



# Distribution and Publication With Atom Web Services

Ben Ramsey ■ ZendCon ■ 16 Sep 2008

Hi, I'm **Ben Ramsey**.

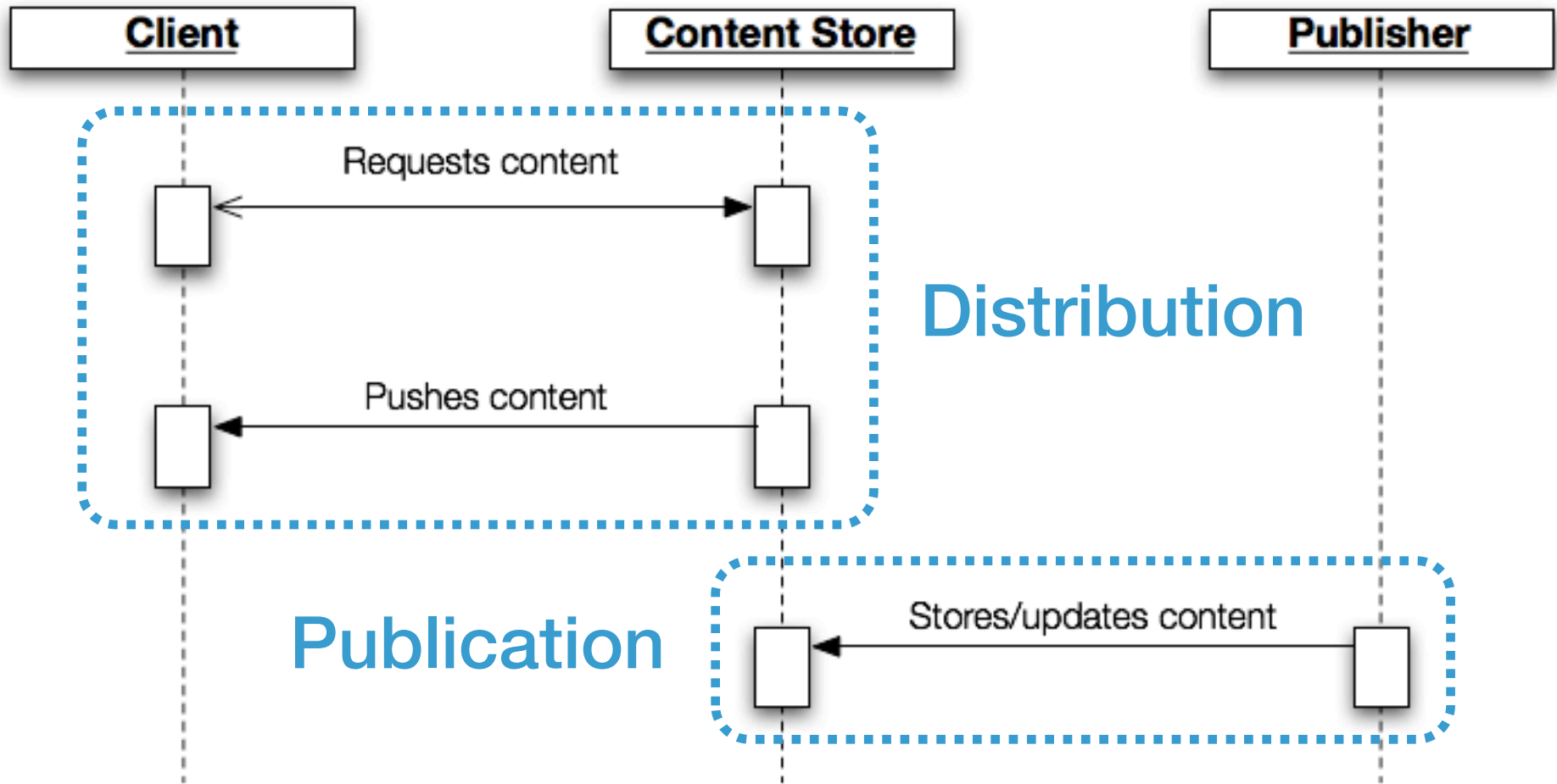
- Software Architect
- Organizer of Atlanta PHP user group
- Co-author of *php|architect's Zend PHP 5 Certification Study Guide*
- HTTP pedant and advocate of RESTful web applications

