Atom: From Blogging to Data, Web (Services) 2.0

Scott Davis
Davisworld Consulting
Introduction

- My name is Scott Davis
  - Editor in Chief of http://aboutGroovy.com
- Author
  - Groovy Recipes: Greasing the Wheels of Java (Pragmatic Bookshelf)
  - GIS for Web Developers (Pragmatic Bookshelf)
  - Google Maps API (Pragmatic Bookshelf)
  - JBoss At Work (O’Reilly)
Web Services

- There are three major types of Web Services in common use today:
  - SOAP
  - REST
  - JSON

*See Real World Web Services:*
- SOAP is the dialect that everyone is most familiar with, but it is falling out of favor:
The creator of XML-RPC, SOAP, and RSS says this at 42:00 in the podcast:

“I love SOAP. SOAP was the lawyer of the family. It was the one that had the potential to bring everyone together.

But you’d have to say at this point that SOAP is a failure. How long are we supposed to wait for the magic of SOAP to happen?”

Dave Winer
http://www.itconversations.com/shows/detail260.html
“[With SOAP] we tried really hard to create a level playing field for small developers and large developers.”

Dave Winer
Beyond the SOAP Search API
Dec 20, 2006 - Permalink

Post by Mark Lucovsky, Software Engineer

On December 5th, we stopped accepting new sign-ups for the Google SOAP Search API. This change does not impact current users of the SOAP Search API -- you can continue to execute queries, and we have no plans to turn off the service in the future.

While the product was Google's first API and inspired a lot of Google's current developer products, we are no longer devoting resources to increase the capacity of the service, instead focusing our efforts on the AJAX Search API. While the AJAX Search API does not provide server-side access to search results, it has a number of more powerful features, including access to Video, Maps, Blog Search, and News search results.
Google Data APIs (Beta) Developer's Guide

Use Data from Google on Your Own Website

Get Started with GData

The Google data APIs ("GData" for short) provide a simple standard protocol for reading and writing data on the web.

GData uses either of two standard XML-based syndication formats: Atom or RSS. It also has a feed-publishing system that consists of the Atom publishing protocol plus some extensions (using Atom's standard extension model) for handling queries.

Each of the following Google services provides a Google data API:

- [Google Apps Provisioning](#)
- [Google Base](#)
- [Blogger](#)
- [Google Calendar](#)
- [Google Code Search](#)
- [Google Notebook](#)
- [Google Spreadsheets](#)
So effectively:

- SOAP == Web Services 1.0
- Atom == Web Services 2.0
But before we get too far, let’s make sure that we have a solid understanding of an important term: REST
What is REST?

- The other white meat:

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 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.

Systems that follow Fielding's REST principles are often referred to as RESTful; REST's most zealous advocates call themselves RESTafarians.

Creating a REST Request

The Yahoo! Search Web Services are all REST services. That means you can easily construct request URLs that will work in your browser, on the command line, and in your code. For our example, we'll construct a query to search the web for PDF files containing the term "finances". See the web search documentation to understand the details of this service.
The method is followed by the actual query parameters, which take the form argument=value, where the arguments and values are url encoded. Multiple parameters are separated by an ampersand (&).

http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=jboss+at+work
<Resultset xsi:schemaLocation="urn:yahoo:srch http://api.search.yahoo.com/WebSearchService">
  <Result>
    <Title>
      Amazon.com: JBoss at Work: A Practical Guide: Books
    </Title>
    <Summary>
      Amazon.com: JBoss at Work: A Practical Guide: Books by Tom Marrs, Scott Davis ...
    </Summary>
    <Url>
    </Url>
    <ClickUrl>
      http://uk.wrs.yahoo.com/_ylt=A9htfSGYz_pDVWAATqDdmMwF:_ylu=X3oDMTBjcV
    </ClickUrl>
    <ModificationDate>1137312000</ModificationDate>
    <MimeType>text/html</MimeType>
    <Cache>
      <Url>
        http://uk.wrs.yahoo.com/_ylt=A9htfSGYz_pDVWAAU6DdmMwF:_ylu=X3oDMTA
      </Url>
      <Size>112159</Size>
    </Cache>
  </Result>
</Resultset>
GETful vs. RESTful

Q: What is REST?
REST stands for *Representational State Transfer*. Most of the Yahoo! Web Services use "REST-Like" RPC-style operations over HTTP GET or POST requests with parameters URL encoded into the request.

