Beyond Blogging: Feeds in Action

Dave Johnson

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

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.

java.sun.com/javaone

Agenda

The web is bloggy

Understanding RSS and Atom

Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

The future…

java.sun.com/javaone

Why Talk About Blogging at 2007 JavaOne^{SM} Conference?

- Blogs made the web easier
- For writers, readers, and software developers
- Blogs brought XML to the masses
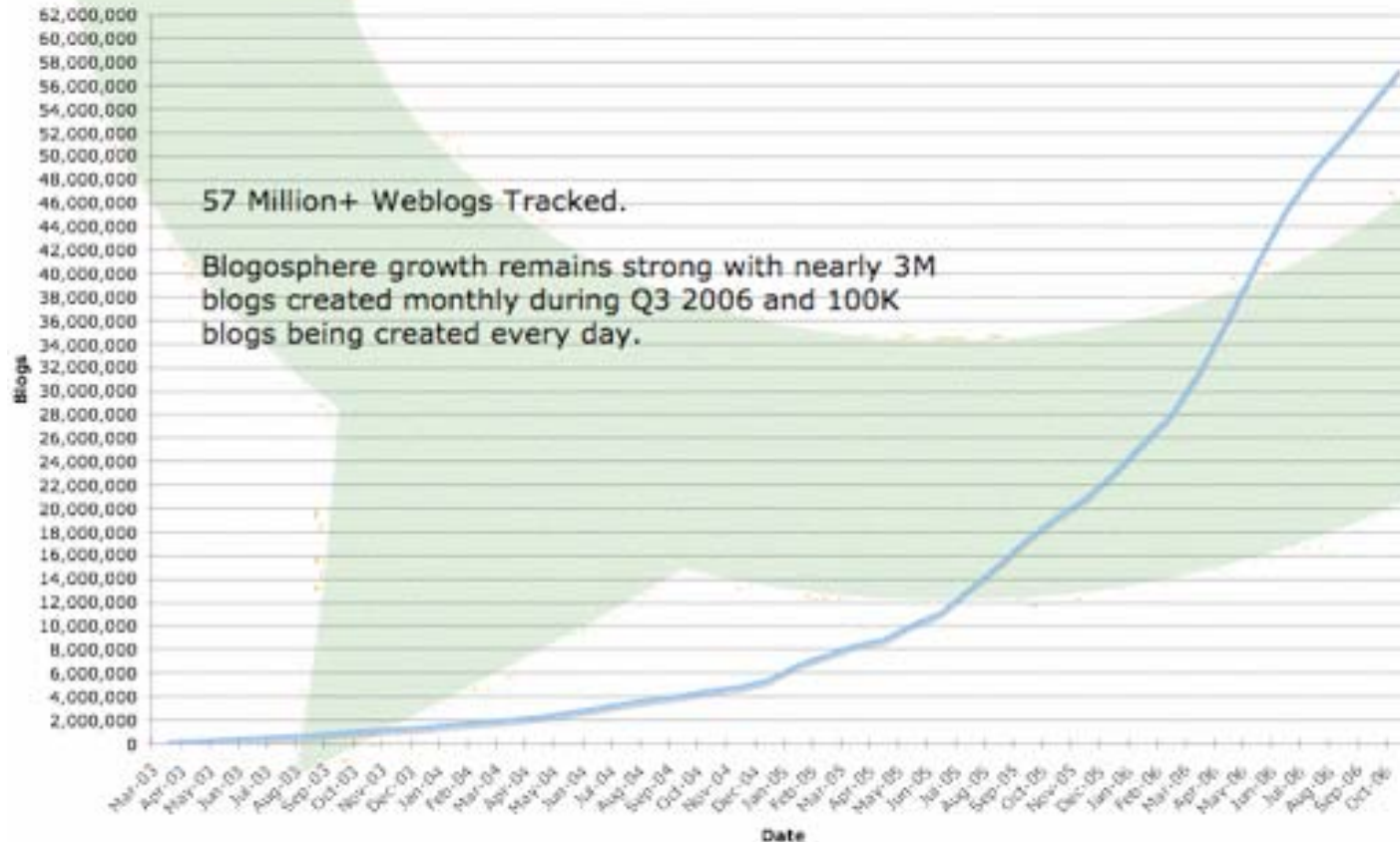
java.sun.com/javaone

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…

java.sun.com/javaone

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
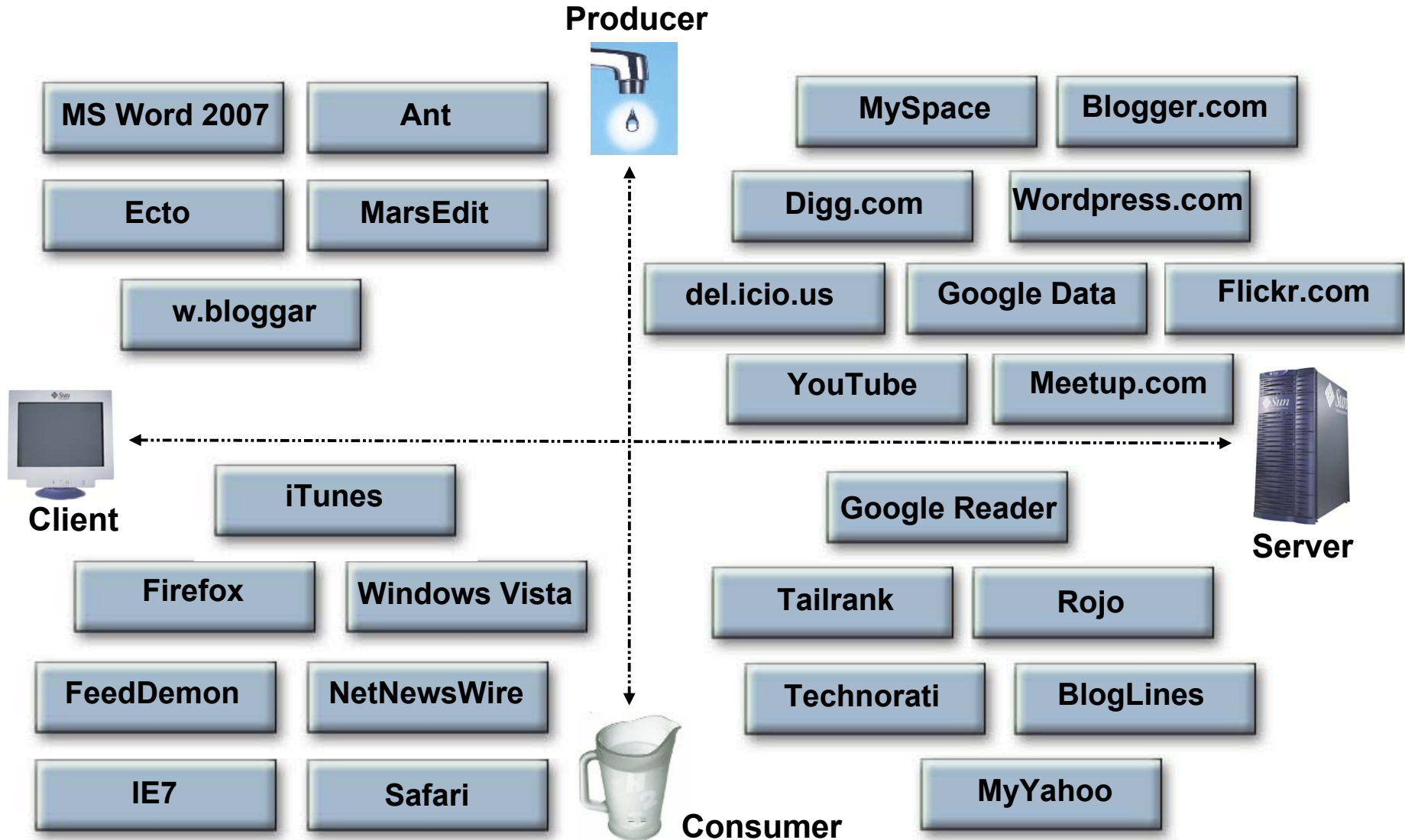blogs being created every day.

