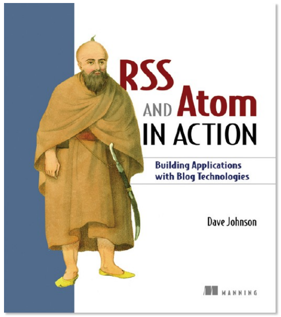


Beyond Blogging: Feeds in Action

Dave Johnson

Staff Engineer / SW
Sun Microsystems, Inc.
http://rollerweblogger.org/roller

Session TS-6029

Goal

What you'll learn in this session

Understand RSS and Atom feed formats, the Atom Publishing Protocol.

Understand how to use ROME to consume and produce feeds.

Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

The future...

Why talk about blogging at JavaOne?

- Blogs made the web easier
- For writers, readers and *software developers*
- Blogs brought XML to the masses

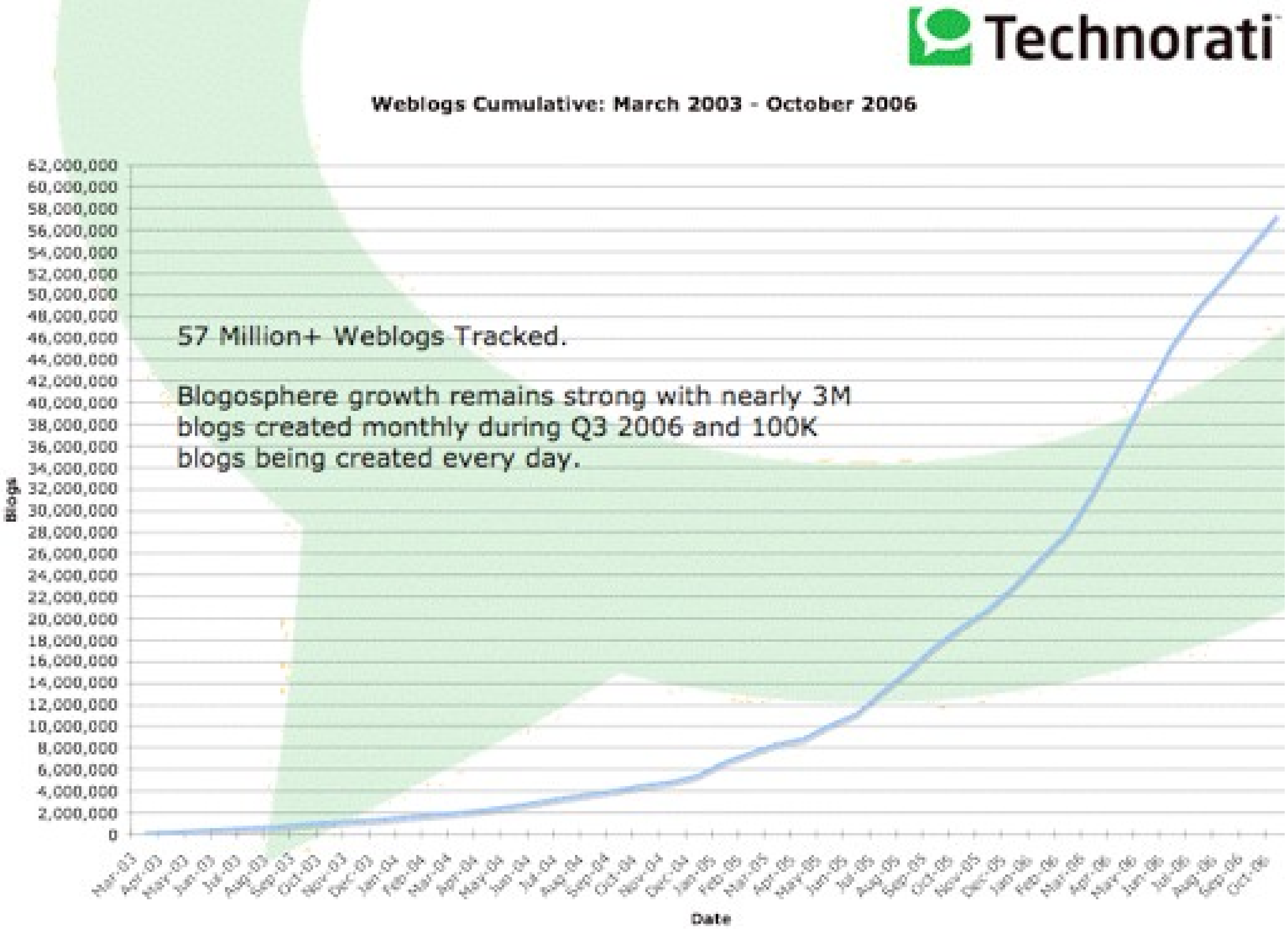
Bloggers didn't invent XML

- But they perfected and popularized XML feeds
  - e.g. Dave Winer, Dan Libby and **RSS**
  - e.g. Gregorio, Pilgrim, Ruby and **Atom**

- And kicked off XML web services
  - e.g. Dave Winer created XML-RPC, precursor to SOAP, for his Frontier CMS

- And then blogging hit the big time...

State of the blogosphere



Weblogs Cumulative: March 2003 - October 2006

57 Million+ Weblogs Tracked.

Blogosphere growth remains strong with nearly 3M blogs created monthly during Q3 2006 and 100K blogs being created every day.

Suddenly everybody has a blog

- Suddenly it's easy for software to monitor, parse, publish, filter and aggregate web content

- And the web is bloggy
    - Every web site has XML feeds
    - Every web site has a simple XML API

- Bloggy?

That's right, bloggy

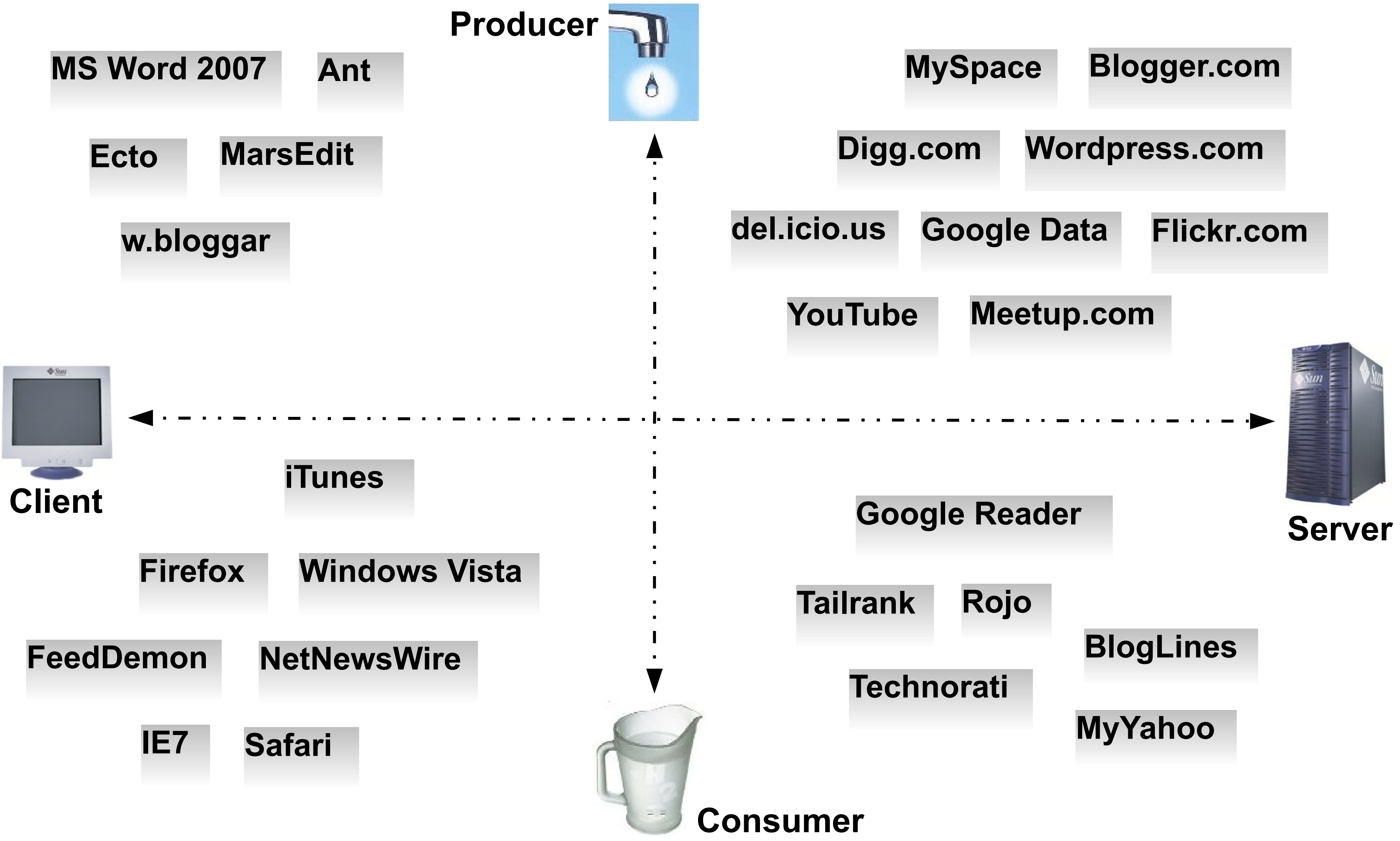
- Everything is a time-stamped, uniquely identified chunk of data with meta-data

- News stories
- Search results
- Uploaded photos
- Events and meetups
- Podcasts and Vodcasts

- Bug reports
- Wiki changes
- Source code changes
- O/S log messages

- OK, not everything, but you get the idea...

