

The Rise of AtomPub in the Enterprise

About me

- ▶ Software Architect, MuleSource
- ▶ Open Source: Mule Galaxy & ESB
CXF/XFire, Abdera, Apache-*
- ▶ Exploring how we *should* be building distributed enterprise architectures





Context

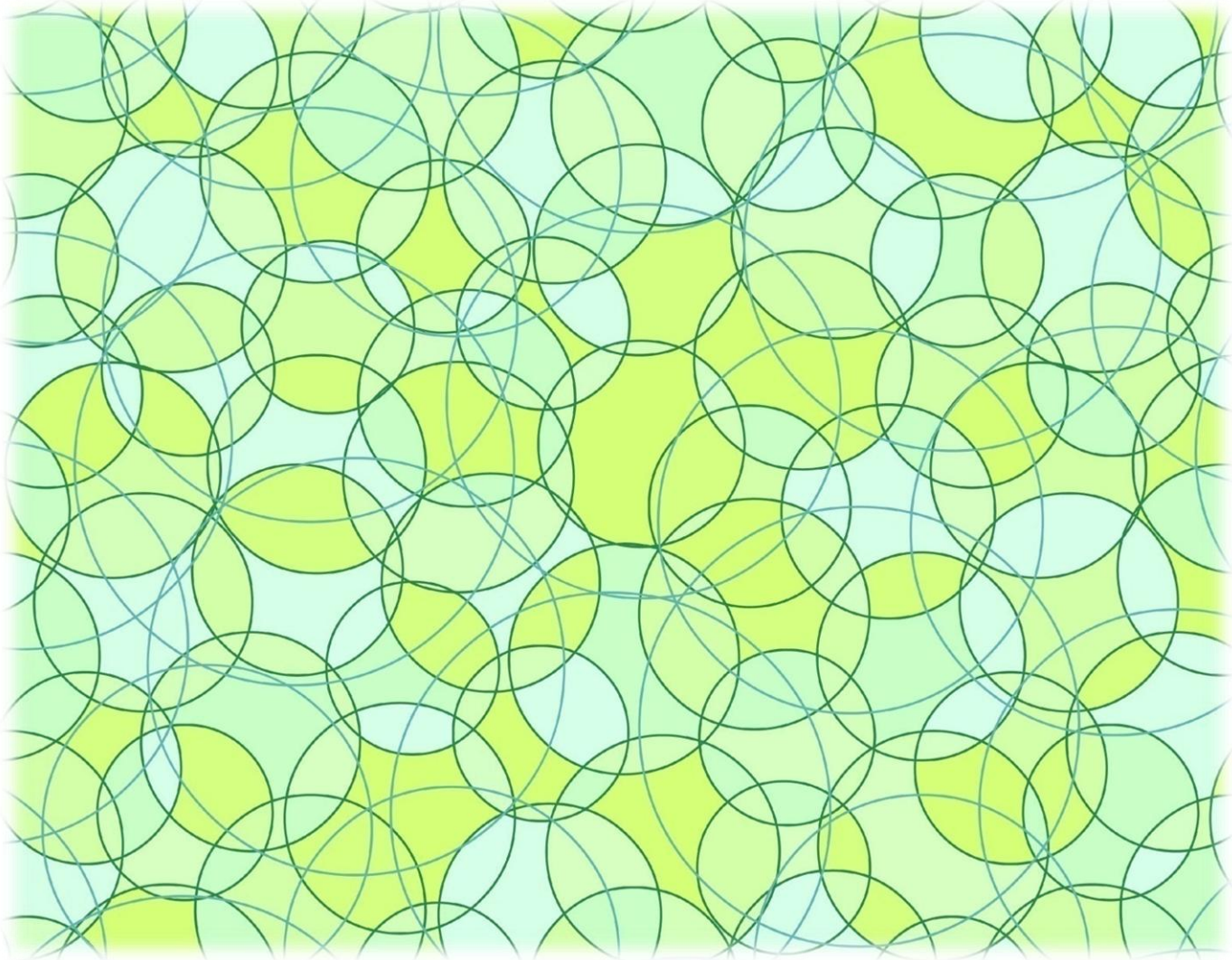
REST

Enterprise
Services

Syndication

Our story starts with the enterprise...