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…

java.sun.com/javaone

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

iTunes

Firefox

Windows Vista

FeedDemon

NetNewsWire

IE7

Safari

Server

Google Reader

Tailrank

Rojo

Technorati

BlogLines

MyYahoo

Consumer

java.sun.com/javaone

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

java.sun.com/javaone

Agenda

The web is bloggy

Understanding RSS and Atom
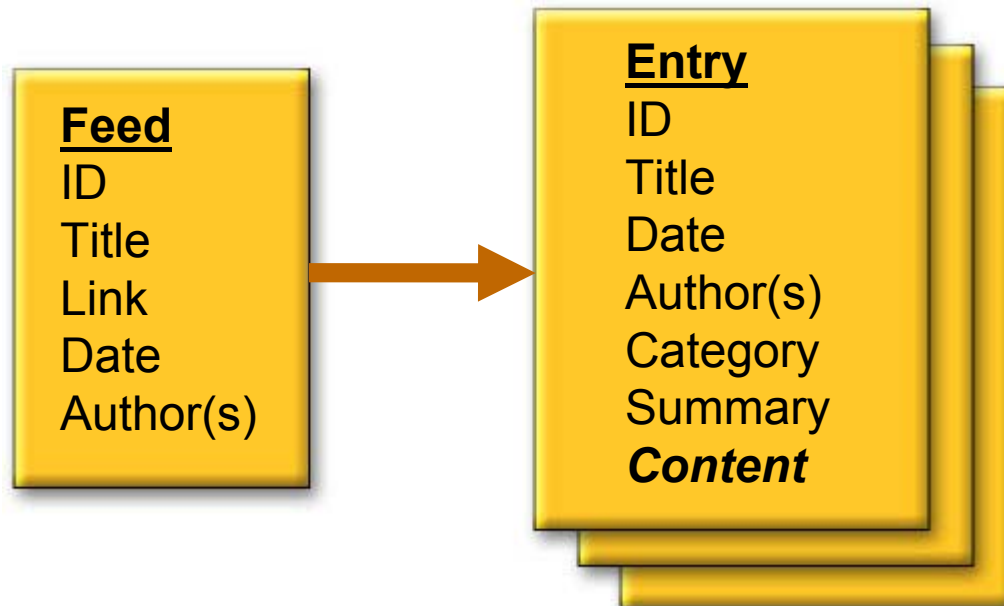
Consuming feeds with ROME