Feeds on the web today



Producer

MS Word 2007

Ant

Ecto

MarsEdit

w.bloggar

MySpace

Blogger.com

Digg.com

Wordpress.com

del.icio.us

Google Data

Flickr.com

YouTube

Meetup.com

Client

Server

iTunes

Firefox

Windows Vista

FeedDemon

NetNewsWire

IE7

Safari

Google Reader

Tailrank

Rojo

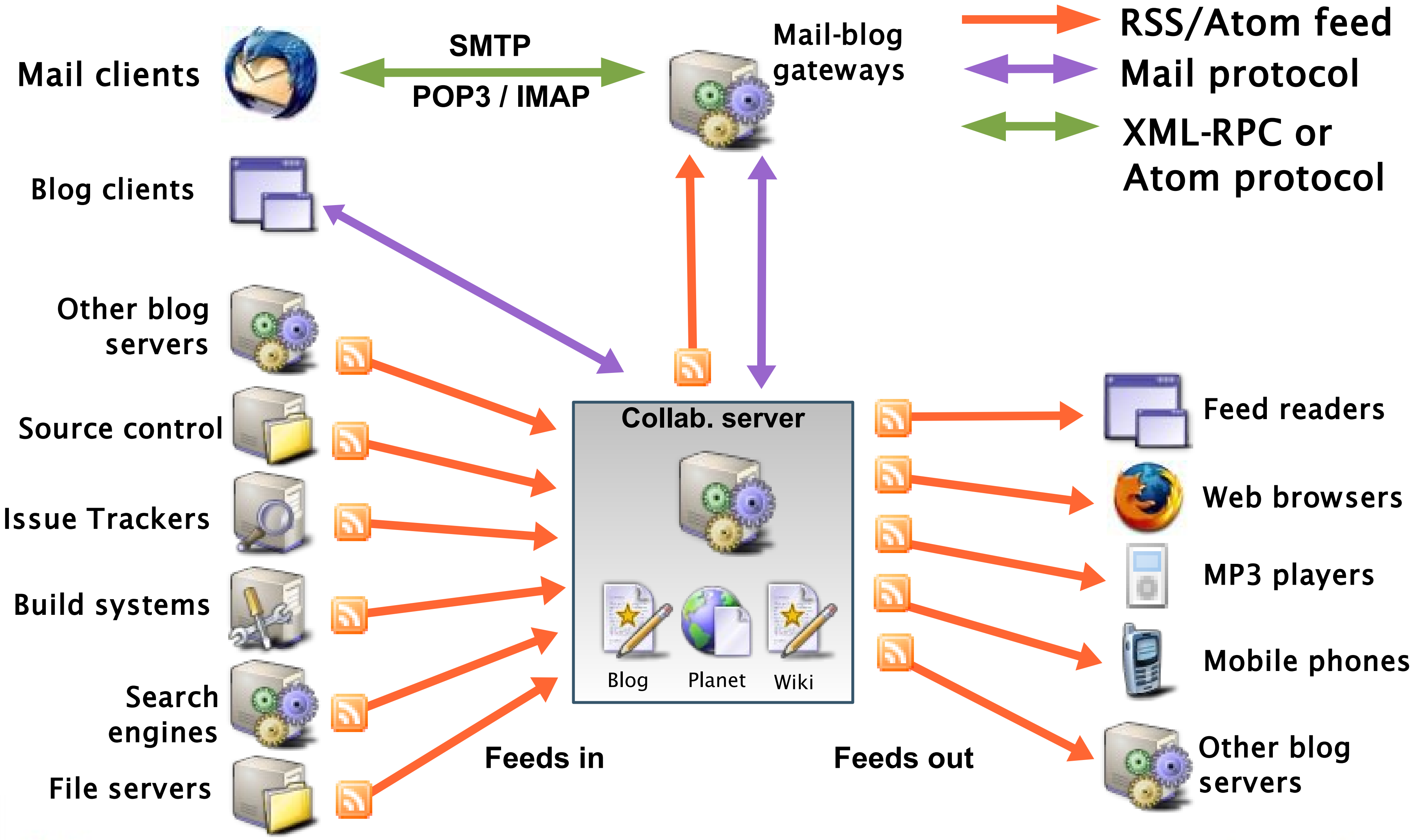
BlogLines

Technorati

MyYahoo

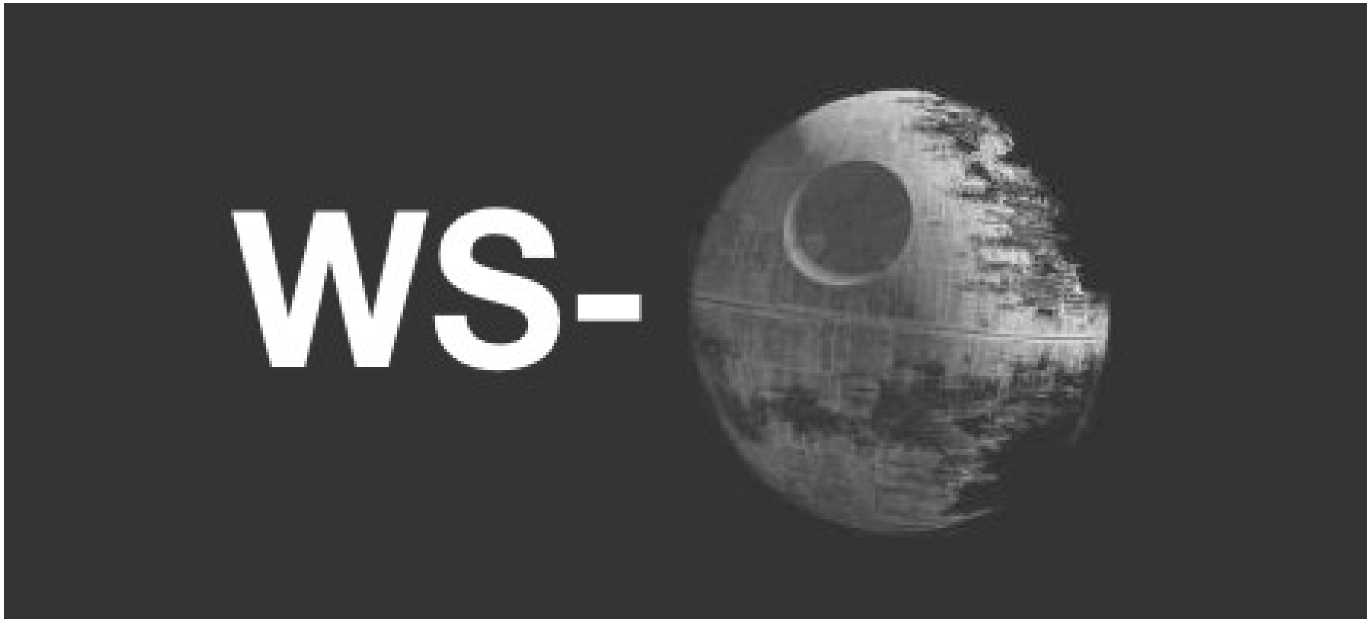
Consumer

Feeds as an integration technology



Meanwhile: web services got uppity

- SOAP took over where XML-RPC left off
- WSDL, UDDI and Schema exploded into today's complex WS-* stack.



But most developers didn't follow

- **Developers prefer REST**
  - *“Amazon has both SOAP and REST interfaces to their web services, and 85% of their usage is of the REST interface.”* -- Tim O'Reilly

- **And even WS-Advocates agree**
  - *“for applications that require Internet scalability (e.g., mass consumer-oriented services),* ***plain old XML (POX) is a much better solution than WS-*.*** *”*

  -- Anne Thomas Mannes

And now RSS and Atom are emerging

- As a foundation for simple web services
- For example:
  - Yahoo Pipes for end-user mash-ups via RSS
  - Google Data using Atom Publishing Protocol
  - Lucene-WS using Atom Publishing Protocol
  - Eclipse's Europa build system
- Let's return to the topic of feeds

Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

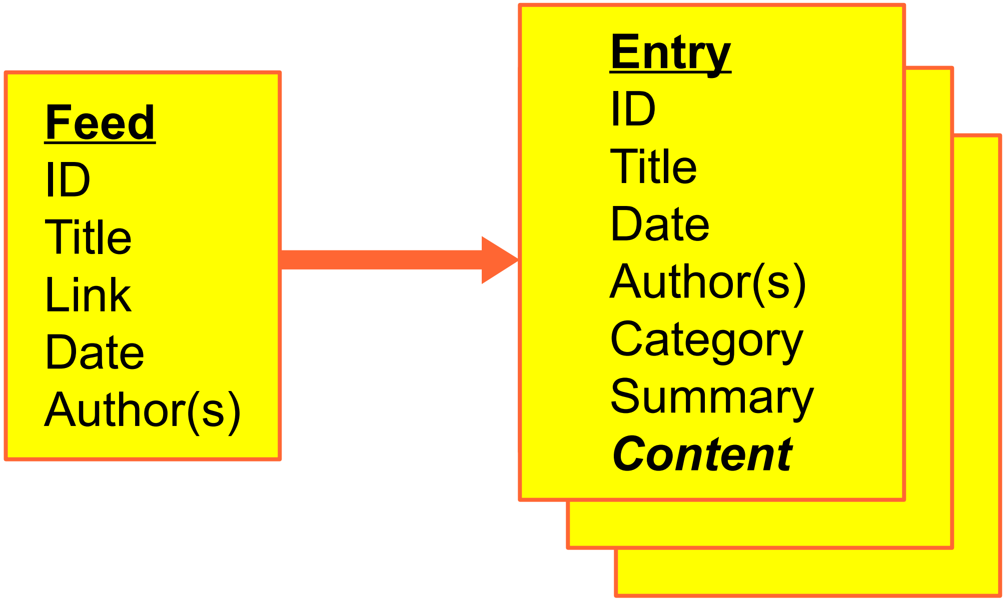
Producing feeds with ROME

Publishing with ROME Propono

The future...

What Is a Feed?

