A RESTful Web service, an example

It's often hard for people to "get" REST, this is mostly due to the fact that REST isn't a tangible thing like a piece of software or even a specification, it's a selection of ideals, of best practices distilled from the HTTP specs.

I've always found that the best way to understand something is to see an example, to see the principles in action first and worry about the details later once I understand the general gist. So here's a little example of a RESTful version of a simple Web service you might already know about, the Delicious API.

Delicious has "a simple REST API", or rather, a simple POX over HTTP API, that is, it has a perfectly usable HTTP and XML based API for accessing your bookmarks and tags, but it isn't very RESTful. Why not?

- First class objects aren't exposed as resources, so we can't access our bookmarks or tags directly.
- HTTP methods not used correctly, everything is done via GET even operations that change things.
- Resource representations not interconnected, you can't traverse from a list of bookmarks to a single bookmark.

So (ignoring why) how can we make it better, make it more RESTful? Lets start with describing what it is the Web service does.

It allows us to:

- Get a list of all our bookmarks and to filter that list by tag or date or limit by number
- Get the number of bookmarks created on different dates
- Get the last time we updated our bookmarks
- Get a list of all our tags
- Add a bookmark
- Edit a bookmark
- Delete a bookmark
- Rename a tag

From this we can identify two first class objects; bookmarks and tags, so we'll expose both of these as resources at the URL paths of `http://del.icio.us/api/[username]/bookmarks` and `http://del.icio.us/api/[username]/tags` where `[username]` is the username of the users bookmarks we're interested in. We'll still use HTTP Authentication to limit peoples access to bookmarks, but we've added the users username into the URL since otherwise we'd be using HTTP Auth as part of our resource identification instead of just the URL and that'd be unRESTful.

We'll need to define some XML document formats to represent our bookmarks and tags. It doesn't matter exactly what they look like as long as they convery the correct information and we define mimetypes to identify them, but I'll include examples of my versions of them as we go along to make things clearer.

Mimetype	Description
delicious/bookmarks+xml	A list of bookmarks
delicious/bookmark+xml	A bookmark
delicious/bookmarkcount+xml	Count of bookmarks per date
delicious/update+xml	When the bookmarks were last updated
delicious/tags+xml	A list of tags
delicious/tag+xml	A tag

Table of XML document types we'll invent

Getting Bookmarks

In the POX Delicious API bookmarks are accessed by a RPC style request to http://del.icio.us/api/posts/get with a number of optional querystring parameters that influence the returned results. To do a similar thing RESTfully, we'll define a similar resource at `http://del.icio.us/api/[username]/bookmarks/` that returns a list of bookmarks. We'll also define an infinite number of resources at `http://del.icio.us/api/[username]/bookmarks/[hash]` that represent our individual bookmarks where `[hash]` is the Delicious hash used to identify a bookmark.

Get all bookmarks

URL	http://del.icio.us/api/[username]/bookmarks/	
Method	GET	
Querystring	tag=	Filter by tag
	dt=	Filter by date
	start=	The number of the first bookmark to return
	end=	The number of the last bookmark to return
Returns	200 OK & XML (delicious/bookmarks+xml)	
	401 Unauthorized	
	404 Not Found	

The bookmarks XML should include the URLs of each individual bookmark so a program can easily traverse to a single bookmark from the list of bookmarks. An example `delicious/bookmarks+xml` document might look something like:

```
GET http://del.icio.us/api/peej/bookmarks/?start=1&end=2

<?xml version="1.0"?>
<bookmarks start="1" end="2"
     next="http://del.icio.us/api/peej/bookmarks?start=3&amp;end=4">
     <bookmark url="http://www.example.org/one" tags="example,test"
          href="http://del.icio.us/api/peej/bookmarks/a211528fb5108cddaa4b0d3aec
          time="2005-10-21T19:07:30Z">
         Example of a Delicious bookmark
    </bookmark>
     <bookmark url="http://www.example.org/two" tags="example,test"
          href="http://del.icio.us/api/peej/bookmarks/e47d06a59309774edab5681343
          time="2005-10-21T19:34:16Z">
         Another example of a Delicious bookmark
    </bookmark>
</bookmarks>
```

You'll notice the `next` attribute of the bookmarks element, this contains a URL to the next "page" of our returned results. Since a user might have thousands of bookmarks and we don't want to return them all, we page the results and provide `next` and `previous` attributes to allow us to get the the next and previous pages (we have no `previous` attribute in this example since we are on the first page). These URLs use the `start` and `end` querystring parameters to limit the results to a specific page.