For more information about REST please check out the following:
- [The Beauty of REST](#) by [Jon Udell](#)
- [Second Generation Web Services](#) by [Paul Prescod](#)
- [Representational State Transfer](#) in the [Wikipedia](#)
“Pure” REST

- Transport: HTTP
  [GET, PUT, DELETE, POST]
- Request format: URI
  - The URI remains the same, the methods change
  - The URI is the “Primary Key” of the object
- Response format: XML
RPC

REST versus RPC

A REST web application requires a different design approach than an RPC application. In RPC, the emphasis is on the diversity of protocol operations, or *verbs*; for example, an RPC application might define operations such as the following:

```python
getUser()
addUser()
removeUser()
updateUser()
getLocation()
addLocation()
removeLocation()
updateLocation()
listUsers()
listLocations()
findLocation()
findUser()
```

(Source: http://en.wikipedia.org/wiki/REST)
With REST, on the other hand, the emphasis is on the diversity of resources, or *nouns*; for example, a REST application might define the following two resource types:

```
User {}
Location {}
```

Each resource would have its own location, such as `http://www.example.org/locations/us/ny/new_york_city`. Clients work with those resources through the standard HTTP operations, such as GET to download a copy of the resource, PUT to upload a changed copy, or DELETE to remove all representations of that resource. Note how each object has its own URL and can easily be cached, copied, and bookmarked. POST is generally used for actions with side-effects, such as placing a purchase order, or adding some data to a collection.

What is Atom?

For a great example of a “Pure” RESTful WS, let’s take a look at Atom

Atom is the name of a specific web feed format. Web feeds, from a user's perspective, allow Internet users to subscribe to websites that change or add content regularly. To use this technology, site owners create or obtain specialized software (such as a content management system) which, in the machine-readable XML format, presents new articles in a list, giving a line or two of each article and a link to the full article or post.

(Source: http://en.wikipedia.org/wiki/Atom_%28standard%29)

➢ Atom is an IETF standard format for weblogs
  • RSS is a non-standardized, RPC-centric protocol
  • Most applications support both RSS and Atom
Why Atom?

- Standards are your friend
  - Exhibit A:
    Java: Write Once, Run Anywhere
  - Exhibit B:
    JavaScript: Write Once, Debug Everywhere

- Atom is an IETF standard (RFC 4287)
  - RSS is... well... umm...
# RSS’ Troubled Childhood

<table>
<thead>
<tr>
<th></th>
<th>When?</th>
<th>Who?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSS 0.9</td>
<td>Jan 1999</td>
<td>Netscape</td>
</tr>
<tr>
<td>RSS 0.91</td>
<td>Jul 1999</td>
<td>Netscape</td>
</tr>
<tr>
<td>RSS 0.91</td>
<td>Jun 2000</td>
<td>Dave Winer</td>
</tr>
<tr>
<td>RSS 1.0</td>
<td>Dec 2000</td>
<td>RSS-DEV</td>
</tr>
<tr>
<td>RSS 0.92</td>
<td>Dec 2000</td>
<td>Dave Winer</td>
</tr>
<tr>
<td>RSS 0.93</td>
<td>Apr 2001</td>
<td>Dave Winer</td>
</tr>
<tr>
<td>RSS 0.94</td>
<td>Jun 2002</td>
<td>Dave Winer</td>
</tr>
<tr>
<td>RSS 2.0</td>
<td>Aug 2002</td>
<td>Dave Winer</td>
</tr>
<tr>
<td>RSS 2.0.1</td>
<td>Jul 2003</td>
<td>Harvard</td>
</tr>
</tbody>
</table>

Source: RSS and Atom in Action
What is RSS?