- XML representation of uniquely identified, time-stamped data items with metadata
- Available on the web at a fixed URL



Feed
ID
Title
Link
Date
Author(s)

Entry
ID
Title
Date
Author(s)
Category
Summary
Content

The birth of the RSS feed format

- RSS began life at Netscape in 1999
  - First spec was RSS 0.90 by Dan Libby
  - Created for the My Netscape portal
  - Known as RDF Site Summary (RSS)
- Dave Winer helped with 0.91, removed RDF
- 0.9X formats are obsolete but still in use today

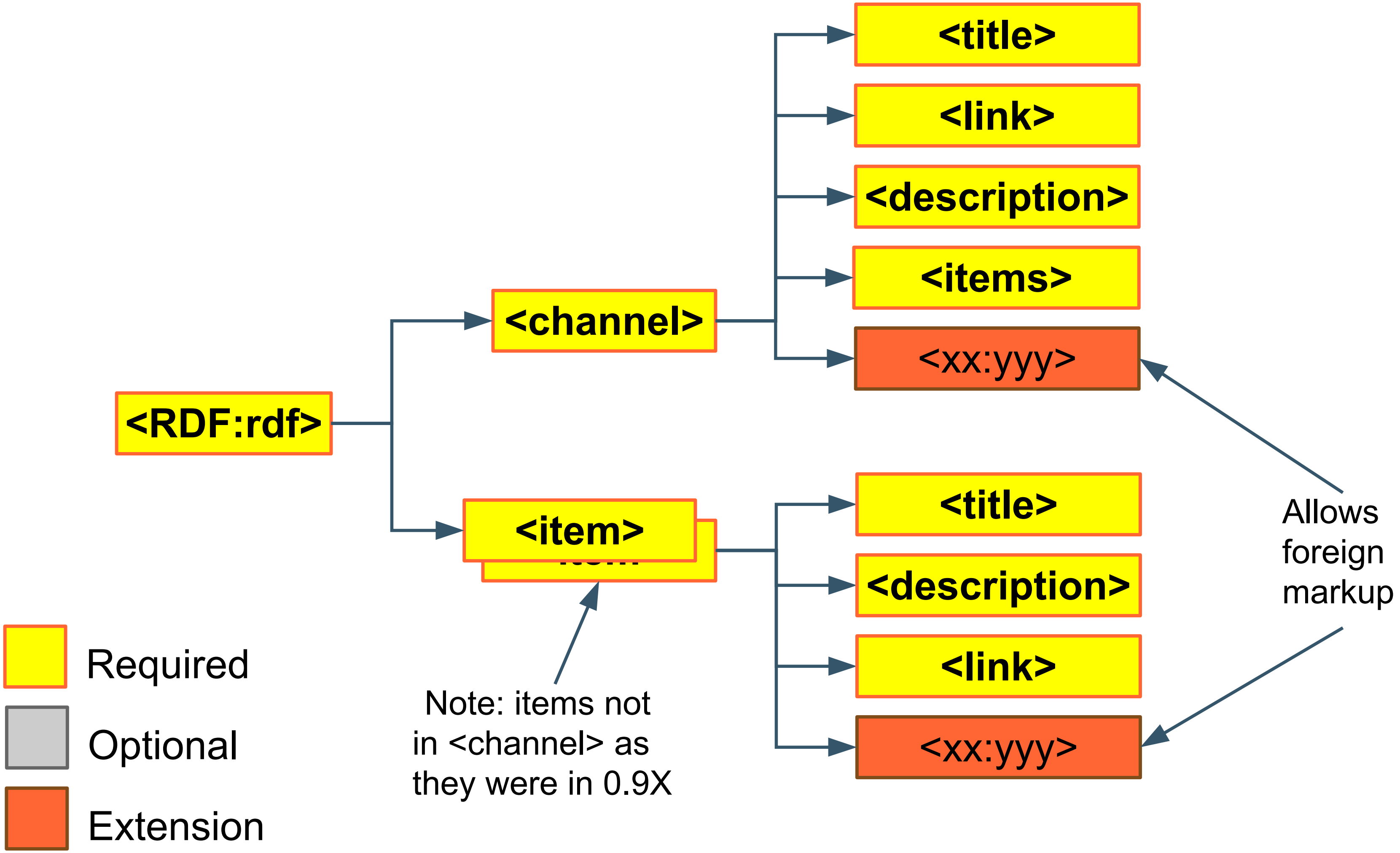
The RDF fork: RSS 1.0

- After RSS 0.91, Winer tried to keep RSS simple
- RDF folks argued for extensibility

- The RDF folks declared victory and released 1.0
  - Small set of elements, augmented by RDF
  - And **Extension Modules**

- Adopted by Movable Type and many others
- RSS 1.0 is still widely used today

Elements of RSS 1.0 (abridged)



<RDF:rdf>

<channel>

<title>

<link>

<description>

<items>

<xx:yyy>

<item>

<title>

<description>

<link>

<xx:yyy>

Allows foreign markup

Note: items not in <channel> as they were in 0.9X

🟨 Required

⬜ Optional

🟧 Extension

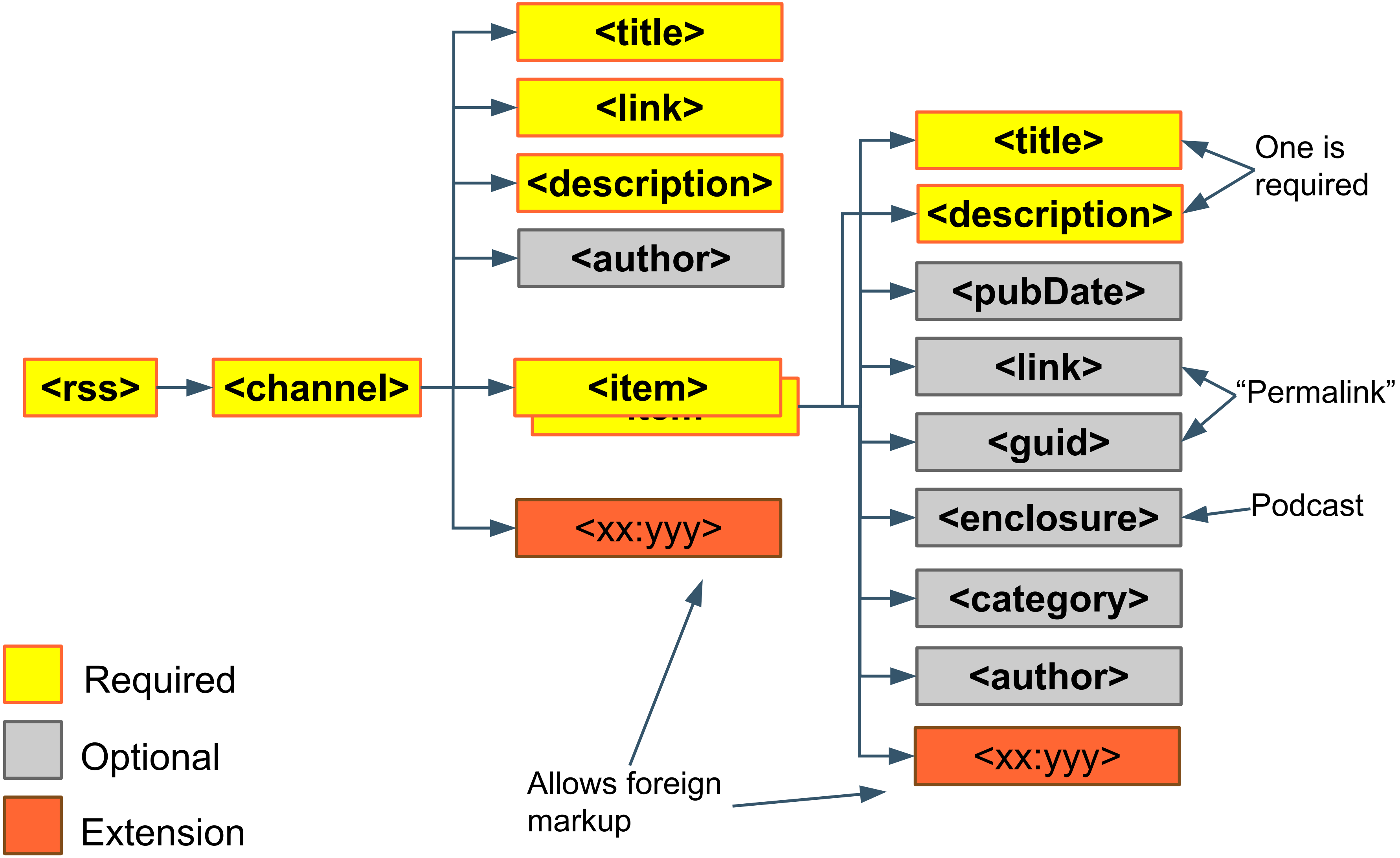
Feed Extension Modules

- *An Extension Module is a set of XML extension elements sharing a common name-space*
- Examples:
  - GeoRSS
  - iTunes
  - Slashdot
  - etc.