Get info on a specific bookmark

URL	http://del.icio.us/api/[username]/bookmarks/[hash]
Method	GET
Returns	200 OK & XML (delicious/bookmark+xml)
	401 Unauthorized
	404 Not Found

Since a bookmark has a number of tags, it'd be RESTful to have the bookmark XML provide links to each tag:

```
GET http://del.icio.us/api/peej/bookmarks/a211528fb5108cddaa4b0d3aeccdbdcf

<?xml version="1.0"?>
<bookmark url="http://www.example.org/one" time="2005-10-21T19:07:30Z">
    <description>
        Example of a Delicious bookmark
    </description>
    <tags count="2">
         <tag name="example" href="http://del.icio.us/api/peej/tags/example"/>
         <tag name="test" href="http://del.icio.us/api/peej/tags/test"/>
    </tags>
</bookmark>
```

Delicious provides two further retrieval "operations" on your list of bookmarks, a count of the number of bookmarks created on each date, and the time the user last updated their bookmarks. We'll implement these as two resources within the bookmarks path that return this information.

Get count of posts per date

URL	http://del.icio.us/api/[username]/bookmarks/count	
Method	GET	
Querystring	tag=	filter by tag
Returns	200 OK & XML (delicious/bookmarkcount+xml)	
	401 Unauthorized	
	404 Not Found	

Get last update time for the user

URL	http://del.icio.us/api/[username]/bookmarks/update
Method	GET

Returns	200 OK & XML (delicious/update+xml)
	401 Unauthorized
	404 Not Found

Modifying Bookmarks

In the original API, you add a new bookmark by issuing a RPC style GET request to an add URL. Not very RESTful. Modifying is done by using the same add URL, and deleting is done in the same way but via a delete URL. We'll be more RESTful by using the correct HTTP methods on existing bookmark resources to add, modify and delete bookmarks.

Add a bookmark to Delicious

To create a bookmark we'll POST an XML representation of a bookmark to our bookmarks URL.

URL	http://del.icio.us/api/[username]/bookmarks/
Method	POST
Request Body	XML (delicious/bookmark+xml)
Returns	201 Created & Location
	401 Unauthorized
	415 Unsupported Media Type

Since to create a new bookmark Delicious doesn't need all of the information that we've added to our bookmark XML format, we'll allow some of the XML attributes and nodes to be optional:

```
POST http://del.icio.us/api/peej/bookmarks/

<?xml version="1.0"?>
<bookmark url="http://www.example.org/one" time="2005-10-21T19:07:30Z">
    <description>
        Example of a Delicious bookmark
    </description>
    <tags>
        <tag name="example"/>
        <tag name="test"/>
    </tags>
</bookmark>
```

The `415 Unsupported Media Type` error is returned if an incorrect XML document is sent.

On success, a `201 Created` response is returned along with a location header containing the URL of the newly created bookmark resource.

Modify a bookmark

To modify an existing bookmark, we'll PUT an XML representation of a bookmark to the URL of our bookmark.

URL	http://del.icio.us/api/[username]/bookmarks/[hash]
Method	PUT
Request Body	XML (delicious/bookmark+xml)
Returns	201 Created & Location
	401 Unauthorized
	404 Not Found
	415 Unsupported Media Type

Since we don't want people to create new bookmarks using PUT, only update existing bookmarks, we'll return a `404 Not Found` error if the resource doesn't already exist.

Delete a bookmark from Delicious

To delete a bookmark we'll use the HTTP DELETE method.

URL	http://del.icio.us/api/[username]/bookmarks/[hash]
Method	DELETE
Returns	204 No Content
	401 Unauthorized
	404 Not Found

On success, we'll return a `204 No Content` code since we won't return an response body.

Tags

The last part of the Delicious API deals with the users tags, it allows us to get tags and change the names of them.