Producing feeds with ROME

Publishing with ROME Propono

The future…

java.sun.com/javaone

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
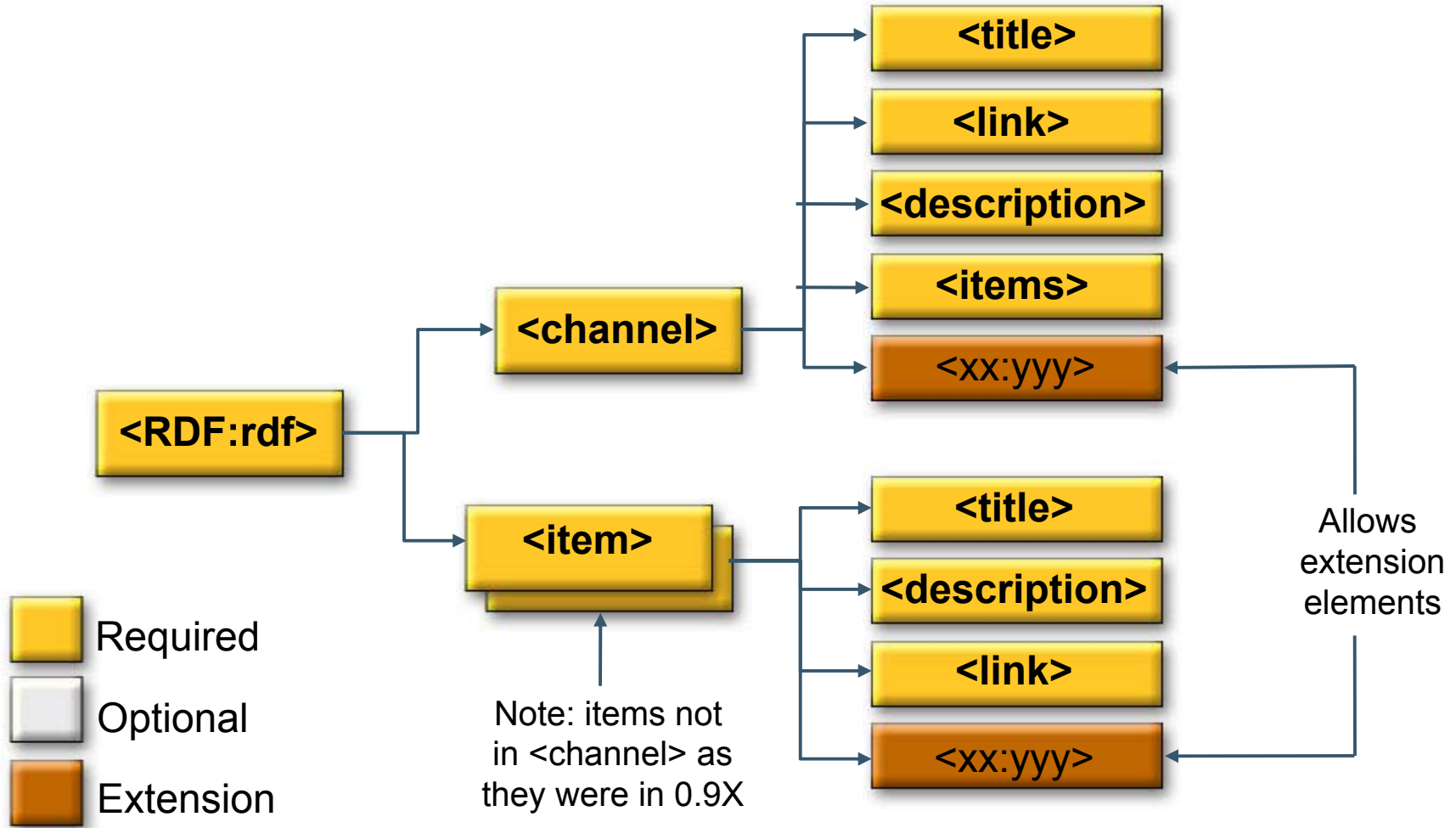*Content*

java.sun.com/javaone

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

java.sun.com/javaone

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
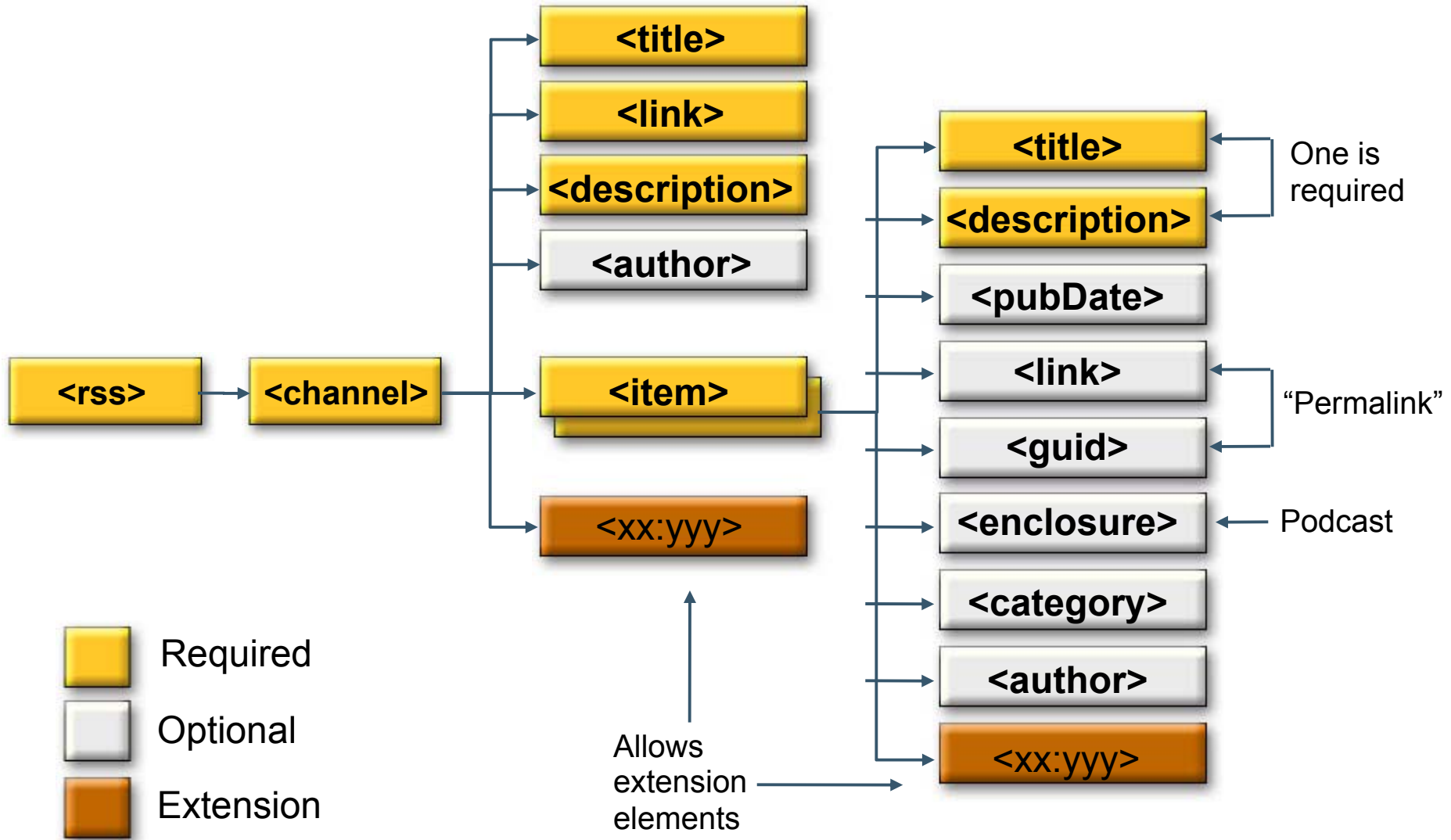- RSS 1.0 is still widely used today

Elements of RSS 1.0 (Abridged)

Required

Optional

Extension

`<RDF:rdf>`

`<channel>`

`<item>`

`<title>`

`<link>`

`<description>`

`<items>`

`<xx:yyy>`

`<title>`

`<description>`

`<link>`

`<xx:yyy>`

Allows extension elements

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

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

java.sun.com/javaone

Elements of RSS 2.0 (Abridged)