The simple fork: RSS 0.92 – RSS 2.0

- Winer rejected 1.0 and continued with 0.92, 0.93 and *finally* 2.0. Along the way RSS:
  - Added more metadata
  - Added `<enclosure>` element – Podcasting!
  - Added support for Extension Modules
  - Made elements under `<item>` optional
- RSS 2.0 declared to be final version of RSS

Elements of RSS 2.0 (abridged)



<rss>

<channel>

<title>

<link>

<description>

<author>

<item>

<xx:yyy>

<title>

<description>

<pubDate>

<link>

<guid>

<enclosure>

<category>

<author>

<xx:yyy>

One is required

“Permalink”

Podcast

Allows foreign markup

- 🟨 Required
- ⬜ Optional
- 🟧 Extension

RSS 2.0 Example

```
<rss version="2.0">
<channel>
<title>Latest Bugs</title>
<link>http://bugtrack/bugreport</link>

  <item>
    <title>Blue screen on refresh</title>
    <link>http://bugtrack/bugreport?id=132</link>
    <description>
      This is &lt;b&gt;very&lt;b&gt; bad.
    </description>
    <pubDate>Fri, 11 May 2007 15:00:00 EDT</pubDate>
  </item>

</rss>
```

Funky RSS: overuse of extensions?

```
<rss version="2.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
<title>Latest Bugs</title>
<link>http://bugtrack/bugreport</link>
  <item>
    <title>Blue screen on refresh</title>
    <link>http://bugtracker/bugreport?id=132</link>
    <description>This is &lt;b&gt;very&lt;b&gt; bad.
    </description>
    <dc:date>2007-05-11T15:00:00-00:00</dc:date>
    <dc:creator>Joe Tester</dc:creator>
  </item>
</rss>
```

RSS limitations

- Spec is too loose and unclear
  - What fields can be escaped HTML?
  - How many enclosures are allowed per item?
- Content model is weak
  - No support for summary and content
  - Content-type and escaping not specified
- Specification is final and can not be clarified

What is Atom?

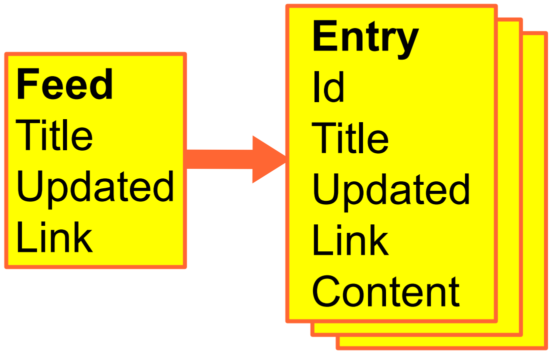
- From the IETF Atom WG charter:

> *Atom defines a* ***feed format for representing and a protocol for editing Web resources*** *such as Weblogs, online journals, Wikis, and similar content.*

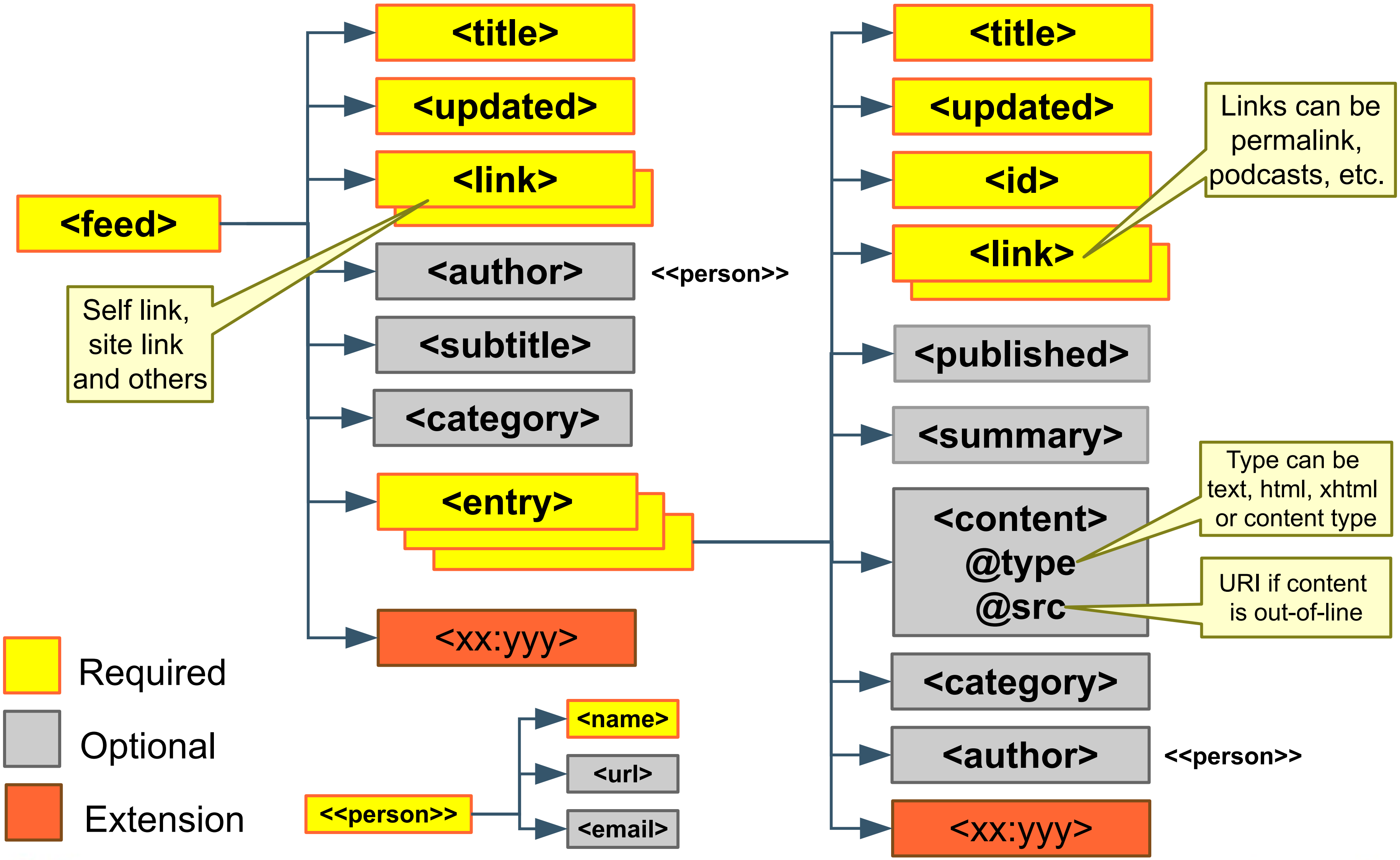
- Feed format is now IETF RFC-4287
- Protocol will be finalized in 2007

Atom Publishing Format

- An XML feed format. Feed contains entries
- Entries are
  - Time-stamped, uniquely ID'ed chunks of data
  - With meta-data: title, dates, categories
  - Entry content can be:
    - TEXT, HTML, XHTML or *any content-type*
    - In-line or out-of-line specified by URI
    - Binary data w/Base64 encoding
- *It's generic, not just for blogs.*



Elements of Atom (abridged)



<feed>

<title>

<updated>

<link>

<author> <<person>>

<subtitle>

<category>

<entry>

<xx:yyy>

Self link, site link and others

<title>

<updated>

<id>

<link>

Links can be permalink, podcasts, etc.

<published>

<summary>

<content> @type @src

Type can be text, html, xhtml or content type

URI if content is out-of-line

<category>

<author> <<person>>

<xx:yyy>

<<person>>

<name>

<url>

<email>

🟨 Required

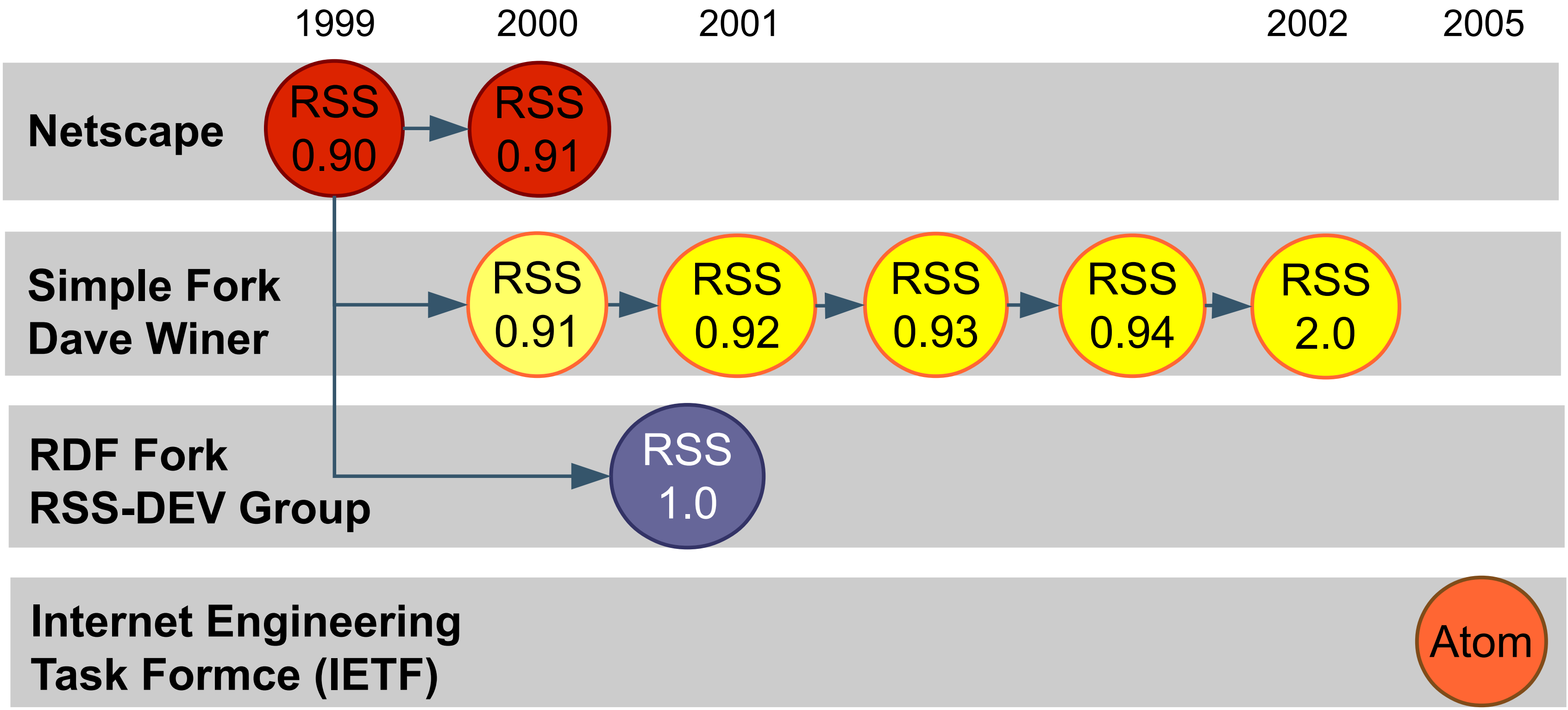
⬜ Optional

🟧 Extension

Atom <feed> with one <entry>

```
<feed xmlns='http://www.w3.org/2005/Atom'>
   <title>Latest Bugs</title>
   <link href='http://bugtracker/bugreport' />
   <link rel='self'
      href='http://bugtracker/feeds/bugreport'/>
   <updated>2007-05-11T15:00:00-00:00</updated>
   <author><name>BugTracker-5000-XL</name></author>
   <entry>
      <title>Blue screen on refresh</title>
      <link href='http://bugtracker/bugreport?id=132' />
      <id>http://bugtracker/bugreport?id=132</id>
      <updated>2007-05-11T15:00:00-00:00</updated>
      <content type='html'>
         This is &lt;b&gt;very&lt;b&gt; bad.
      </content>
   </entry>
</feed>
```

RSS and Atom feed family tree



Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

The future...