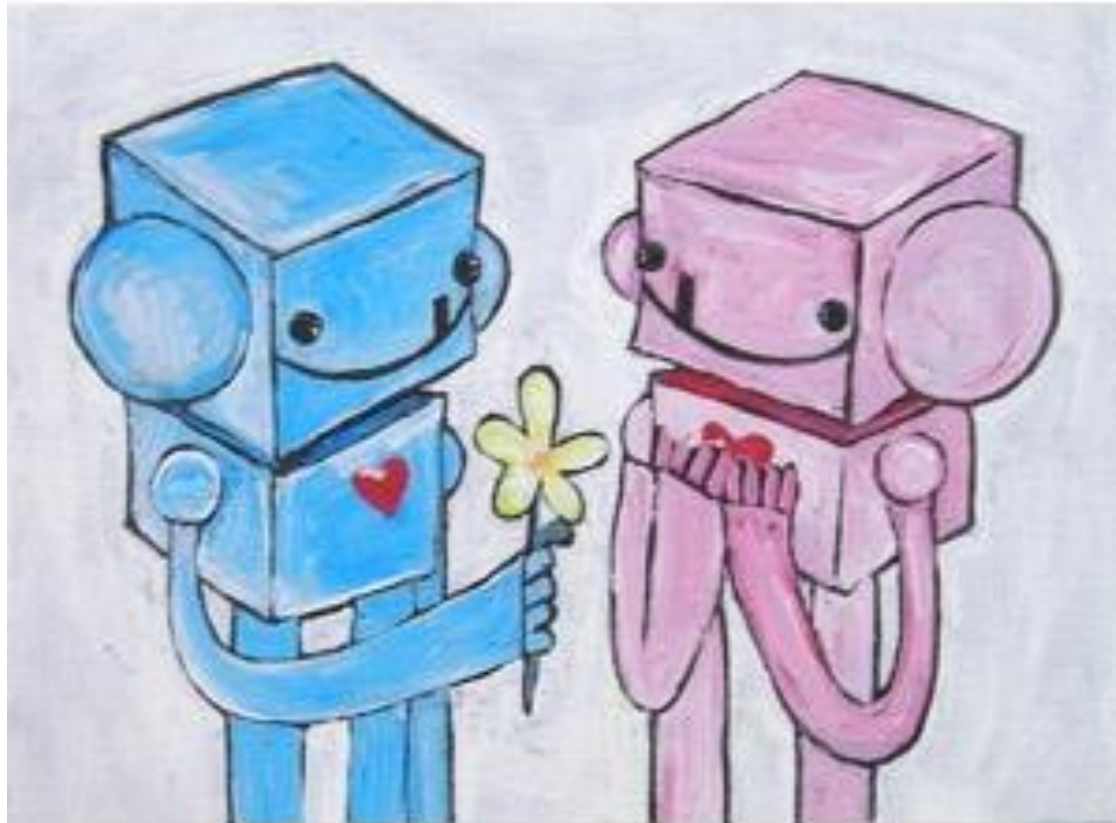
Services



<?xml?>

<soap:Envelope/>

Machine to Machine Interactions



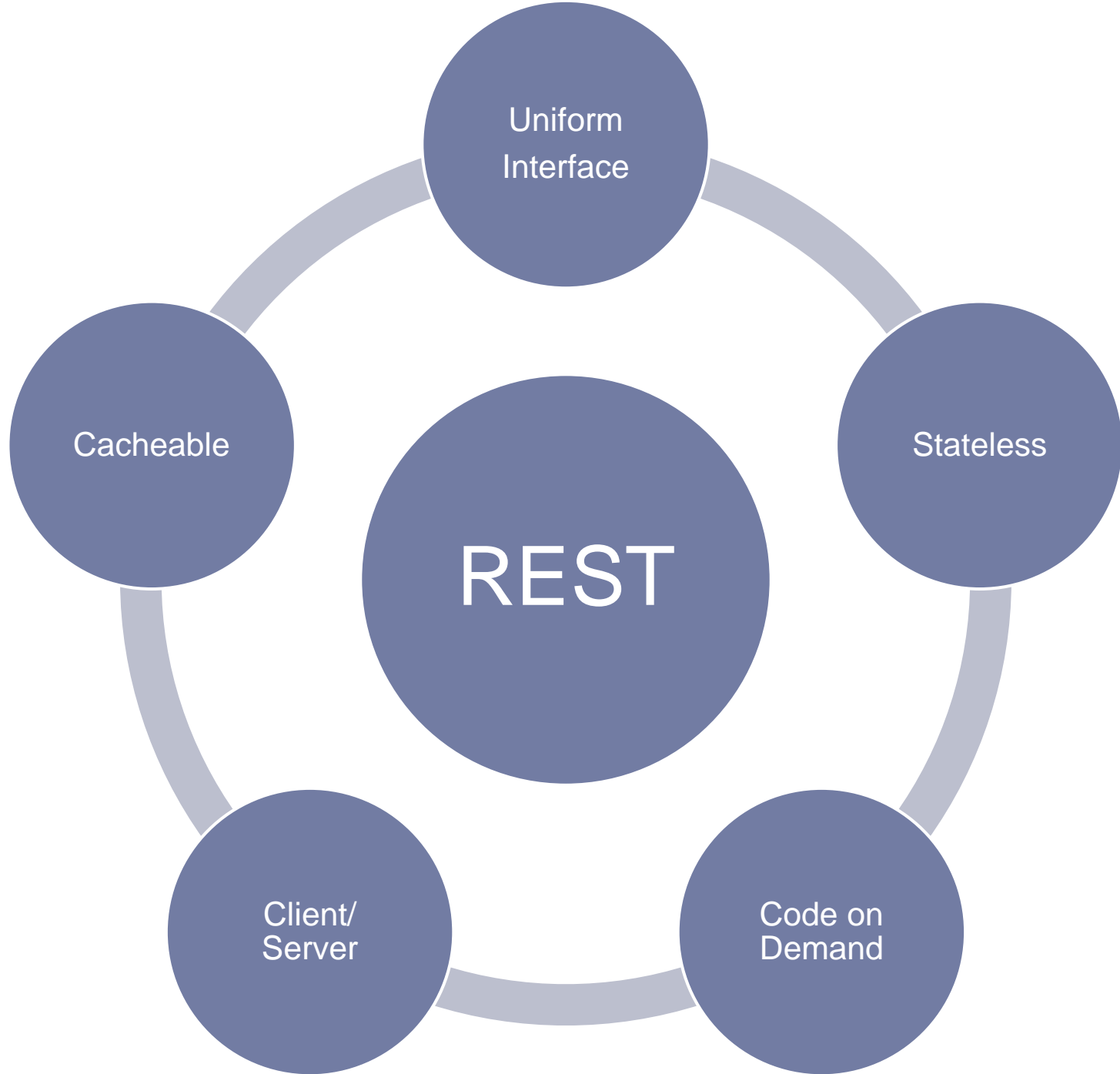
Composable



Meanwhile, back at the web...

“The design rationale behind the Web architecture can be described by an architectural style consisting of the set of constraints applied to elements within the architecture. By examining the impact of each constraint as it is added to the evolving style, we can identify the properties induced by the Web's constraints.”

