.dnsalias.com/rest/tasks>
- <http://project-open.dnsalias.com/rest/users>
- <http://project-open.dnsalias.com/rest/timesheet>

2nd Level Identifiers:

- Project by Project Nr.
<http://project-open.dnsalias.com/rest/project/345> (logical URI)
get's project w/ No. 345

Resource: Project

Schema to use:

Microsoft Project Schema (Project Server)

Representation Format:

- XML

Representation Schema:

- See <http://msdn.microsoft.com/en-us/library/aa679870%28office.11%29.aspx>

Resource: User

Representation Format:

- XML

Representation Schema:

```
<user xmlns='http://example.org/my-example-ns/'>  
  <name>Full name goes here.</name>  
  <title>Persons title goes here.</title>  
  <phone-number>Phone number goes here  
  .</phone-number>  
</user>
```

tbc...

]po[Documentation

- Document extensive and accurate
- Document every class, every method, interface, constructor, parameter (units, form ownership)
- Add search interface in 2nd phase

Sources: Reading order

- a) <http://rest.blueoxen.net/cgi-bin/wiki.pl?RestFaq>
- b) <http://www.prescod.net/rest/>
- c) http://ajaxpatterns.org/RESTful_Service#Motivating_REST
- d) <http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>
- e) RESTful Service http://ajaxpatterns.org/RESTful_Service
- f) REST Wiki <http://rest.blueoxen.net/cgi-bin/wiki.pl?FrontPage>
- g) http://www.project-open.org/product/modules/xml-rpc/decision_thread.html
- h) The power of the URL-line - RPC-style services don't replace the humble URL <http://207.22.26.166/bytecols/2001-08-15.html>
- i) Common REST mistakes <http://www.prescod.net/rest/mistakes/>
- j) <http://www.infoq.com/presentations/effective-api-design>
- k) How to create a REST Protocol
<http://www.xml.com/pub/a/2004/12/01/restful-web.html>
- l) REST forAL Server http://www.rexx.com/~dkuhlman/rest_howto.html

Thanks for your attention



]project-open[

Ronda Sant Antonio 51, 1o 2a
08011 Barcelona
Spain

Tel: +34 933 250 914

Cell: +34 609 953 751

Fax: +34 932 890 729

www.project-open.com

www.project-open.org

wastebasket