- RSS has stood for at various times:
  - RDF Site Summary
  - Rich Site Summary
  - Really Simple Syndication
  - Nothing: just three plain old letters

*It’s a pity nobody used an ‘S’ for “Standardization” …*
RSS is a tale of two cities:

- Netscape and Semantic Web folks wanted to use RDF
- Dave Winer took the populist approach, wanting it to be as simple as possible

The result is a splintered landscape and bitter in-fighting that has succeeded in spite of itself

http://en.wikipedia.org/wiki/RSS_(file_format)
Atom and the Blogosphere

- For example, my blog is at the URI
  http://www.davisworld.org/blojsom/blog/

  ➢ You can read it using a standard web browser or specialized software called a “feed reader” or “news aggregator”
    - The aggregator retrieves the XML and formats it independently of the L&F of the website
    - It is like a combination of a web browser and an email client
      ✓ I subscribe to different feeds, and new/unread entries “appear” in my “inbox”
Each new “feed” is a “syndication”

- The URI remains the same, but new data can always be found at the same address
  - Each entry has a unique URI as well — the “permalink”
    - http://www.davisworld.org/blojsom/blog/default/2006/02/01/Evolve.html
  - Since the resources are accessed via a simple GET, I can easily bookmark them
The Atom Publishing Protocol is fully RESTful

The Atom Publishing Protocol uses HTTP to edit and author web resources. The Atom Protocol uses the following HTTP methods:

- **GET** is used to retrieve a representation of a resource or perform a query.
- **POST** is used to create a new, dynamically-named resource.
- **PUT** is used to update a known resource.
- **DELETE** is used to remove a resource.

You can use the ROME framework to interact with Atom and RSS feeds programmatically

> https://rome.dev.java.net/
Let’s examine [http://www.softwaresummit.com/coloradosoftwaresummit.atom](http://www.softwaresummit.com/coloradosoftwaresummit.atom)
<feed xmlns="http://www.w3.org/2005/Atom"
    xml:base="http://www.softwaresummit.com/coloradosoftwaresummit.atom"
    xml:lang="en-us">
  <title>Colorado Software Summit</title>
  <id>http://www.softwaresummit.com/</id>
  <link href="http://www.softwaresummit.com/" />
  <link rel="self" href="coloradosoftwaresummit.atom" />
  <logo>graphics/atomfeed.png</logo>
  <icon>favicon.ico</icon>
  <updated>2007-09-17T18:36:11-07:00</updated>
  <author>
    <name>Wayne Kovsky</name>
  </author>
  <rights>Copyright (c) 2007 Kovsky Conference Productions Inc.</rights>
</feed>
Atom Entry

<entry xml:base=".">
  <title>September 17, 2007</title>
  <link href="2007/whatsnew.htm#20070917"/>
  <id>http://softwaresummit.com/2007/whatsnew.htm#20070917</id>
  <published>2007-09-17T18:01:11-07:00</published>
  <updated>2007-09-17T18:36:11-07:00</updated>
  <content type="xhtml">
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Added this speaker and topic to the 2007 agenda:</p>
      <ul>
        <li><a href="../2007/speakers/soyring.htm">John Soyring</a>, Vice President, Solutions and Software, IBM Corporation: "Keynote: Conference Opening Keynote Presentation"</li>
      </ul>
      <p>We have added several new books to the page that contains books that are of particular interest to anyone attending Colorado Software Summit 2007. The books on this list are referenced by one or more of our speakers in their presentations for Colorado Software Summit 2007, or they were written by one of our speakers at Colorado Software Summit 2007: <a href="../2007/books2007.htm">Books for Colorado Software Summit 2007</a>. We may add some books to this list in the next few weeks.</p>
    </div>
  </content>
</entry>
Google Calendar Data API Overview

Start using the Calendar data API

Google Calendar allows client applications to view and update calendar events in the form of Google data API ("GData") feeds. Your client application can use the Google Calendar data API to create new events, edit or delete existing events, and query for events that match particular criteria.

Here are some of the things you can do with the Calendar data API:

- Create a web front end to let people view their Google Calendar information from within your site.
- Publicize upcoming events by programmatically adding them to Google Calendar.
- Mix and match Google APIs—for example, display upcoming events in Google Maps by combining the Google Calendar data API and the Google Maps API.
GData Authentication

ClientLogin username/password authentication

The ClientLogin method requires having access to the username and password for a Google Account. To receive an authentication token using the ClientLogin mechanism, send a POST request to the following URL:

https://www.google.com/accounts/ClientLogin

The POST body should contain a set of query parameters, as described in the following table. They should look like parameters passed by an HTML form, using the application/x-www-form-urlencoded content type.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>The user's email address.</td>
</tr>
<tr>
<td>Passwd</td>
<td>The user's password.</td>
</tr>
<tr>
<td>source</td>
<td>Identifies your client application. Should take the form companyName-applicationName-versionID; below, we'll use the name exampleCo-exampleApp-1.</td>
</tr>
<tr>
<td>service</td>
<td>The string cl, which is the service name for Google Calendar.</td>
</tr>
</tbody>
</table>

If the authentication request fails, you'll receive an HTTP 403 Forbidden status code.