Required

Optional

Extension

`<rss>` → `<channel>` →

- `<title>`
- `<link>`
- `<description>`
- `<author>`
- `<item>`
- `<xx:yyy>`

`<item>` →

- `<title>` ⎤
- `<description>` ⎦ One is required
- `<pubDate>`
- `<link>` ⎤
- `<guid>` ⎦ "Permalink"
- `<enclosure>` ← Podcast
- `<category>`
- `<author>`
- `<xx:yyy>`

Allows extension elements

java.sun.com/javaone

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>
```

java.sun.com/javaone

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 cannot be clarified
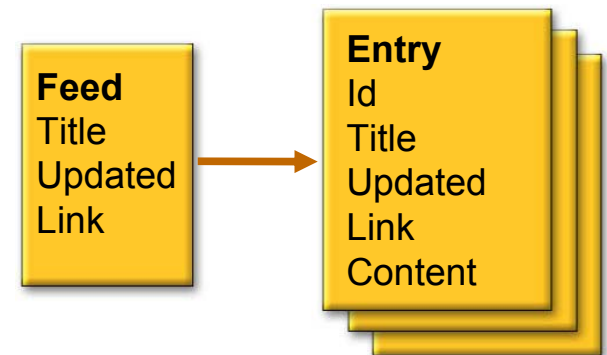
java.sun.com/javaone

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
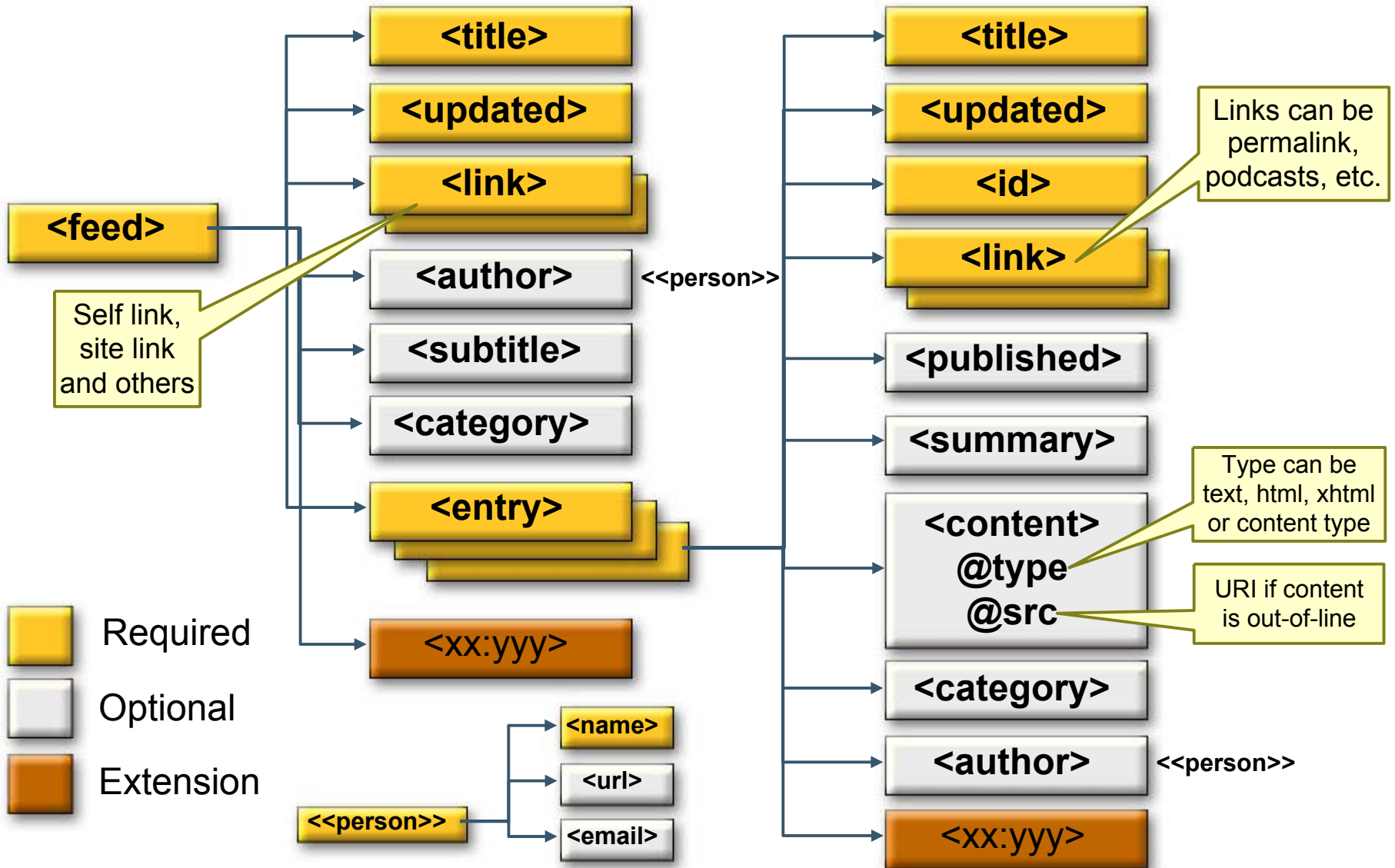
- Protocol will be finalized in 2007