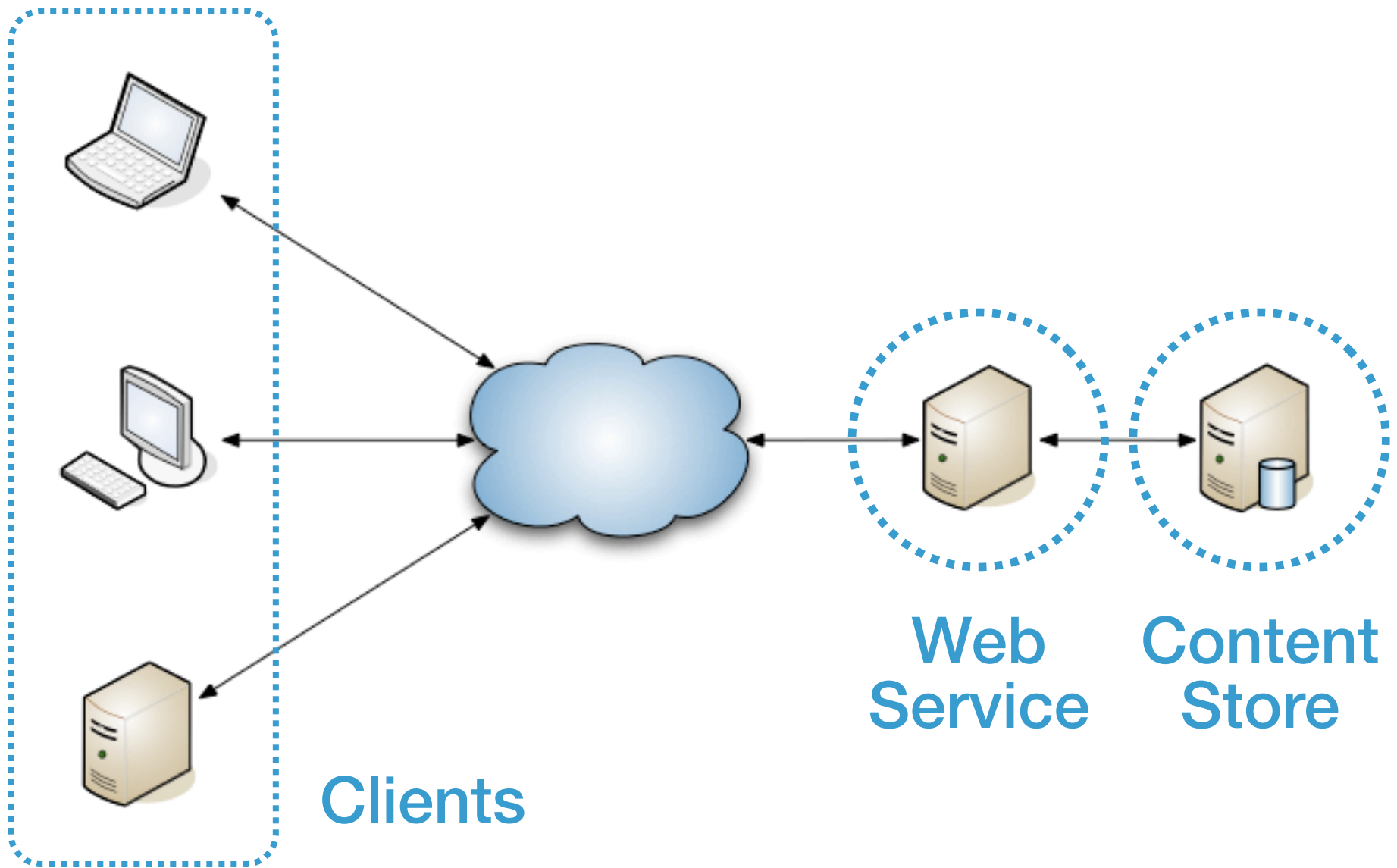
# Distribution

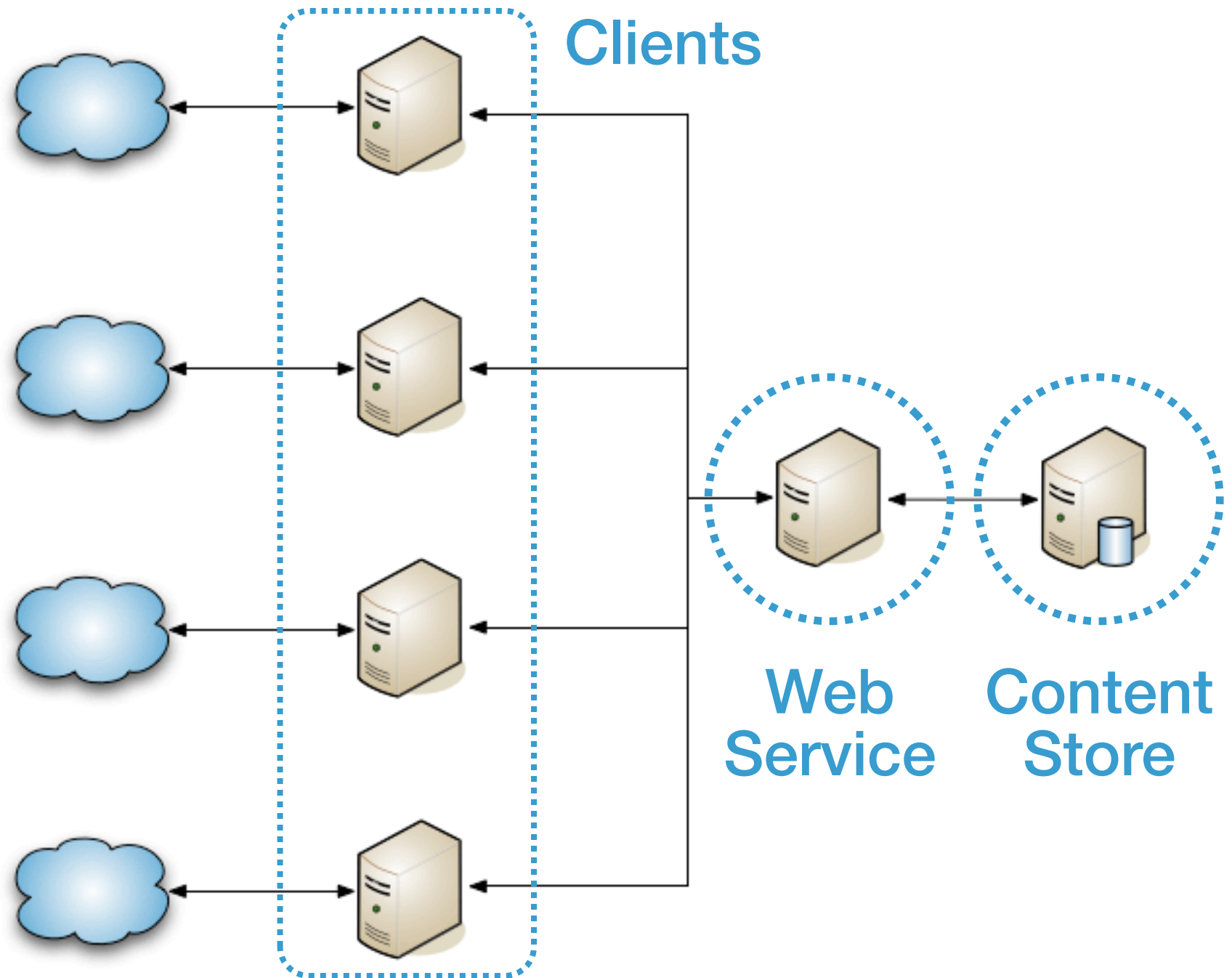
To deliver or pass out something

# Publication

To issue something for distribution







Over a network, we'll use HTTP  
for **distribution** and **publication**.

- Constrained interface
- Client-server
- Stateless
- Cacheable
- Layered

HTTP is **RESTful**.

What is **REST**?

“REST strictly refers to a collection of network architecture principles which outline how resources are defined and addressed.”

— Wikipedia

# What is REST?

- Representational State Transfer
- Not an architecture or a standard for developing web services
- Not a particular format or pattern
- It is a set of design principles

# Principles of REST

- Application state and functionality abstracted into resources
- Resources are uniquely addressable
- Resources share a uniform interface for transfer of state:
  - Constrained set of operations
  - Constrained set of content types

# Principles of REST

- A protocol that is:
  - Client-server
  - Stateless
  - Cacheable
  - Layered

A **resource-oriented** architecture is RESTful.