Parsing and fetching feeds

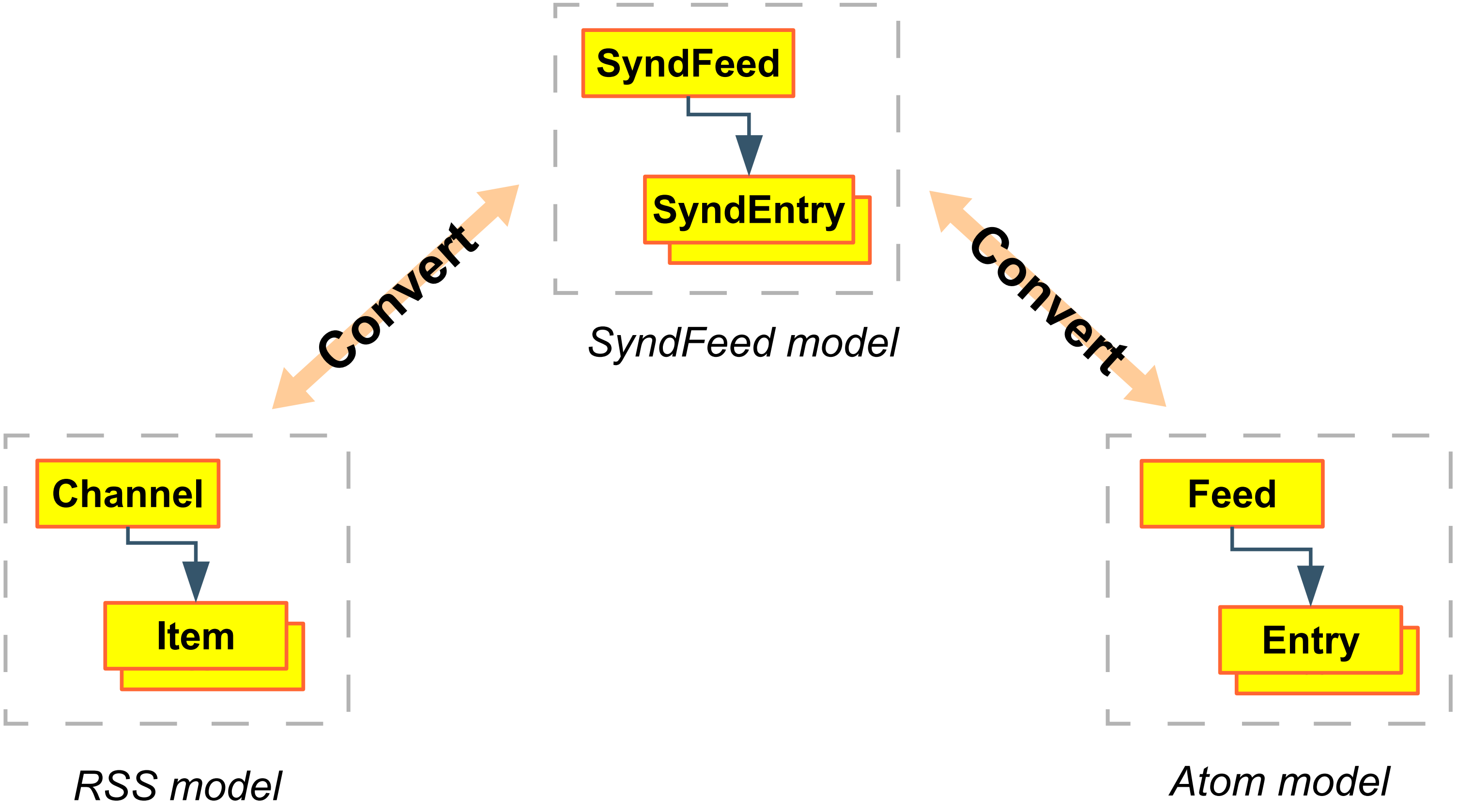
- It's just XML!
  - Use your favorite parsing technique

- Or better yet... use a parser library
  - **ROME**: DOM based feed parser/generator (Java)
  - **Abdera**: STAX based *Atom-only* parser (Java)
  - **Universal Feed Parser** (Python)
  - **Windows RSS Platform**: parser built-in to IE7

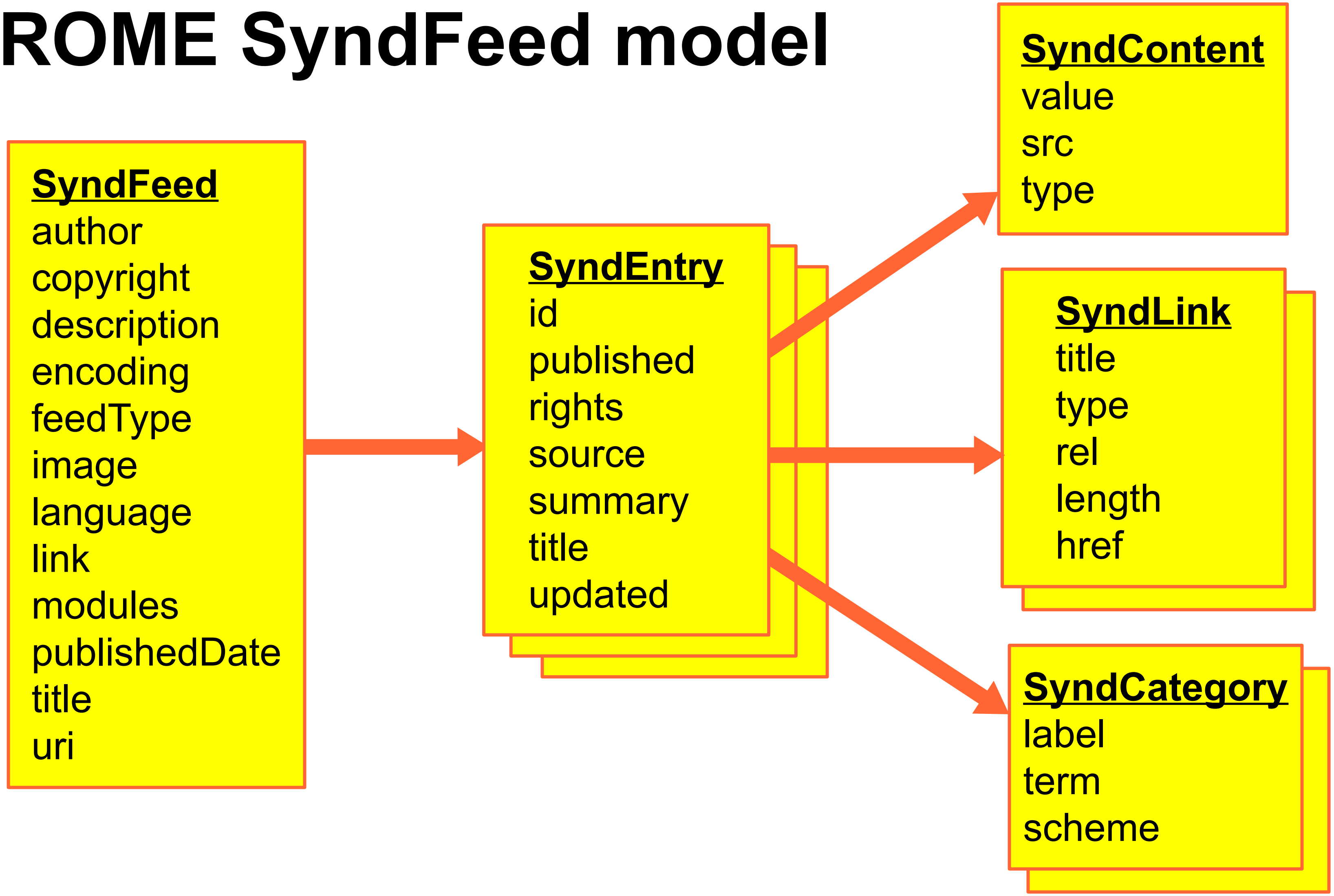
ROME RSS/Atom feed utilities

- Most capable Java based toolkit
- Pros
  - Parses / generates all forms of RSS and Atom
  - Highly pluggable/extensible, based on JDOM
  - Parses to Atom, RSS or abstract object model
- Con: DOM based
- Free and open source (Apache license)

How does ROME work?