-- Roy Fielding's Thesis



Resources, resources, resources

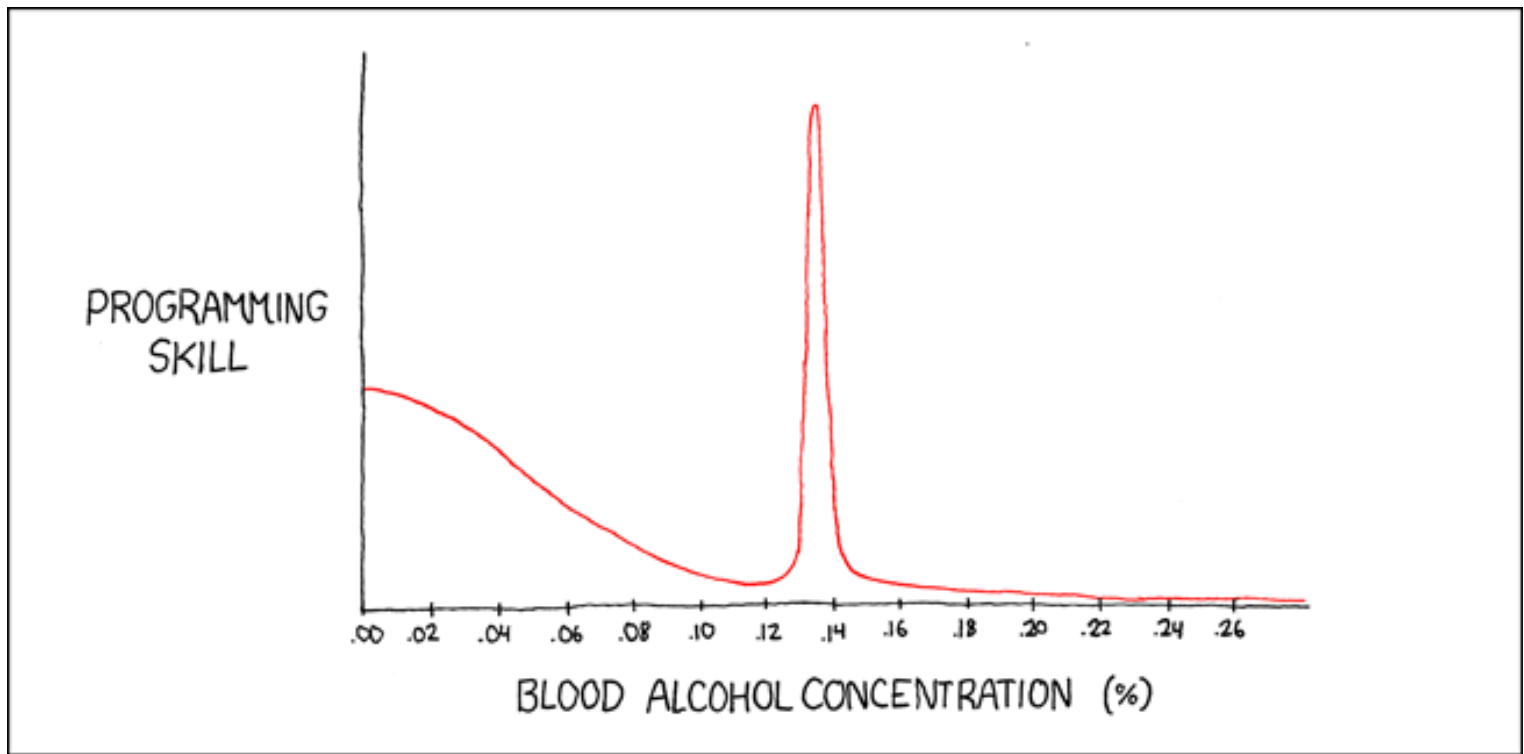
- ▶ **Everything** is a resource
- ▶ Resources are addressable via **URIs**
- ▶ Resources are **self descriptive**
 - ▶ Typically through content types (“application/xml”) and sometimes the resource body (i.e. an XML QName)
- ▶ Resources are ***stateless***
- ▶ Resources are manipulated via **verbs** and the **uniform interface**





(Credit to Paul Downey)

Meanwhile, back at my home computer...