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
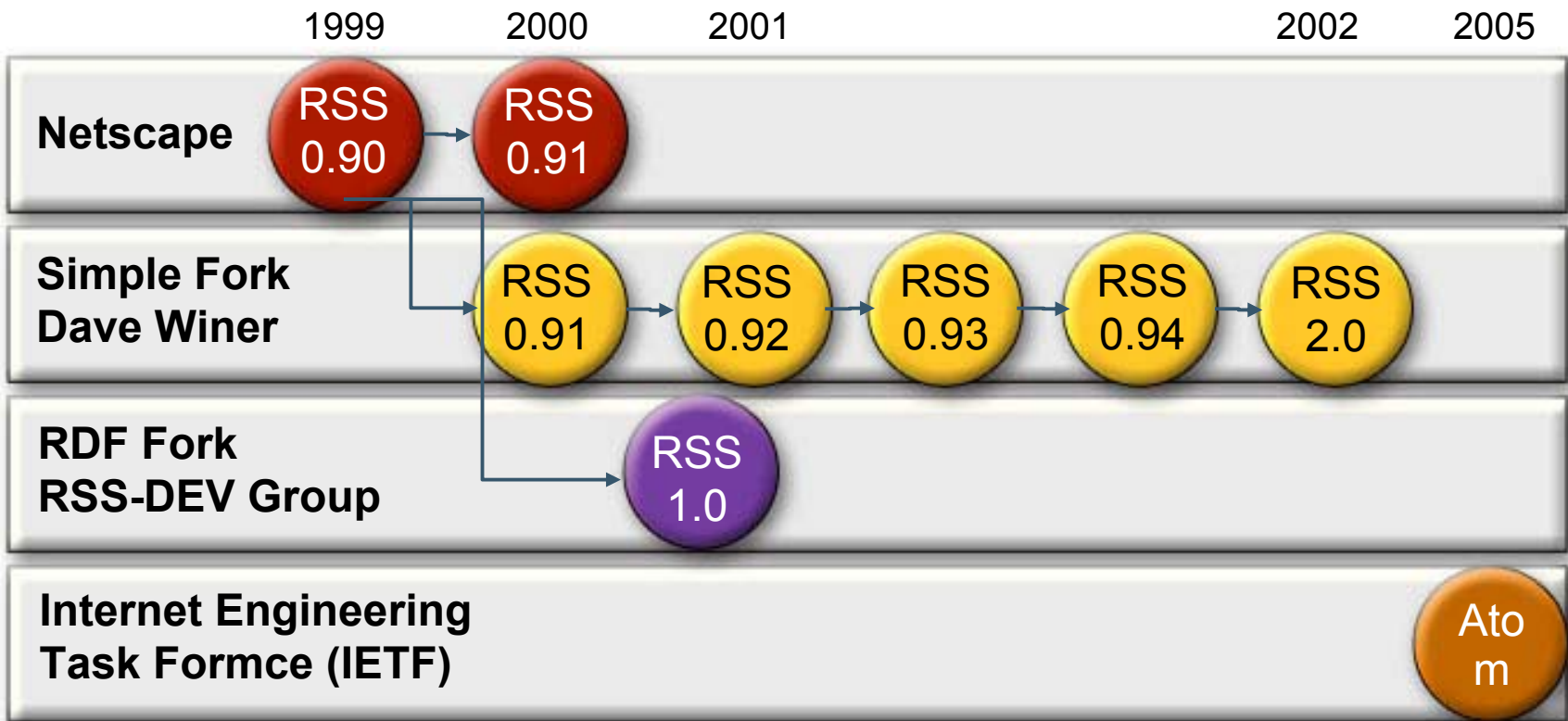- It's generic, not just for blogs

Feed
Title
Updated
Link

Entry
Id
Title
Updated
Link
Content

Elements of Atom (Abridged)

<feed>
- <title>
- <updated>
- <link> — Self link, site link and others
- <author> <<person>>
- <subtitle>
- <category>
- <entry>
- <xx:yyy>

<entry>
- <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>

Required

Optional

Extension

<<person>>
- <name>
- <url>
- <email>

java.sun.com/javaone

<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

	1999	2000	2001		2002	2005
Netscape	RSS 0.90	RSS 0.91				
Simple Fork Dave Winer		RSS 0.91	RSS 0.92	RSS 0.93	RSS 0.94	RSS 2.0
RDF Fork RSS-DEV Group		RSS 1.0				