# Resource-oriented architecture

- Resources
- Their names (addresses)
- Their representations
- The links between them

“A resource can be anything that has identity. Familiar examples include an electronic document, an image, a service (e.g., ‘today’s weather report for Los Angeles’), and a collection of other resources. Not all resources are network ‘retrievable’; e.g., human beings, corporations, and bound books in a library can also be considered resources.”

— RFC 2396

# ROA basics

- Each resource knows how to do three simple things:
  1. Represent itself to the client
  2. Transition from one state to the next
  3. Destroy itself
- Additionally, the ROA provides a means to create a resource

# ROA properties

- Resources are addressable
- Resources have no state
- Resources are connected
- Resources share the same interface

This is REST in a nutshell.



What is **Atom**?

“The name Atom applies to a pair of related standards. The Atom Syndication Format is an XML language used for web feeds, while the Atom Publishing Protocol (short AtomPub or APP) is a simple HTTP-based protocol for creating and updating web resources.”

— Wikipedia

Atom is both a **format** and a **protocol**.

# The Atom format

- An XML-based web content and metadata syndication format
- Defined by RFC 4287
- Developed as an open alternative to the “frozen” RSS 2.0 format
- XML namespace:  
<http://www.w3.org/2005/Atom>

# The Atom Publishing Protocol

- An application-level protocol for publishing and editing web resources using HTTP and XML
- Defined by RFC 5023
- Also called “AtomPub” or “APP”
- XML namespace:  
<http://www.w3.org/2007/app>

# A common format

- Widespread adoption means increased interoperability between disparate systems
- Applications now have a common language and publishing protocol, leveraging HTTP as a vehicle
- Supported by a diverse and wide-reaching community through libraries for consuming and creating Atom services

# Principles of Atom

- Resource-oriented
- Discovery of resource relationships
- Categorization of resources
- Provides service auto-discovery
- Support for all content types
- Content can be published

# Atom is resource-oriented

- Think of an Atom web service as a catalog of resources that you want to expose
- Each piece of content in your system is a resource
- Each resource has a unique identifier
- The resources have no state
- They are cacheable
- They are connected
- They share the same interface: HTTP verbs

# Atom exposes relationships

- Atom documents are connected through links
- Links can have defined relationships:
  - alternate
  - related
  - self
  - enclosure
  - via

# Atom categorizes resources

- Atom documents may belong to multiple categories
- Categories belong to namespaces
- Categories are very flexible; up to implementor to decide how to use them
- APP category documents expose categories, optionally defining limitations

# Atom provides auto-discovery

- APP service documents expose the collections of an Atom web service
- Service documents provide flexible workspaces for the implementor to define
- May define endpoints for publication of resources, acceptable content types, and categories

# Atom delivers content

- Collection documents (feeds) deliver lists of resources
- Collection documents can be paginated to allow paging through content
- Entry documents deliver individual resources, describing relationships with other resources and providing links to follow to more content

# Atom publishes content

- Individual entries may be published to an Atom web service to add or update content
- Atom allows for the deletion or removal of content entries
- Atom is flexible enough to allow for whole collection documents to be published, perhaps creating or updating many resources at one time

# Distribution with Atom

# Atom terminology

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

# Entry

- Represents an individual piece of content (post, article, page, image, video, document, etc.)
- Entries can be related to other entries or feeds
- This resource represents a specific resource (i.e. <http://example.org/posts/1234>)
- Content type:  
`application/atom+xml;type=entry`