Get all tags

URL	http://del.icio.us/api/[username]/tags/	
Method	GET	
Querystring	start=	The number of the first tag to return
	end=	The number of the last tag to return
Returns	200 OK & XML (delicious/tags+xml)	
	401 Unauthorized	
	404 Not Found	

Like the bookmarks XML, the tags XML should include the URLs of each individual tag:

```
GET http://del.icio.us/api/peej/tags/?start=1&end=2

<?xml version="1.0"?>
<tags start="1" end="2" next="http://del.icio.us/api/peej/tags?start=3&amp;end=
    <tag name="example" count="2" href="http://del.icio.us/api/peej/tags/exampl
    <tag name="test" count="2" href="http://del.icio.us/api/peej/tags/test"/>
</tags>
```

Get info on a specific tag

URL	http://del.icio.us/api/[username]/tags/[name]
Method	GET
Querystring	start= The number of the first bookmark for this tag to return
	end= The number of the last bookmark for this tag to return
Returns	200 OK & XML (delicious/tag+xml)
	401 Unauthorized
	404 Not Found

Since a tag has a number of bookmarks, it'd be RESTful to have the tag XML provide links to each bookmark:

```
GET http://del.icio.us/api/peej/tags/example?start=1&end=2

<?xml version="1.0"?>
<tag name="example">
    <bookmarks start="1" end="2"
         next="http://del.icio.us/api/peej/tags/example?start=3&amp;end=4">
        <bookmark url="http://www.example.org/one"
         href="http://del.icio.us/api/peej/bookmarks/a211528fb5108cddaa4b0d3aec
        <bookmark url="http://www.example.org/two"
         href="http://del.icio.us/api/peej/bookmarks/e47d06a59309774edab568134
    </tags>
</tag>
```

Rename a tag

To rename a tag we'll use the PUT method of a tag to upload a new representation of the tag. The problem with this is that the name of the tag is part of the tags URL, so changing its name also changes its URL, so we return a location header of the resources new URL.

URL	http://del.icio.us/api/[username]/tags/[name]
Method	PUT
Request Body	XML (delicious/tag+xml)
Returns	204 No Content & Location
	401 Unauthorized
	404 Not Found

Finishing the API

Okay, so now that we've defined our new RESTful API, how do we use it? Well, REST says to "use hypertext as the engine of application-state transitions", meaning, given a starting URL, we should be able to fetch a resource and use the URLs within it to navigate to the other resources that we want. We shouldn't have to guess or construct any URLs ourselves since URLs are transparent and don't have to follow any particular formula or pattern (and since software can't easily spot patterns and construct URLs by itself).

So we're missing a homepage for our API, let's create one. This will be the entry point for our RESTful API, by giving this URL to a program that knows about our Delicious XML data formats, it can find all the data it requires without having to know anything about the rest of the API.

URL	http://del.icio.us/api/	
Method	GET	
Querystring	start=	The number of the first user to return
	end=	The number of the last user to return
Returns	200 OK	

```
GET http://del.icio.us/api/?start=1&end=2

<?xml version="1.0"?>
<users start="1" end="2" next="http://del.icio.us/api/?start=3&amp;end=4">
    <user name="Peej"
        href="http://del.icio.us/api/peej/"
        bookmarks="http://del.icio.us/api/peej/bookmarks/"
        tags="http://del.icio.us/api/peej/tags"/>
    <user name="Joshua"
        href="http://del.icio.us/api/joshua/"
        bookmarks="http://del.icio.us/api/joshua/bookmarks/"
        tags="http://del.icio.us/api/joshua/tags"/>
</users>
```

Each user then needs to be defined as a resource, we failed to spot the user as a first class object when we started, so we'll add it now.