Internet Engineering Task Formce (IETF)						Atom

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™ platform)**
  - **Abdera: STAX-based *Atom-only* parser (Java platform)**
  - **Universal Feed Parser (Python)**
  - **Windows RSS Platform: Parser built in to IE7**

java.sun.com/javaone

ROME RSS/Atom Feed Util

- Most capable Java platform-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)

java.sun.com/javaone

How Does ROME Work?

SyndFeed

SyndEntry

SyndFeed model

Convert

Convert

Channel

Item

RSS model

Feed

Entry

Atom model

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
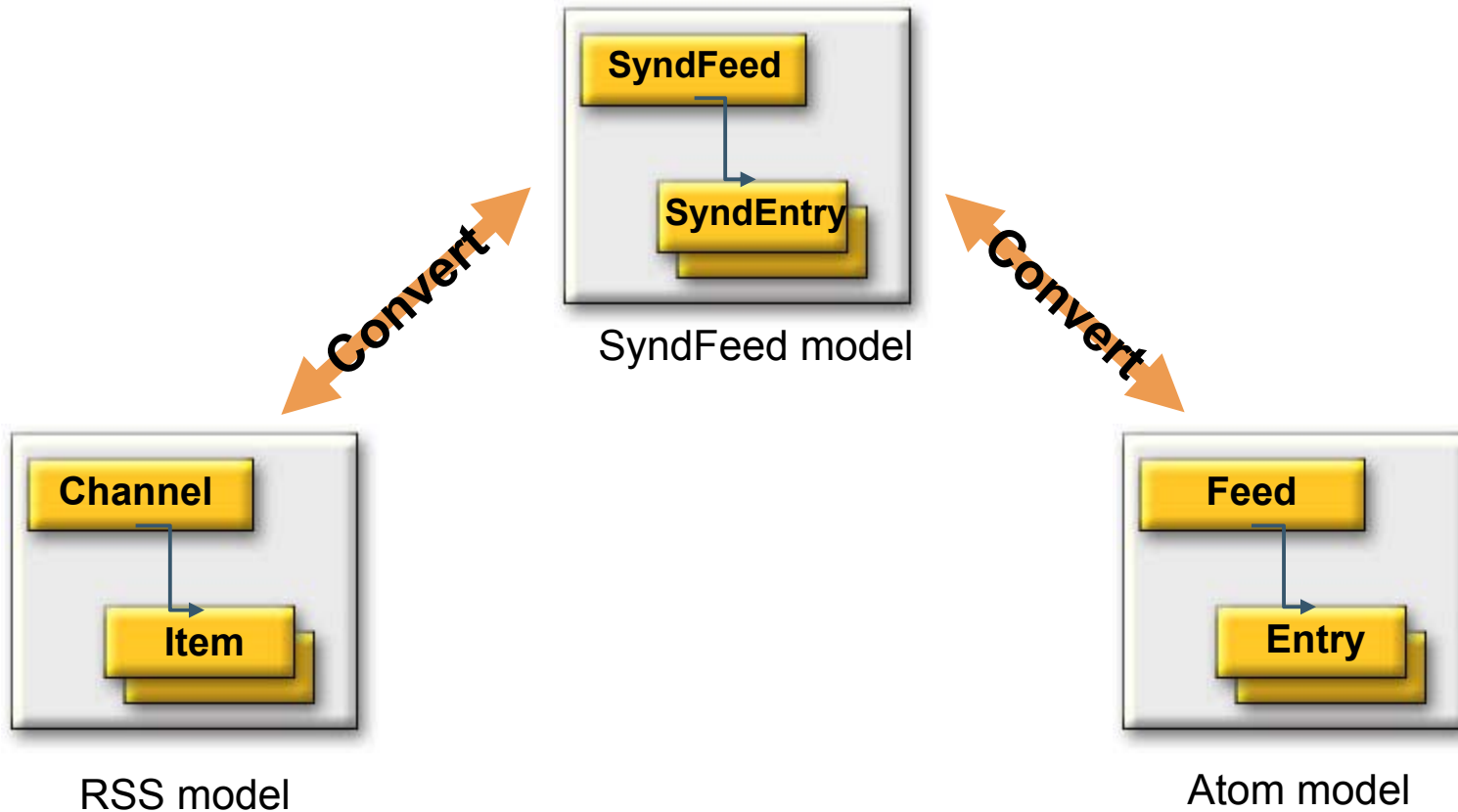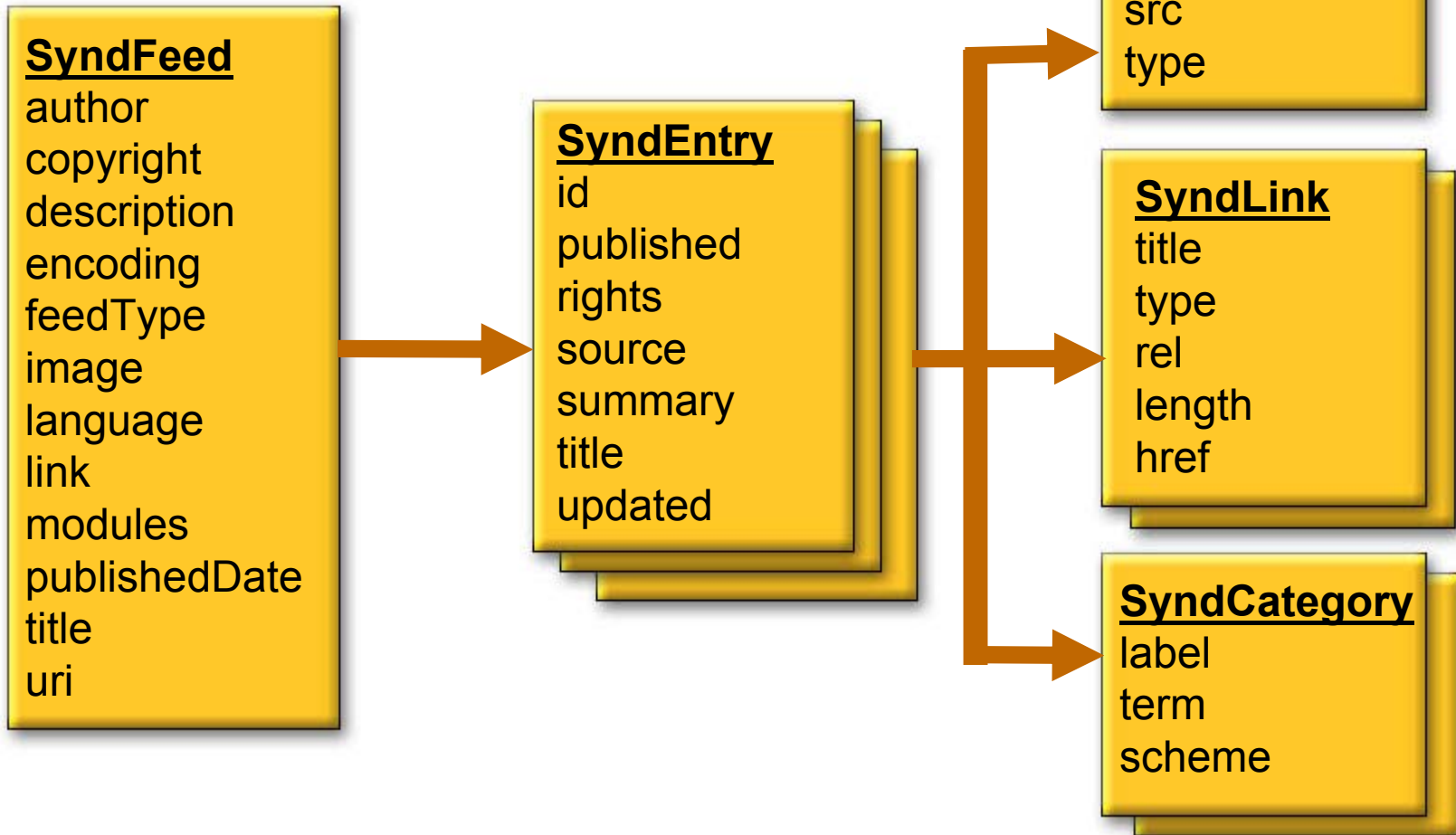
Parsing a Feed With ROME SyndFeed