ROME SyndFeed model



SyndFeed
author
copyright
description
encoding
feedType
image
language
link
modules
publishedDate
title
uri

SyndEntry
id
published
rights
source
summary
title
updated

SyndContent
value
src
type

SyndLink
title
type
rel
length
href

SyndCategory
label
term
scheme

Parsing a feed with ROME SyndFeed

```
SyndFeedInput input = new SyndFeedInput();
SyndFeed feed = input.build(
   new InputStreamReader(inputStream));

Iterator entries = feed.getEntries().iterator();

while (entries.hasNext()) {
   SyndEntry entry = (SyndEntry)entries.next();
   System.out.println("Title: " + entry.getTitle());
   System.out.println("Link: " + entry.getLink());
   System.out.println("\n");
}
```

How to fetch feeds

- Be nice and conserve bandwidth
    - Use HTTP conditional GET or Etags
    - Don't poll too often

- Your parser library might do the work for you
    - ROME's Fetcher provides a caching feed-store
    - Other parsers do too

Fetching a feed with ROME Fetcher

```
FeedFetcherCache cache =
   new DiskFeedInfoCache("/var/rome-fetcher/cache");
FeedFetcher fetcher = new HttpURLFeedFetcher(cache);

SyndFeed feed = fetcher.retrieveFeed(
   new URL("http://bugtracker/feeds/bugreport"));

Iterator entries = feed.getEntries().iterator();
while (entries.hasNext()) {
   SyndEntry entry = (SyndEntry)entries.next();
   // ... omitted: print out entry ...
}
```

Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

The future...

Serving feeds: generate XML

- Use your favorite XML tools or...
- Templates languages like JSP, PHP, ASP.Net

- Or better yet: a feed toolkit like ROME

Generating Atom with ROME, pt. 1/2

```
SyndFeed syndFeed = new SyndFeedImpl();
syndFeed.setTitle("Latest bugs");
syndFeed.setAuthor("BugTrack-9000-XL");
syndFeed.setPublishedDate(BugManager.getUpdateDate());
syndFeed.setLink("http://localhost/bugtracker");
syndFeed.setUri(syndFeed.getLink());

SyndLink selfLink = new SyndLinkImpl();
selfLink.setRel("self");
selfLink.setHref("http://localhost/bugtracker/latest.atom");
syndFeed.setLinks(Collections.singletonList(selfLink));

List entries = new ArrayList();
syndFeed.setEntries(entries);
```

Atom ID

Generating Atom with ROME, pt. 2/2

```
Iterator bugs = BugManager.getLatestBugs(20).iterator();
while (bugs.hasNext()) {
   Bug bug = (Bug)bugs.next();
   SyndEntry entry = new SyndEntryImpl();
   entry.setTitle(bug.getTitle());
   entry.setUpdatedDate(bug.getDateAdded());
   entry.setLink(
      "http://bugtracker/?bugid=" + bug.getId());
   entry.setUri(entry.getLink());
   SyndContent content = new SyndContentImpl();
   content.setValue(bug.getDescription());
   content.setType("html");
   entry.setContents(Collections.singletonList(content));
   entries.add(entry);
}
```

Atom ID

Serving feeds: serve it up

- Set the right content-type

```
application/rss+xml
application/atom+xml
```

- Cache cache cache!
  - On client-side via HTTP Conditional GET
  - On proxy servers via HTTP headers
  - On server-side via your favorite cache tech.

Serving Atom with ROME, pt. 1/2

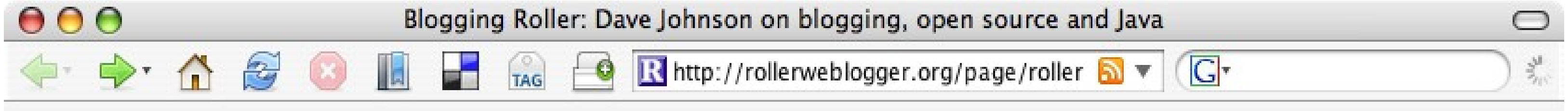
```
public class BugFeedServlet extends HttpServlet {
   LRUCache cache = new LRUCache(5, 5400);

   protected void doGet(HttpServletRequest req, // ...omitted

      Date since = new Date(
         req.getDateHeader("If-Modified-Since"));
      if (sinceDate != null) {
         if (BugManager.getUpdateDate().compareTo(since) <= 0) {
            res.sendError(HttpServletResponse.SC_NOT_MODIFIED);
            return;
         }
      }
      res.setDateHeader("Last-Modified",
         BugManager.getUpdateDate().getTime());
      res.setHeader("Cache-Control",
         "max-age=5400, must-revalidate");
```

Serving Atom with ROME, pt. 2/2

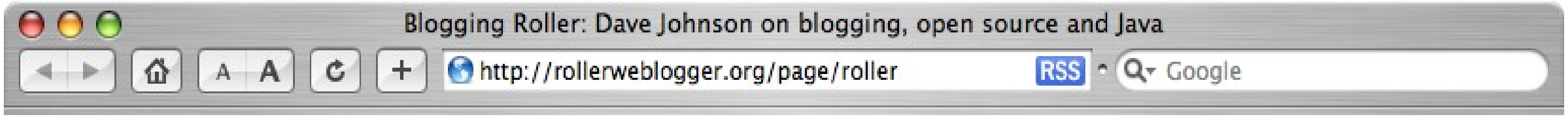
```
      String url = request.getRequestURL().toString();
      if (cache.get(url) == null) {

         SyndFeed syndFeed = // ...omitted
         syndFeed.setFeedType("atom_1.0");

         StringWriter stringWriter = new StringWriter();
         SyndFeedOutput output = new SyndFeedOutput();
         output.output(syndFeed, stringWriter);

         cache.put(request.getRequestURL().toString(),
                   stringWriter.toString());
      }
      response.setContentType(
         "application/xml+atom;charset=utf-8");
      response.getWriter().write((String)cache.get(url));
   }
}
```

Feed auto-discovery

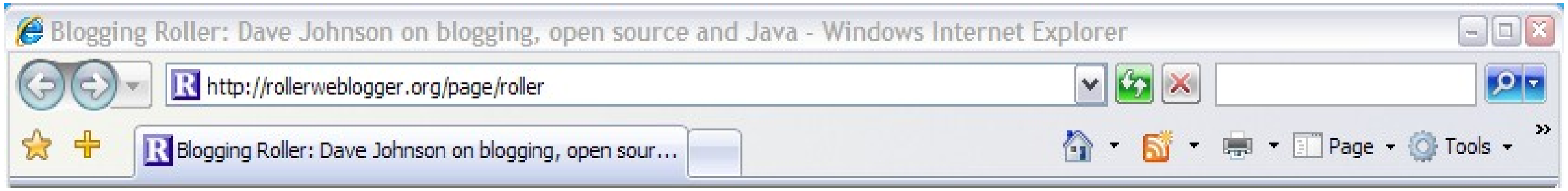
- Make it easy for applications to find your feeds
- Firefox can do it


Blogging Roller: Dave Johnson on blogging, open source and Java
TAG
http://rollerweblogger.org/page/roller

- Safari can too


Blogging Roller: Dave Johnson on blogging, open source and Java
A A
http://rollerweblogger.org/page/roller
RSS
Google

- And even IE


Blogging Roller: Dave Johnson on blogging, open source and Java - Windows Internet Explorer
http://rollerweblogger.org/page/roller
Blogging Roller: Dave Johnson on blogging, open sour…
Page
Tools

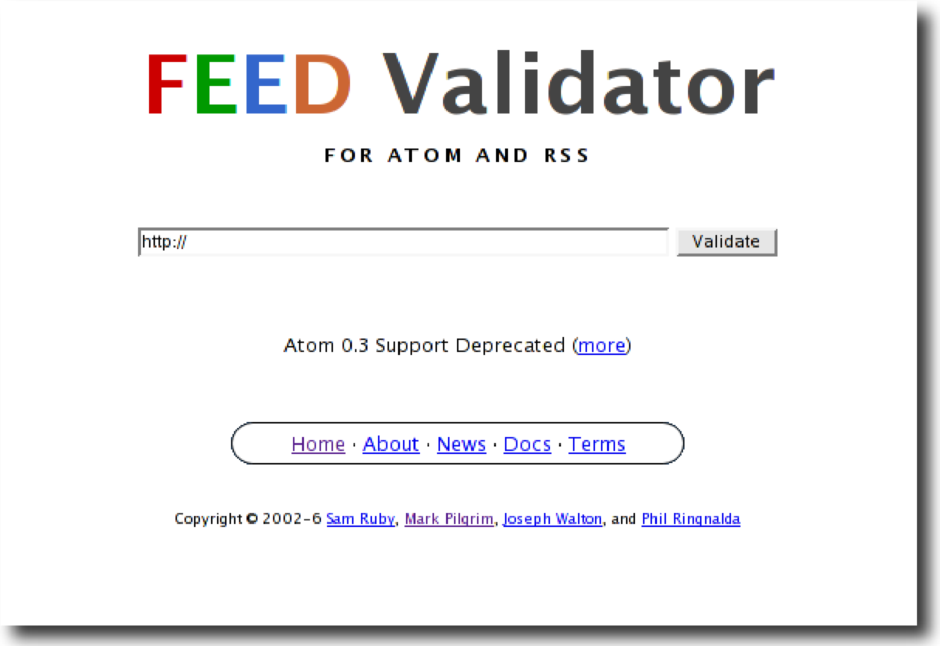
Feed auto-discovery

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html” />

  <link rel="alternate"
    type="application/atom+xml" title="Latest bugs (Atom)"
    href="http://bugtracker/feeds/bugreport" />

  <link rel="alternate"
    type="application/rss+xml" title="Latest bugs (RSS)"
    href="http://bugtracker/feeds/bugreport?format=rss" />

