Web Service Design with AtomPub

Ben Ramsey ▪ Code Works

Atom & AtomPub?

What's the difference?

Atom:
RFC 4287

Atom is an XML language.

AtomPub:
RFC 5023

AtomPub is a publishing protocol.

What about RSS?

Content types

- RSS allows only for plain text and escaped HTML content types
- Atom provides for plain text, escaped HTML, XHTML, XML, and Base64-encoded binary data

Internationalization

- RSS may have a language set for a feed, but doesn't have a way to indicate language for items in the feed
- Atom uses the xml:lang attribute to specify language on a per-element basis
- Atom uses IRIs, which allow the usage of characters in identifiers outside of ASCII

Modularity

- RSS vocabulary elements aren't reusable in other XML vocabularies
- Atom was designed to allow its vocabulary to be mixed with other XML vocabularies
- Namespaces! Atom has one; RSS does not

Other things

- RSS has no schema; Atom has a RelaxNG schema
- RSS defines no mechanism for handling relative URIs; Atom uses xml:base
- Various implementations in RSS leads to interoperability problems
- No standard protocol for publishing

Atom was created to solve the RSS problems.

Atom profile

- Atom Syndication Format
- An XML-based web content and metadata syndication format
- Defined by IETF **RFC 4287**
- Fixes the “problems” of RSS
- XML namespace: **http://www.w3.org/2005/Atom**

AtomPub profile

- Atom Publishing Protocol
- A protocol for publishing and editing web resources using HTTP and XML
- Defined by IETF **RFC 5023**
- Uses Atom as it's XML syntax
- XML namespace: **http://www.w3.org/2007/app**

AtomPub basics

- Each resource has a unique identifier
- The resources are well-connected
- Resources share the same interface
- There is no state; requests are atomic
- Follows a resource-oriented architecture

Some terms...

- Entry
- Feed/Collection
- Category Document
- Service Document

Content types...

- Entry:
  application/atom+xml;type=entry
- Feed/Collection:
  application/atom+xml
- Category Document:
  application/atomcat+xml
- Service Document:
  application/atomsvc+xml

Designing an AtomPub service

What will our service do?

- Expose users
- Expose content
- Allow users to manipulate content

1. Define our URIs

Users

- /user
- /user/ramsey

Content

- /content
- /content/1234

2. Define the relationships

- user ⇛ content:
  one to many

- content ⇛ user:
  one to one

3. Define the interface

Methods
GET PUT POST DELETE

Cut & Paste
Copy Paste Over Paste After Cut

- Retrieve user collection:
  GET /user

- Create a new user:
  POST /user

- Modify an existing user:
  PUT /user/ramsey

- Delete a user:
  DELETE /user/ramsey

- Retrieve content:
  GET /content

- Create new content:
  POST /content

- Modify content:
  PUT /content/1234

- Remove content:
  DELETE /content/1234

- Service discovery:
  GET /

- Retrieve content for a particular user:
  GET /user/ramsey/content

Let’s see it in action

```
GET / HTTP/1.1
Host: atom.example.org
```

```
HTTP/1.x 200 OK
Date: Mon, 21 Sep 2009 16:33:45 GMT
Content-Type: application/atomsvc+xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<service xmlns="http://www.w3.org/2007/app"
         xmlns:atom="http://www.w3.org/2005/Atom"
         xml:base="http://atom.example.org/">
 <workspace>
  <atom:title>Our Content Store</atom:title>
  <collection href="user">
   <atom:title>Users</atom:title>
   <accept>application/atom+xml;type=entry</accept>
   <categories href="cat/user"/>
  </collection>
  <collection href="content">
   <atom:title>Content</atom:title>
   <accept>application/atom+xml;type=entry</accept>
   <categories href="cat/content"/>
  </collection>
 </workspace>
</service>
```

```
GET /user HTTP/1.1
Host: atom.example.org
```

```
HTTP/1.x 200 OK
Date: Mon, 21 Sep 2009 16:34:26 GMT
Content-Type: application/atom+xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app"
      xml:base="http://atom.example.org/">
 <title>Users</title>
 <updated>2009-09-21T05:21:19Z</updated>
 <id>tag:example.org,2009-09:user</id>
 <app:collection href="user">
  <title>Users</title>
  <app:accept>application/atom+xml;type=entry</app:accept>
  <app:categories href="cat/user"/>
 </app:collection>
 <link rel="first" href="user"/>
 <link rel="last" href="user?p=23"/>
 <link rel="next" href="user?p=2"/>
 <entry>
  ...
 </entry>
</feed>
```

Manipulate a user

```
GET /user/ramsey HTTP/1.1
Host: atom.example.org
```

```
HTTP/1.x 200 OK
Date: Mon, 21 Sep 2009 16:34:26 GMT
Content-Type: application/atom+xml;type=entry
```