CALLED THE BALLMER PEAK, IT WAS DISCOVERED BY MICROSOFT IN THE LATE 80'S. THE CAUSE IS UNKNOWN, BUT SOMEHOW A B.A.C. BETWEEN 0.129% AND 0.138% CONFERS SUPERHUMAN PROGRAMMING ABILITY.

A stick figure stands on a stage, pointing with a stick to a graph on a screen. The graph on the screen is a smaller version of the one in the top panel, showing the same peak at approximately 0.138% BAC.

HOWEVER, IT'S A DELICATE EFFECT REQUIRING CAREFUL CALIBRATION—YOU CAN'T JUST GIVE A TEAM OF CODERS A YEAR'S SUPPLY OF WHISKEY AND TELL THEM TO GET CRACKING.

A stick figure stands on a stage, speaking to an audience of four seated stick figures. The figure is gesturing with one hand while speaking.

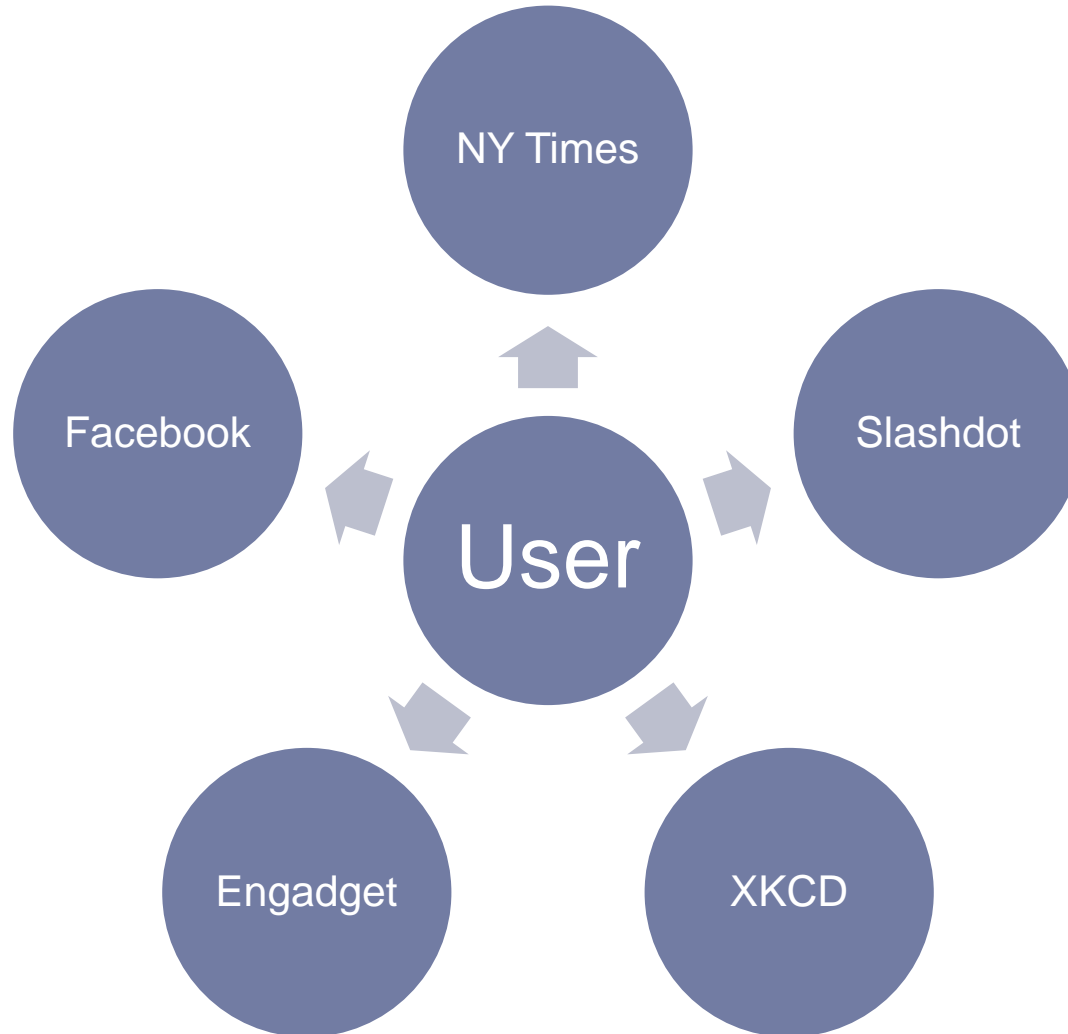
...HAS THAT EVER HAPPENED?