If it succeeds, then the response from the service is an HTTP 200 OK status code, plus three long alphanumeric codes in the body of
GData Header

<feed xmlns='http://www.w3.org/2005/Atom'
     xmlns:gd='http://schemas.google.com/g/2005'>
  <id>http://www.google.com/calendar/feeds/jo@gmail.com/private-magicCookie/full</id>
  <updated>2006-03-29T07:35:59.000Z</updated>
  <title type='text'>Jo March</title>
  <subtitle type='text'>This is my main calendar.</subtitle>
  <link rel='http://schemas.google.com/g/2005#feed'
        type='application/atom+xml'
        href='http://www.google.com/calendar/feeds/jo@gmail.com/private-magicCookie/full'/>
  <link rel='self' type='application/atom+xml'
        href='http://www.google.com/calendar/feeds/jo@gmail.com/private-magicCookie/full'/>
  <author>
    <name>Jo March</name>
    <email>jo@gmail.com</email>
  </author>
  <generator version='1.0' uri='http://www.google.com/calendar/'>CL2</generator>
  <gd:where valueString='California'/>
</feed>
GData Entry

<entry>
  <id>http://www.google.com/calendar/feeds/jo@gmail.com/private-magicCookie/full/entryID</id>
  <published>2006-03-30T22:00:00.000Z</published>
  <updated>2006-03-28T05:47:31.000Z</updated>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/g/2005#event'></category>
  <title type='text'>Lunch with Darcy</title>
  <content type='text'>Lunch to discuss future plans.</content>
  <link rel='alternate' type='text/html'
    href='http://www.google.com/calendar/event?eid=aTJxcnNqbW9tcTJnaTE5cnMybmEwaW04bXMgbWFyY2guam9AZ21haWwuY29t'
    title='alternate'></link>
  <link rel='self' type='application/atom+xml'
    href='http://www.google.com/calendar/feeds/jo@gmail.com/private-magicCookie/full/entryID'></link>
  <author>
    <name>Jo March</name>
    <email>jo@gmail.com</email>
  </author>
  <gd:transparency
    value='http://schemas.google.com/g/2005#event.opaque'></gd:transparency>
  <gd:eventStatus
    value='http://schemas.google.com/g/2005#event.confirmed'></gd:eventStatus>
  <gd:comments>
    <gd:feedLink
      href='http://www.google.com/calendar/feeds/jo@gmail.com/private-magicCookie/full/entryID/comments/'></gd:feedLink>
  </gd:comments>
  <gd:when startTime='2006-03-30T22:00:00.000Z'
    endTime='2006-03-30T23:00:00.000Z'></gd:when>
  <gd:where></gd:where>
</entry>
Adding a new Calendar entry

POST http://www.google.com/calendar/feeds/default/private/full

The content that you send with the POST request should be the <entry> element you created above, using the application/atom+xml content type.

When you send that second POST request (or the first one in cases where there's no redirect), Calendar creates a calendar event, then returns an HTTP 201 CREATED status code, along with a copy of the new event in the form of an <entry> element. The returned entry is similar to the one you sent, but the returned one contains various elements added by Calendar, such as an <id> element.
For More Information…

Web Services for the Real World
Leonard Richardson & Sam Ruby
Foreword by David Heinemeier Hansson
O’REILLY

Understanding Web Internals
David Gourley & Brian Totty
with Marjorie Sayer, Sails Reidy & Arshin Agarwal
O’REILLY

HTTP
The Definitive Guide
O’REILLY
Conclusion

- Effectively:
  - SOAP == Web Services 1.0
  - Atom == Web Services 2.0
Conclusion

- Thanks for your time!

➤ Questions?

➤ Email: scottdavis99@yahoo.com

➤ Download slides and code examples:
  - http://www.davisworld.org/presentations