```java
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

java.sun.com/javaone

Fetching a Feed With ROME Fetcher

```java
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…
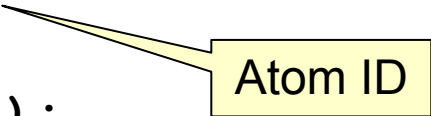
java.sun.com/javaone

Serving Feeds: Generate XML

- Use your favorite XML tools or…

- Templates languages like JavaServer Pages™ (JSP) technology, PHP, ASP.Net

- Or better yet: a feed toolkit like ROME

Generating Atom With ROME, Pt. 1/2

```java
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

```java
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

java.sun.com/javaone

Serving Atom With ROME, Pt. 1/2

```java
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

```java
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

- Safari can too

- And even IE

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" />

. . .
```
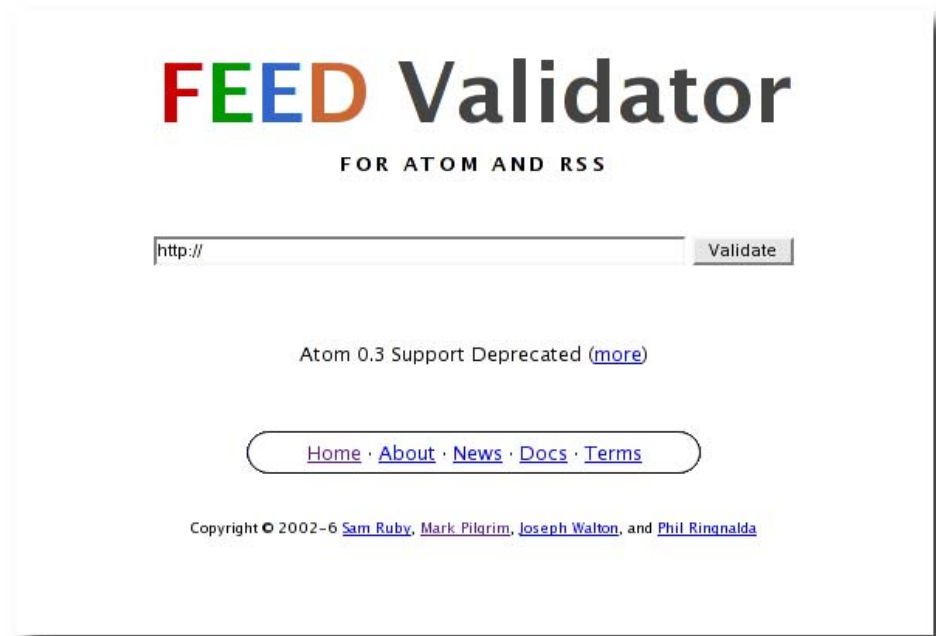
Serving Valid Feeds

- Ensure HTML is properly escaped
- Ensure XML is well formed
- Validate!
- feedvalidator.org

FEED Validator

FOR ATOM AND RSS

| http:// | Validate |

Atom 0.3 Support Deprecated (more)

Home · About · News · Docs · Terms

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…

java.sun.com/javaone

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

java.sun.com/javaone

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, and delete*
- *Based on principals of REST*

java.sun.com/javaone

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
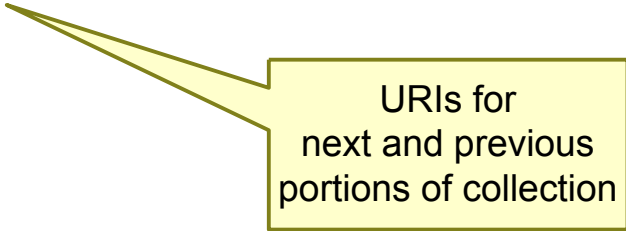- HTTP DELETE to delete entries

APP Introspection

client — GET from endpoint URI → server

server — Service document → client
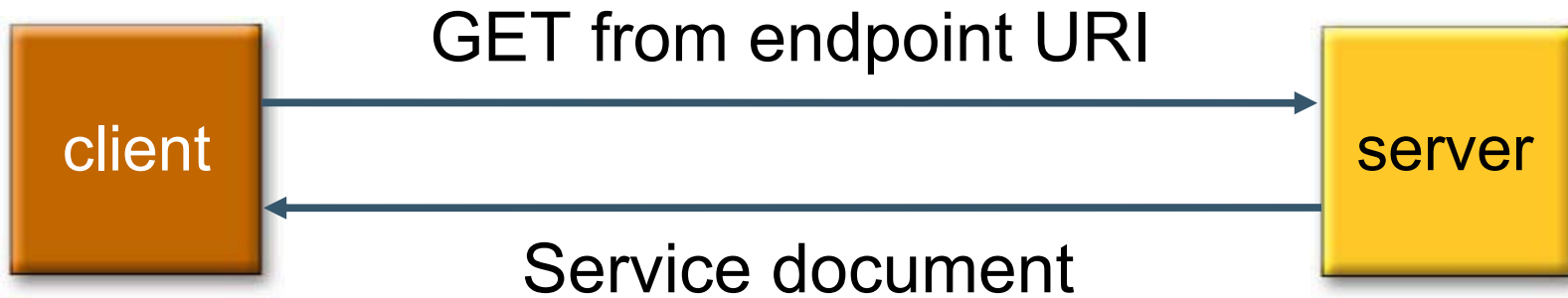
APP Introspection Document

```xml
<?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>
```
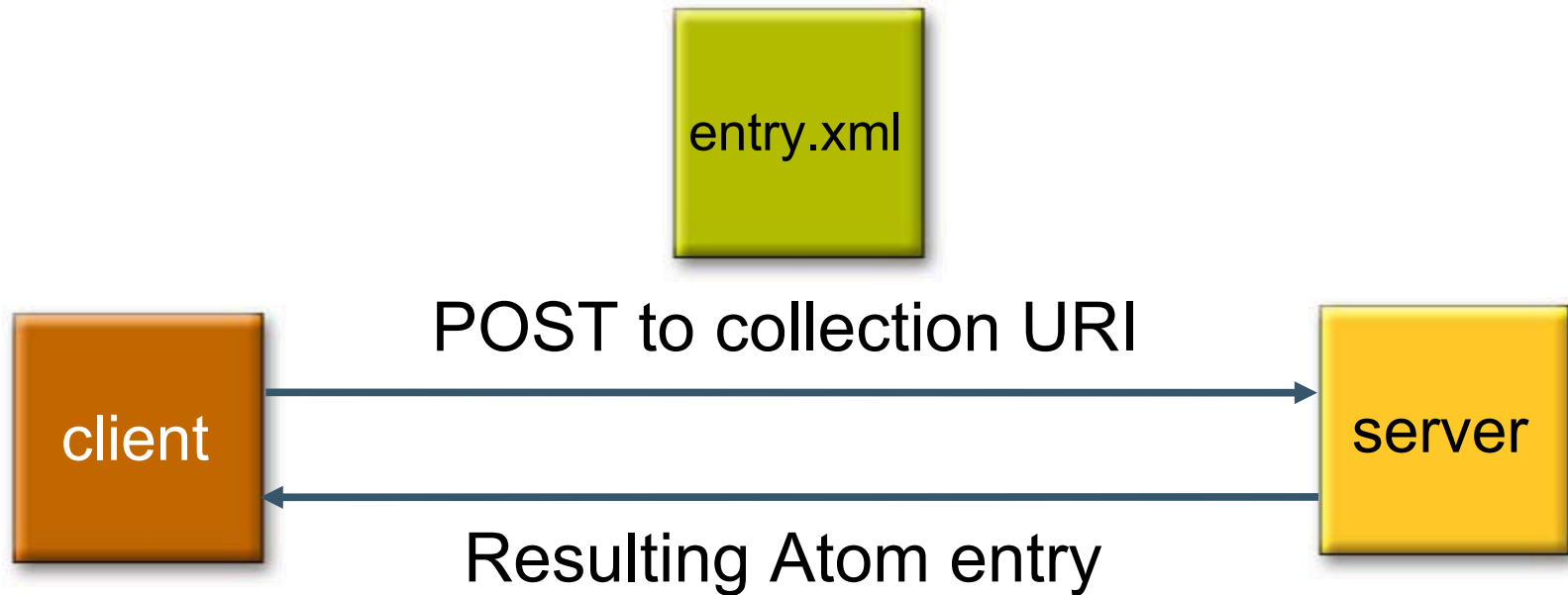
URIs for
next and previous
portions of collection