REMEMBER WINDOWS ME?

I KNEW IT!

A stick figure stands on a stage, asking a question to an audience of three seated stick figures. The figure is gesturing with one hand while speaking.

Syndication



RSS was born

- ▶ RDF Site Summary (0.9, 1.0)
- ▶ Rich Site Summary (0.91)
- ▶ Really Simple Syndication (2.0)
- ▶ What a mess...
 - ▶ So many versions
 - ▶ Underspecified
 - ▶ Are fields HTML? Text?
 - ▶ What fields are required?
 - ▶ Not vendor neutral



“At the present time, I would like limit the scope of this wiki to describing a conceptual data model of what constitutes a well formed log entry.”

Sam Ruby, June 16 2003

Minimum Structure

Feed

- ▶ ID
- ▶ Author
- ▶ Link
- ▶ Title
- ▶ Updated
- ▶ *

Entry

- ▶ ID
- ▶ Updated
- ▶ Link
- ▶ Summary/Content
- ▶ *




The Bare Minimum Atom Feed

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Dan's Blog</title>
  <link href="http://netzoid.com/blog/" />
  <updated>2007-11-07T18:30:02Z</updated>
  <author>
    <name>Dan Diephouse</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b91c-0003939e0af6</id>

  <entry>
    <title>Building services with AtomPub</title>
    <link href="http://netzoid.com/blog/atompub_services" />
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2007-11-07T18:30:02Z</updated>
    <content>
      ... (you must have content or a summary)
    </content>
  </entry>

</feed>
```





Atom Publishing Protocol is born...

“An application-level protocol
for publishing and editing Web
Resources”

Atom is more than just blogs

- ▶ **Anything that can fit in a list**
 - ▶ Helps if its associated with times/dates
- ▶ **Entries can represent anything**
 - ▶ Blogs
 - ▶ Calendar
 - ▶ Product directory
 - ▶ Purchase Orders
 - ▶ Videos

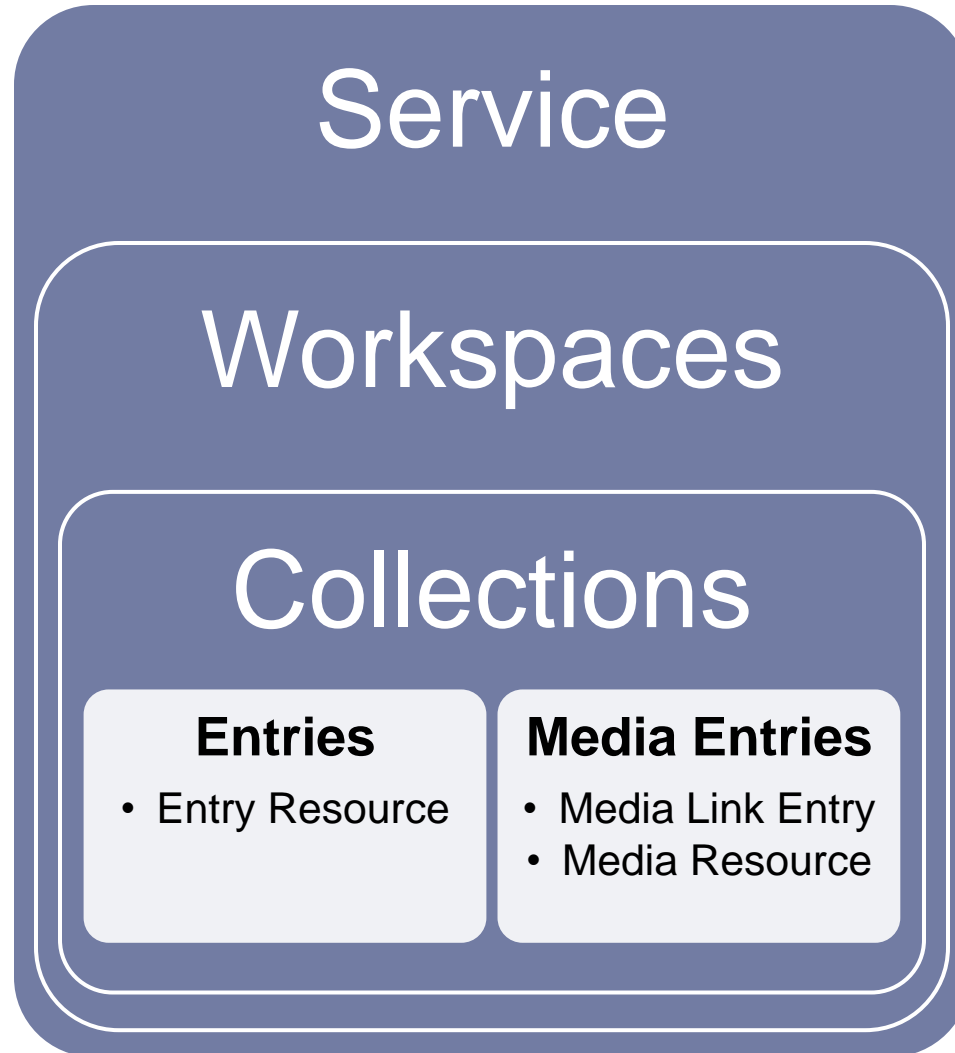


What is the Atom Publishing Protocol?

- ▶ Create, edit, delete *resources* over HTTP
- ▶ Extensible Protocol
 - ▶ Authentication extensions (i.e. WSSE)
 - ▶ Opensearch
 - ▶ GData
- ▶ Properly uses HTTP in a RESTful manner
 - ▶ It is scalable and reliable
- ▶ Builds on Atom model



Atom Publishing Protocol Model



Mechanics of an AtomPub service

AtomPub Resources

Resources

/

/products

/products/abc.atom

/products/abc

Description

Service

Collection


Entry

Alternate
Representation



GET /