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
  <title>Atom Web Services</title>
  <author>
    <name>Ben Ramsey</name>
  </author>
  <link rel="self" href="http://example.org/2008/06/article.atom"/>
  <link rel="edit" type="application/atom+xml;type=entry"
        href="http://example.org/2008/06/article.atom"/>
  <link rel="alternate" type="text/html"
        href="http://example.org/2008/06/article.html"/>
  <id>tag:example.org,2008-06-03:article</id>
  <updated>2008-06-03T13:45:00Z</updated>
  <published>2008-06-03T13:45:00Z</published>
  <summary>Some text.</summary>
</entry>
```

# Feed

- Represents a collection of Atom entries
- Feeds can be related to other feeds or entries
- This resource represents a specific collection of resources (i.e. <http://example.org/posts>)
- Atom collections are feeds
- Content type:  
`application/atom+xml;type=feed`

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
  <title>Archives</title>
  <updated>2008-06-03T05:21:19Z</updated>
  <id>tag:example.org,2008-06:archives</id>
  <app:collection href="http://example.org/archives">
    <title>Archives</title>
    <app:accept>application/atom+xml;type=entry</app:accept>
    <app:categories href="https://example.org/categories"/>
  </collection>
  <link rel="self" type="application/atom+xml;type=feed"
        href="http://example.org/archives?page=3"/>
  <link rel="first" type="application/atom+xml;type=feed"
        href="http://example.org/archives?page=1"/>
  <link rel="last" type="application/atom+xml;type=feed"
        href="http://example.org/archives?page=12"/>
  <link rel="next" type="application/atom+xml;type=feed"
        href="http://example.org/archives?page=4"/>
  <link rel="previous" type="application/atom+xml;type=feed"
        href="http://example.org/archives?page=2"/>
  <entry>
    ...
  </entry>
</feed>
```

# Category

- Categories provide metadata to describe an Atom entry; Atom itself doesn't define usage of categories
- Category documents describe the categories that are allowed in collections
- Content type:  
`application/atomcat+xml`

```
<?xml version="1.0" encoding="utf-8"?>
<app:categories xmlns="http://www.w3.org/2005/Atom"
                xmlns:app="http://www.w3.org/2007/app"
                scheme="http://example.org/categories">
  <category term="blog"/>
  <category term="php"/>
  <category term="conference"/>
  <category term="elephant"/>
</app:categories>
```

# Service

- A service discovery document that describes the locations and capabilities of collections
- Usually the entry point of the Atom web service
- 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">
  <workspace>
    <atom:title>Ben's Blog</atom:title>
    <collection href="http://example.org/archives">
      <atom:title>Archives</atom:title>
      <accept>application/atom+xml;type=entry</accept>
      <categories href="http://example.org/categories"/>
    </collection>
  </workspace>
</service>
```

# Publication with Atom

# Creating resources

- Client “discovers” the resource through which the Atom service accepts POST requests to create resources (defined by app:collection)
- Client sends resource in POST request:  
POST /archives HTTP/1.1
- Client receives a 201 Created response in the event of a success

# Modifying resources

- Client requests a representation of the resource:  
GET /archives/1234 HTTP/1.1
- Client modifies the resource and tells the service to update it:  
PUT /archives/1234 HTTP/1.1
- Service responds with a 200 OK status to indicate success
- Client uses “edit” relation for this logic

# Removing resources

- Client sends a DELETE request to the service:

```
DELETE /archives/1234 HTTP/1.1
```

- Service responds with a 204 No Content status code to indicate success

# Authentication

- Atom does not provide or recommend an auth mechanism
- Early favorite of the Atom community was WSSE
- Consider the use of OAuth
- Send authentication info through HTTP headers on each request requiring authenticated access

# Encryption

- Atom does not provide for encryption
- Use SSL for all sensitive communication
- Atom Entry and Feed Documents can contain XML Digital Signatures
- Can be encrypted using XML Encryption

# Extending Atom

- Atom is XML; as such, it may be extended using other vocabularies (namespaces)
- Use other namespaces to convey other information
- e.g. OpenSearch
- e.g. Dublin Core

# Parsing Atom

- DOM, XMLReader, XMLWriter, SimpleXML
- Zend\_Feed, PEAR::XML\_Feed\_Parser, phpatomlib
- Apache Abdera

# Out-of-box Atom clients

- Fude -  
<http://www.witha.jp/eXeries/software/>
- Atomic (Firefox plugin) -  
<https://addons.mozilla.org/en-US/firefox/addon/3188>
- APP Test Client -  
<http://bitworking.org/projects/apptestclient/>

# For more information...

- <http://tools.ietf.org/html/rfc4287>
- <http://tools.ietf.org/html/rfc5023>
- <http://atomenabled.org/>
  
- My blog: <http://benramsey.com/>