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xml:base="http://atom.example.org/">
 <title>ramsey</title>
 <author>
  <name>ramsey</name>
 </author>
 <link rel="self" href="user/ramsey"/>
 <link rel="edit" type="application/atom+xml;type=entry"
  href="user/ramsey"/>
 <link rel="related" href="user/ramsey/content"/>
 <id>tag:example.org,2008:user/ramsey</id>
 <updated>2009-09-21T13:45:00Z</updated>
 <published>2008-05-23T16:23:34Z</published>
 <content type="xhtml">
   <div class="vcard">
    <a class="fn">Ben Ramsey</a>
    <span class="tel">123-456-7890</span>
   </div>
 </content>
</entry>
```

Modify the entry locally

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xml:base="http://atom.example.org/">
 <title>ramsey</title>
 <author>
  <name>ramsey</name>
 </author>
 <link rel="self" href="user/ramsey"/>
 <link rel="edit" type="application/atom+xml;type=entry"
  href="user/ramsey"/>
 <link rel="related" href="user/ramsey/content"/>
 <id>tag:example.org,2008:user/ramsey</id>
 <updated>2009-09-22T09:14:58Z</updated>
 <published>2008-05-23T16:23:34Z</published>
 <content type="xhtml">
   <div class="vcard">
    <a class="fn url" href="http://benramsey.com/">Ben Ramsey</a>
    <span class="org">Schematic</span>
    <span class="tel">123-456-7890</span>
   </div>
 </content>
</entry>
```

```
PUT /user/ramsey HTTP/1.1
Host: atom.example.org
Content-Type: application/atom+xml;type=entry

{body here}
```

```
HTTP/1.x 200 OK
Date: Mon, 22 Sep 2009 09:14:59 GMT
Content-Type: application/atom+xml;type=entry
```

Add some content

First, a few notes

- You need authentication!
- Perhaps you need encryption
- You definitely need validation
- And I’m not going to tell you how

That’s out of scope.

:-)

But I'll show you the basics...

```
<?xml version="1.0" encoding="utf-8"?>
<service xmlns="http://www.w3.org/2007/app"
         xmlns:atom="http://www.w3.org/2005/Atom"
         xml:base="http://atom.example.org/">
 <workspace>
  <atom:title>Our Content Store</atom:title>
  <collection href="user">
   <atom:title>Users</atom:title>
   <accept>application/atom+xml;type=entry</accept>
   <categories href="cat/user"/>
  </collection>
  <collection href="content">
   <atom:title>Content</atom:title>
   <accept>application/atom+xml;type=entry</accept>
   <categories href="cat/content"/>
  </collection>
 </workspace>
</service>
```

Get the categories

```
GET /cat/content HTTP/1.1
Host: atom.example.org
```

```
HTTP/1.x 200 OK
Date: Mon, 22 Sep 2009 09:39:26 GMT
Content-Type: application/atomcat+xml
```

```
<?xml version="1.0"?>
<app:categories
  xmlns:app="http://www.w3.org/2007/app"
  xmlns:atom="http://www.w3.org/2005/Atom"
  fixed="yes"
  scheme="http://atom.example.com/cat/content">
 <atom:category term="audio"/>
 <atom:category term="video"/>
 <atom:category term="game"/>
</app:categories>
```

Create the entry document locally

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xml:base="http://atom.example.org/">
 <title>Perfect</title>
 <author>
  <name>Mark Phelps</name>
 </author>
 <id>tag:example.org,2009:content/perfect</id>
 <published>2009-09-22T20:12:08Z</published>
 <category term="audio"
  scheme="http://atom.example.com/cat/content"/>
 <content type="application/mp4">
TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWF
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGl
...
 </content>
</entry>
```

```
POST /content HTTP/1.1
Host: atom.example.org
Content-Type: application/atom+xml;type=entry

{body here}
```

```
HTTP/1.x 202 Accepted
Date: Mon, 22 Sep 2009 09:14:59 GMT

{body contains status indicator}
```

Authentication?

- Atom doesn't specify a preference
- WSSE Username Token
- OAuth
- Basic authentication
- ???

Did I miss anything?

Oh, yeah.

Where's the PHP code?

- Requires lots of parser code; either DOM or XMLReader/XMLWriter
- No good AtomPub library that I like
- Apache Abdera is good, but it's in Java
- I'm porting it to PHP: http://github.com/ramsey/AbderaPHP

Wrapping up…

- You can extend Atom with other XML vocabularies (Dublin Core, etc.)
- XML Digital Signature or XML Encryption may be used, or encrypt as a bag of bits
- Use your preferred auth method
- Use HTTP in a RESTful fashion
- Use DOM or XMLReader/XMLWriter to parse Atom documents

Questions?

- My website is benramsey.com
- @ramsey on Twitter
- Rate this talk at joind.in

- Read the Atom specs at
  tools.ietf.org/html/rfc4287
  tools.ietf.org/html/rfc5023
- My company is Schematic
  schematic.com

Web Service Design with AtomPub