```
<?xml version='1.0' encoding='UTF8'?>
<service xmlns="http://www.w3.org/2007/app"
          xmlns:atom="http://www.w3.org/2005/Atom" >
  <workspace>
    <atom:title type="text">
      AtomPub in the Enterprise
    </atom:title>
    <collection href="/products">
      <atom:title type="text">
        Product DB
      </atom:title>
      <accept>application/atom+xml;type=entry</accept>
    </collection>
  </workspace>
</service>
```



GET /

```
<?xml version='1.0' encoding='UTF8'?>
<service xmlns="http://www.w3.org/2007/app"
         xmlns:atom="http://www.w3.org/2005/Atom" >
  <workspace>
    <atom:title type="text">
      AtomPub in the Enterprise
    </atom:title>
    <collection href="/products">
      <atom:title type="text">
        Product DB
      </atom:title>
      <accept>application/atom+xml;type=entry</accept>
    </collection>
  </workspace>
</service>
```



GET /products/Miracle_Widget

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Product Database</title>
  <link href="http://netzoid.com/blog/" />
  <updated>2007-12-13T18:30:02Z</updated>
  <author>
    <name>Acme Enterprises</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>

  <entry>
    <title>Miracle Widget</title>
    <link href="http://host/products/miracle_widget.atom" />
    <link href="http:// host/products/miracle_widget.atom"
      rel="edit" />
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2007-12-13T18:30:02Z</updated>
    <content>
      ... (you must have content or a summary)
    </content>
  </entry>

</feed>
```



A note on <link>

- ▶ Entries contain a set of links
- ▶ Each link has a relationship attribute
 - ▶ i.e. an “edit” link
- ▶ No “rel” attribute means its an alternate representation – i.e. HTML



GET /products/Miracle_Widget.atom

```
<entry>
  <title>Miracle Widget</title>
  <link href="http://host/products/Miracle_Widget"/>
  <link href="http://host/products/Miracle_Widget"
        rel="edit" />
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
  <updated>2007-12-13T18:30:02Z</updated>
  <content>
    ...
  </content>
</entry>
```



Live demo...

Microcontent / Microformats

- ▶ Use XHTML for both data & presentation
- ▶ Microcontent/microformats can be anything to do with your application
 - ▶ Easily access XHTML data with XPath
 - ▶ `//*[class='widget']`
- ▶ Atom servers preserve all metadata not specifically updated
- ▶ Becomes searchable with things like OpenSearch & GData...



OpenSearch

- ▶ An XML format for describing how to query resources
- ▶ Like WSDL for search...



Example

```
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortName>Product Search</ShortName>
  <Description>Search the product database.</Description>
  <Tags>products</Tags>
  <Contact>admin@example.com</Contact>
  <Url
    type="application/atom+xml"
    template="http://example.com/?q={searchTerms}&pw={startPage?}" />
</OpenSearchDescription>
```



OpenSearch

- ▶ Offers a way to tell people how to search your service
- ▶ Several standard parameters:
 - ▶ searchTerms: Search criteria
 - ▶ count: The number of results per page
 - ▶ startPage: The page of results
 - ▶ language: The language of the results
 - ▶ etc..
- ▶ Define your own terms as well



Example

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:opensearch="http://a9.com/-
/spec/opensearch/1.1/">
  <!-- ... -->
  <link
    rel="search"
    href="http://example.com/opensearchdescription.xml"
    type="application/opensearchdescription+xml"
    title="Content Search" />
  <!-- ... -->
</feed>
```



GET /feed?q=Miracle&pw=0

```
<feed xmlns="http://www.w3.org/2005/Atom">
```

```
  ...
  <opensearch:totalResults>35</opensearch:totalResults>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>10</opensearch:itemsPerPage>
  <opensearch:Query role="request" searchTerms="Dan" startPage="1" />

  <link rel="alternate" href="...search.html?pw=1" type="text/html"/>
  <link rel="self" href="...?q=Dan&pw=1" type="application/atom+xml"/>
  <link rel="first" href="...?q=Dan&pw=1" type="application/atom+xml"/>
  <link rel="next" href="...?q=Dan&pw=2" type="application/atom+xml"/>
  <link rel="last" href="...?q=Dan&pw=4" type="application/atom+xml"/>
  <link rel="search" type="application/opensearchdescription+xml"
        href="http://example.com/opensearchdescription.xml"/>