URL	http://del.icio.us/api/[username]/	
Method	GET	
Querystring	start=	The number of the first bookmark to return
	end=	The number of the last bookmark to return
Returns	200 OK	
	401 Unauthorized	
	404 Not Found	

```
GET http://del.icio.us/api/peej/?start=1&end=2

<?xml version="1.0"?>
<user email="paul@peej.co.uk" name="Paul James" href="http://www.peej.co.uk/">
    <bookmarks href="http://del.icio.us/api/peej/bookmarks/" start="1" end="2"
        next="http://del.icio.us/api/peej/?start=3&amp;end=4">
        <bookmark url="http://www.example.org/one"
         href="http://del.icio.us/api/peej/bookmarks/a211528fb5108cddaa4b0d3aec
        <bookmark url="http://www.example.org/two"
         href="http://del.icio.us/api/peej/bookmarks/e47d06a59309774edab568134
    </bookmarks>
    <tags href="http://del.icio.us/api/peej/tags/">
        <tag name="example" count="2" href="http://del.icio.us/api/peej/tags/e
        <tag name="test" count="2" href="http://del.icio.us/api/peej/tags/test
    </tags>
</user>
```

The user resource could have a PUT method to allow updating of users settings and a DELETE method to remove the user and all their bookmarks and tags.

An example of using the API

Let's finish this example RESTful API with an example of a client program using it.

I've been messing around with Delicious and have created a number of bookmarks for testing purposes. I don't want them anymore, so our pretend program is going to delete them for me. I tagged them all up with the tag "test" so I'd be able to find them later, I'll use this tag to track them down and remove them.

1. **GET http://del.icio.us/api/**

   First, we start like every client, by fetching the APIs homepage. We parse the response XML and find the tags URL of the user with the name "Peej". This gives us http://del.icio.us/api/peej/tags/.

2. **GET http://del.icio.us/api/peej/tags/**

   So we get the resource that represents my tags. We parse the XML and see if the tag "test" is mentioned. If it isn't, then we GET the URL in the "next" attribute and try again. Eventually we'll get the URL http://del.icio.us/api/peej/tags/test

3. **GET http://del.icio.us/api/peej/tags/test**

   Now we've got the details for the tag "test", so we parse the XML and get a list of bookmarks. If there is a "next" attribute, we'll follow that to get the next page of bookmarks. Eventually we'll have a list of all bookmarks tagged with "test".

4. **DELETE http://del.icio.us/api/peej/bookmarks/[hash]**

   Finally we can walk through our list of bookmarks and call the DELETE HTTP method on each of their URLs.

Now this is quite involved and we could take shortcuts by assuming that we know the URL of the tag we want is `http://del.icio.us/api/peej/tags/test`. What it does show is that with a RESTful API it is possible for a piece of software to use the Web in a similar way that a human does to perform a task, there's no need for description languages like WSDL, REST is self descriptive.

Conclusion

Most "REST" APIs around today are not very RESTful, most are as RESTful as equivalent SOAP or XML-RPC services. If you are using HTTP you are being RESTful to some degree since HTTP is a REST protocol, but to take full advantage of the platform, APIs should embrace RESTful practices as much as possible.

I hope this article has been intersting, and I hope your next Web software product will embrace REST principles for its machine and human interfaces.

More Articles

- HTTP Caching - The lost art of saving bandwidth and CPU time. (May 7, 2006)
- HTTP Auth with HTML Forms - How to use HTTP authentication with HTML forms. (Feb 3, 2006)
- A RESTful Web Service, an Example - An example of a RESTful API for Delicious. (Oct 30, 2005)
- Content Negotiation - Resources, representations, and content negotiation. (Jul 23, 2005)
- XML is not a Data Transfer Format - If it's a message don't send a document. (Jun 18, 2005)
- 3 Tiered REST Architecture - How REST can fit easily and neatly into Web app development. (Mar 1, 2005)
- What are Web Services? - Do they actually exist? (Feb 2, 2005)
- REST Data Format Confusion - The lack of a standard data format is a feature not a bug. (Jan 15, 2005)
- XMLHttpRequest, REST and the Rich User Experience - How to use the Javascipt XMLHttpRequest object to create a richer user experience. (Oct 23, 2004)
- REST - What is Representational State Transfer and how will it save the web? (May 2, 2002)
- XHTML - What is it, why is it, when is it, and how to use it. (Feb 14, 2002)
- Cross Site Scripting - What is it and why does it affect you. (Sep 27, 2001)
- SQL Injection - Why checking user input from web forms is important. (Aug 1, 2001)