. . .
```

Serving valid feeds

- Ensure HTML is properly escaped
- Ensure XML is well formed

- Validate!
- feedvalidator.org



FEED Validator

FOR ATOM AND RSS

Atom 0.3 Support Deprecated (more)

Copyright © 2002-6 Sam Ruby, Mark Pilgrim, Joseph Walton, and Phil Ringnalda

Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

The future...

Feed publishing protocols

- **Blogger API**: Simple XML-RPC based protocol (by Blogger.com)
- **MetaWeblog API**: Extends Blogger API by adding RSS-based metadata (by Dave Winer)
- **Atom Publishing Protocol**: REST-based web publishing protocol uses Atom format (IETF).

The MetaWeblog API

getUserBlogs	Get blogs as array of structures
newPost	Create new blog post by passing in structure*
getPost	Get blog post by id
getRecentPosts	Get most recent N blog posts
editPost	Update existing blog post
deletePost	Delete blog post specified by id
newMediaObject	Upload file to blog (e.g. picture of my cat)
getCategories	Get categories allowed in blog

The Atom Publishing Protocol

“application-level protocol for publishing and editing Web resources using HTTP”

- Based on Atom Publishing Format
- Began as a replacement old blogging APIs
  - Grew into a generic publishing protocol

What does Atom protocol do?

- Everything MetaWeblog API does
- But it's generic, not just for blogs

- Entry can be any type of data
- CRUD on entries organized in collections
- Where CRUD = create, retrieve, update & delete

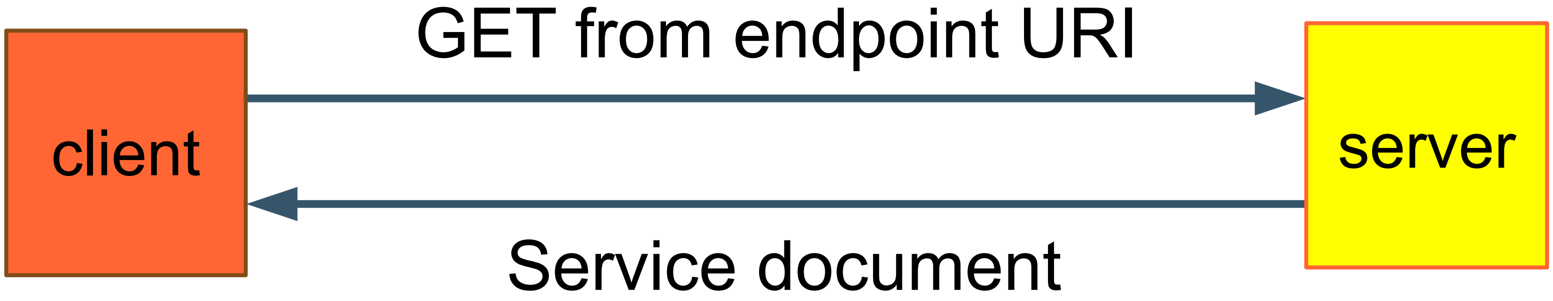
- Based on principals of REST

How does it do all that?

- The REST way:
  - Everything's a resource, addressable by URI
  - HTTP verbs used for all operations

- HTTP POST to create entries
- HTTP GET to retrieve entries and collections
- HTTP PUT to update entries
- HTTP DELETE to delete entries

APP Introspection



client

GET from endpoint URI

Service document

server

APP introspection document

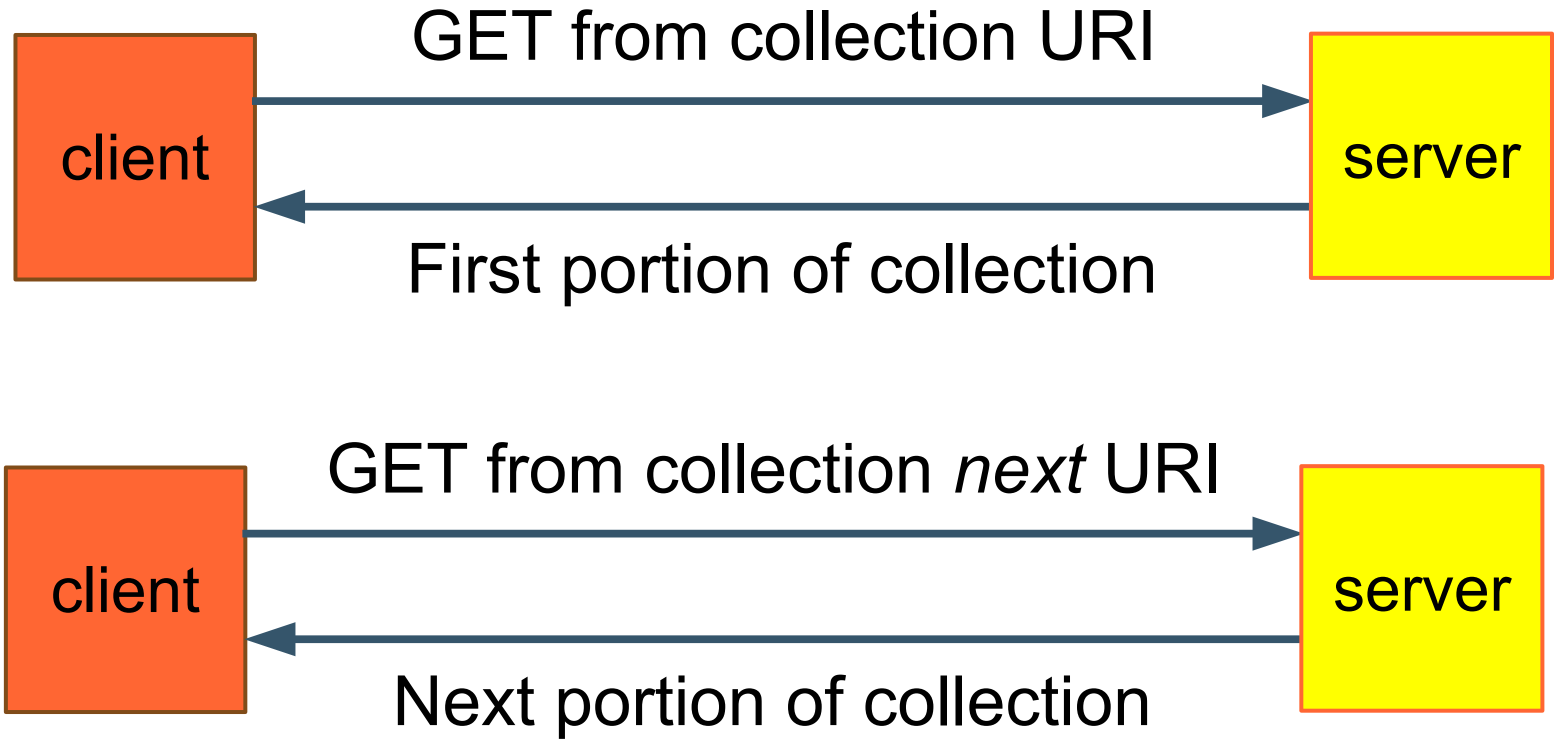
```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://purl.org/atom/app#">
    <workspace title="Order Management issues" >
        <collection title="Bug Reports"
            href="http://bugtrack/app/om/entries" >
            <accept>entry</accept>
        </collection>
        <collection title="Screenshots"
            href="http://bugtrack/app/om/screenshots" >
            <accept>image/*</accept>
        </collection>
    </workspace>
</service>
```

An Atom collection `<feed>`

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

  <link rel="next"
     href="http://example.org/blog/app/entries/60" />
  <link rel="previous"
     href="http://example.org/entries/20" />
  ...
  <entry> ... </entry>
  <entry> ... </entry>
  <entry> ... </entry>
  <entry> ... </entry>
  ...
</feed>
```