  <entry>
    <title>Miracle Widget</title>
    ...
  </entry>
</feed>
```



AtomPub != <?xml?>

Media Resources and Entries

Media Resources and Media Link Entries

- ▶ What about data which you don't want to necessarily redistribute all the time?
 - ▶ i.e. it's too large
- ▶ What about non-XML data?
 - ▶ Images
- ▶ APP defines:
 - ▶ Media Resource: your data
 - ▶ Media Link Entry: An atom entry which describes your data



Media Resources

Resources

/pics

/pics/pic.png

/pics/pic.atom

Description

Collection

Media
Resource

Media Link
Entry



Service

```
<service xmlns="http://www.w3.org/2007/app"
  xmlns:atom="http://www.w3.org/2005/Atom">
  <workspace>
    <atom:title>Dan's Website</atom:title>

    <collection
      href="http://netzoid.com/blog/feed">
      <atom:title>Dan's Blog</atom:title>
    </collection>

    <collection
      href="http://netzoid.com/pics">
      <atom:title>Dan's Pictures</atom:title>
      <accept>image/png</accept>
      <accept>image/jpeg</accept>
      <accept>image/gif</accept>
    </collection>

  </workspace>
</service>
```



POST /pics/ (request)

Content-Type: image/png

Slug: Dan Rambles

... binary data ...



POST /pics/ (response)

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>Dan rambles</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efe6b</id>
  <updated>2007-11-07T17:26:43Z</updated>
  <author>
    <name>Dan Diephouse</name>
  </author>
  <summary type="text" />
  <content type="image/png"
    src="http://netzoid.com/pics/dan_rambles.png" />
  <link rel="edit-media"
    href="http://netzoid.com/pics/dan_rambles.png" />
  <link rel="edit"
    href="http://netzoid.com/pics/dan_rambles.atom" />
</entry>
```



The Media Link Entry

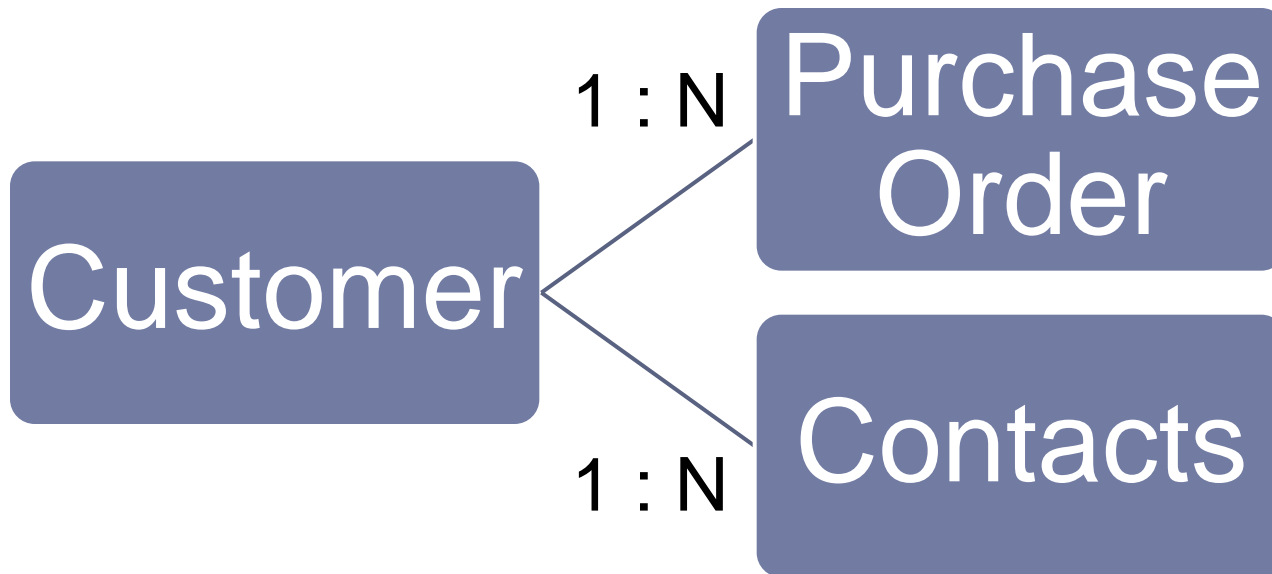
- ▶ You can add summary later or have it auto-generated from the content.
- ▶ Allows you to store, browse, updated, delete *anything!*
 - ▶ Pictures
 - ▶ XML documents
 - ▶ Jars
 - ▶ Videos
 - ▶ Messages...



More ways to use AtomPub

Hierarchies

- ▶ How do I model collections of collections/trees/hierarchies?
- ▶ Example:



Hierarchical data with Atom

```
<entry>
  <title>Customer: Acme Inc.</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-xxxxxxxxxxxx</id>
  <updated>2007-11-07T18:30:02Z</updated>
  <summary>Acme Inc.</summary>
  <collection
    href="http://example.com/customers/acme/purchaseOrders">
    <atom:title>Purchase Orders</atom:title>
  </collection>
</entry>
```



Hierarchical data with Atom

```
<entry>
  <title>Customer: Acme Inc.</title>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-xxxxxxxxxxxx</id>
  <updated>2007-11-07T18:30:02Z</updated>
  <summary>Acme Inc.</summary>
  <collection
    href="http://example.com/customers/acme/purchaseOrders"
    rel="purchase-orders">
    <atom:title>Purchase Orders</atom:title>
  </collection>
  <collection
    href="http://example.com/customers/acme/contacts"
    rel="contacts">
    <atom:title>Contacts</atom:title>
  </collection>
</entry>
```



Thoughts on Hierarchies

- ▶ It works!
- ▶ But is ugly: *Clients do not inherently understand hierarchies*
- ▶ Multiple collections require “rel” attribute, which makes the relationships even less clear



Eventing

- ▶ Publish and consume entries which map to events
- ▶ Application level events
 - ▶ Exceptions/fault monitoring
- ▶ Business level events
 - ▶ A expense over \$1000 was registered
- ▶ Use query parameters to narrow down the criteria
- ▶ Works with any client which understands Atom
- ▶ Powerful combination with opensearch





Why oh why?

Universal

- ▶ Atom is widely understood
- ▶ Provides *ubiquitous elements which have meaning across all contexts*
 - ▶ Summary/Content
 - ▶ Updated date
 - ▶ ID
 - ▶ Links



The Composable Web

- ▶ AtomPub + OpenSearch + XHTML
 - ▶ Create/Edit/Delete
 - ▶ Search & Paging
 - ▶ Presentation
 - ▶ Security
- ▶ Clients do not need to understand your specific application to interact with it



Leverage HTTP

- ▶ AtomPub (nearly) guarantees you'll follow RESTful best practices and have a scalable service
 - ▶ Uniform Interface
 - ▶ ETags
 - ▶ Caching
 - ▶ Reliability
- ▶ Avoid writing your own protocol



Existing Infrastructure

- ▶ Many Atom/AtomPub libraries/frameworks are popping up
 - ▶ Abdera (Java)
 - ▶ Propono (Java)
 - ▶ Amplee (Python)
- ▶ You don't need a framework though: Just use HTTP
 - ▶ wget
 - ▶ commons httpclient
 - ▶ etc





Why not?

Why not AtomPub?

- ▶ More appropriately when not
- ▶ Data is not time indexed
- ▶ Universality does not yield any benefits
- ▶ Batch
- ▶ Performance
- ▶ Messaging may be a more appropriate model
- ▶ Hierarchy kinda sucks
- ▶ Transactions



The one true protocol?

- ▶ No, but...
- ▶ AtomPub can be applied to a wide variety of business applications
- ▶ Universality can be powerful
- ▶ Hypertext model is powerful
- ▶ AtomPub is a great example of how to build a RESTful protocol



Questions?

- ▶ **Blog:** <http://netzoid.com/blog>
- ▶ **Email:** dan@mulesource.com
- ▶ **Some stuff I've been working on:**
 - ▶ <http://incubator.apache.org/abdera>
 - ▶ <http://www.mulesource.org/display/ABDERA>
 - ▶ <http://www.mulesource.org/display/GALAXY>



What is GData?

“simple standard protocol for reading and writing data on the web”



What does that mean?

- ▶ Standard way to query feeds
- ▶ Specifies optimistic concurrency model
- ▶ Way to authenticate users
- ▶ Common elements for Google services
- ▶ A way to do batch operations
- ▶ All built on AtomPub
- ▶ Used for all Google's APIs



Batch

- ▶ APP doesn't specify a way to do batch operations
- ▶ GData supplies one way, but it has received a cold reception as a general purpose way to do batch things
- ▶ Some things to think about:
 - ▶ How do you deal with errors?
 - ▶ Does your batch method lose the benefits of the uniform interface?



GData Batch

```
<feed>
  <entry>
    <batch:operation type="insert" />
    ... what to insert ...
  </entry>
  <entry>
    <batch:operation type="update" />
    ... what to update ...
  </entry>
  <entry>
    <batch:operation type="delete" />
    ... what to delete ...
  </entry>
  <entry>
    <batch:operation type="query" />
    ... what to query ...
  </entry>
</feed>
```



GData Batch response

```
<feed>
  <entry>
    ...
    <batch:operation type="insert"/>
    <batch:id>itemB</batch:id>
    <batch:status code="201" reason="Created"/>
  </entry>
</feed>
```



Thoughts on GData

- ▶ Weakness in APP?
- ▶ Or strength because its extensible?



Security

- ▶ **Goals?**
 - ▶ Privacy, Integrity, authentication, authorization
- ▶ **SSL**
- ▶ **HTTP Auth**
- ▶ **WSSE**
- ▶ **Google Auth**
- ▶ **XML Signature & Encryption**