URIs for
next and previous
portions of collection

Getting an APP collection - with paging



GET from collection URI

client

server

First portion of collection

GET from collection *next* URI

client

server

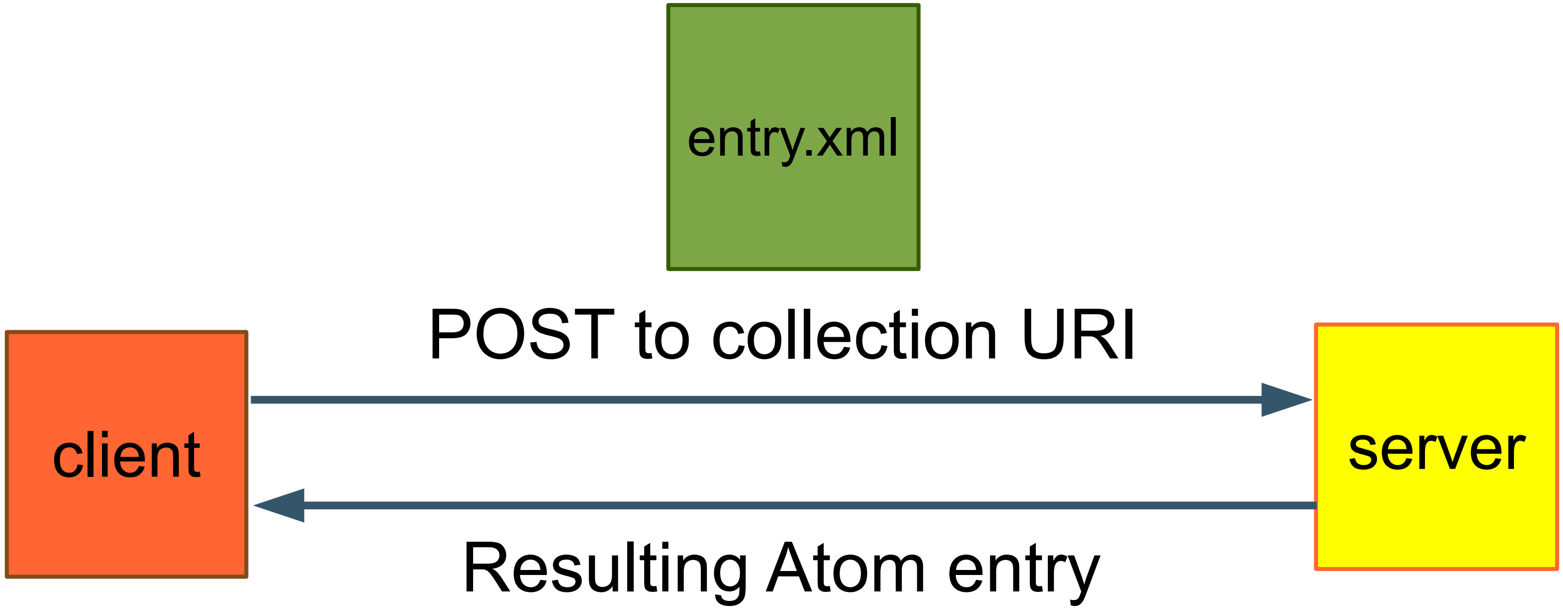
Next portion of collection

`<entry>` in a collection

```
  <entry>
    <title>NPE on new order query</title>
    <link rel="alternate"
       href="http://bugtracker/bugreport?id=757” />
    <link rel="edit"
       href="http://bugtracker/app/bug/757" />
    <id>http://bugtracker/bugreport?id=757</id>
    <updated>2007-05-08T22:08:03Z</updated>
    <published>2007-05-11T01:07:59Z</published>
    <content type="html">This is &lt;bad&gt; bad.
    </content>
  </entry>
</feed>
```

Edit URI for entry

Creating an entry



entry.xml

POST to collection URI

client

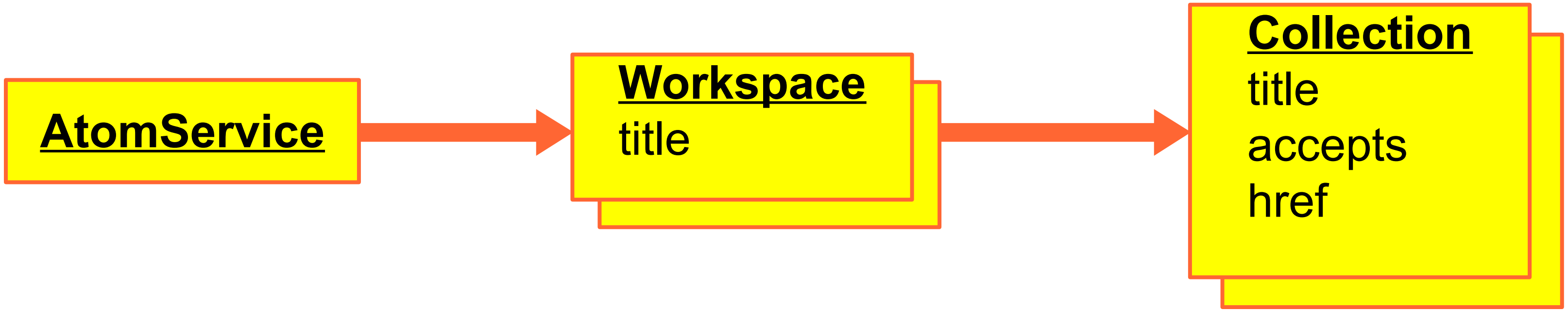
server

Resulting Atom entry

ROME Propono

- APP Client Library
  - Makes it easy to build an APP client app
- APP Server Library
  - Makes it easy to add an APP server to your webapp
- Blog Client Library
  - Suports both MetaWeblog API and APP
  - Blog centric and not as generic as APP Client Library

ROME Propono – Atom Common API

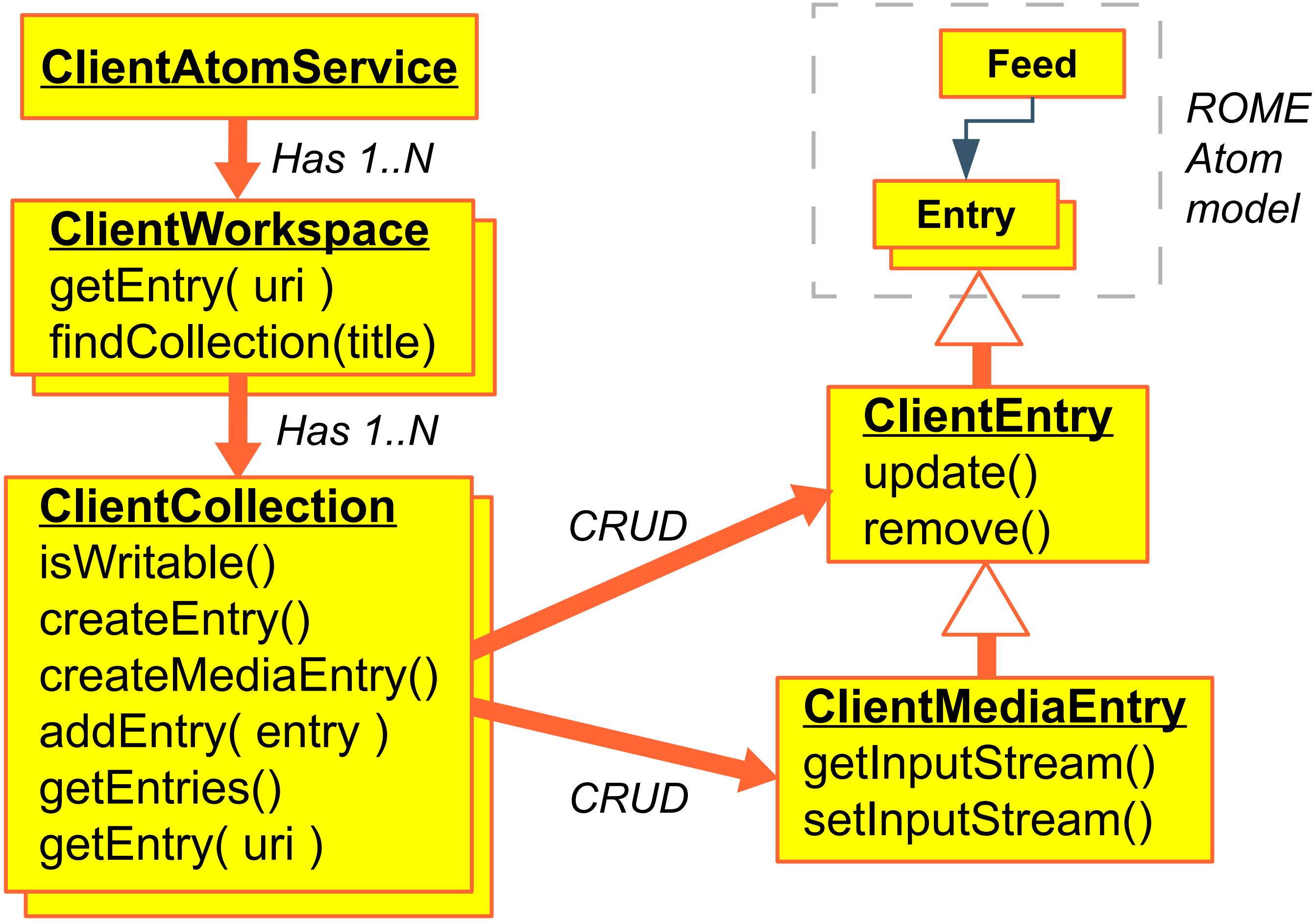


AtomService

Workspace
title

Collection
title
accepts
href

ROME Propono – Atom Client API



ROME Propono: posting an entry

```
ClientAtomService service =
   AtomClientFactory.getAtomService(endpoint, uname, pword);

ClientWorkspace ws =
   (ClientWorkspace)service.findWorkspace("Order System");

ClientCollection collection =
   (ClientCollection)ws.findCollection(null, "entry");

ClientEntry entry = collection.createEntry();
entry.setTitle("NPE on submitting new order query");
entry.setContent(Content.HTML, "This is a <b>bad</b> one!");
collection.addEntry(entry);
```

Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

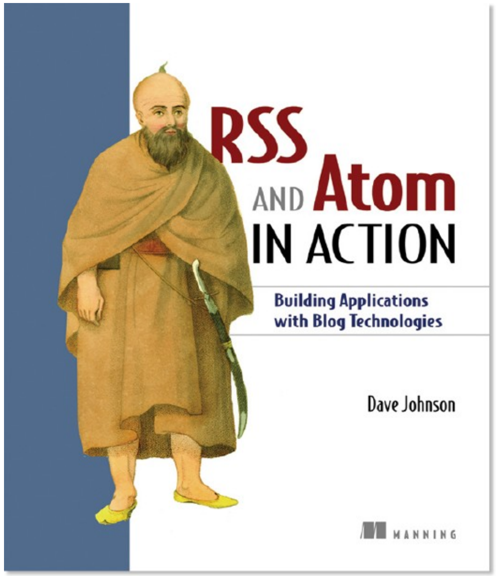
The future...

RSS/Atom trends

- Better RSS/Atom support in Java
  - Thanks to ROME and Abdera. Time for a JSR?
- More REST-based web services in general
  - Made easy by REST API, Restlets, XFire, etc.
- More web services based on Atom
  - APP as canonical REST protocol

For More Information

- Sun Web Developer Pack
  - http://developers.sun.com/web/swdp
- Related open source projects
  - http://rome.dev.java.net
  - http://incubator.apache.org/abdera
  - http://blogapps.dev.java.net
- RSS and Atom in Action
  - http://manning.com/dmjohnson



Summary

- RSS and Atom: not just for blogs anymore
- Feeds should be part of every developers tool-kit
- ROME has the tools you need for
  - Consuming and producing RSS and Atom feeds
  - Publishing to blogs via MetaWeblog API
  - Publishing to other systems via Atom protocol

Q&A

Dave Johnson