Creating an Entry

entry.xml

POST to collection URI

client

server

Resulting Atom entry

`<entry>` in a Collection

```
<entry>
  <title>NPE on new order query</title>
  <link rel="alternate"
     href="http://bugtracker/bugreport?i =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>
```
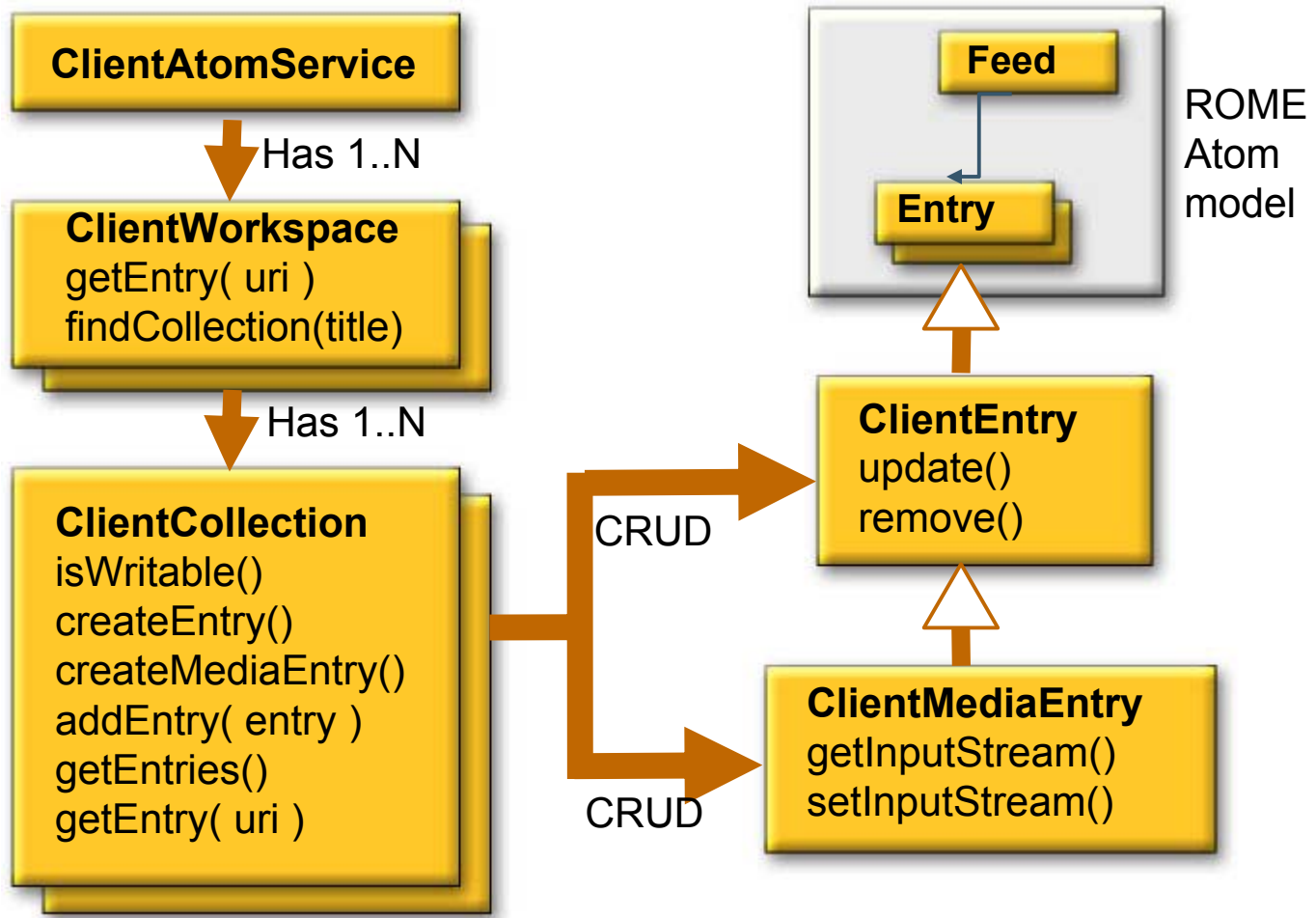
Edit URI for entry

java.sun.com/javaone

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 web app
- Blog Client Library
  - Suports both MetaWeblog API and APP
  - Blog centric and not as generic as APP Client Library

java.sun.com/javaone

ROME Propono: Atom Client API

ClientAtomService

↓ Has 1..N

ClientWorkspace
getEntry(uri)
findCollection(title)

↓ Has 1..N

ClientCollection
isWritable()
createEntry()
createMediaEntry()
addEntry(entry)
getEntries()
getEntry(uri)

→ CRUD → **ClientEntry**
update()
remove()

→ CRUD → **ClientMediaEntry**
getInputStream()
setInputStream()

Feed

Entry

ROME
Atom
model

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
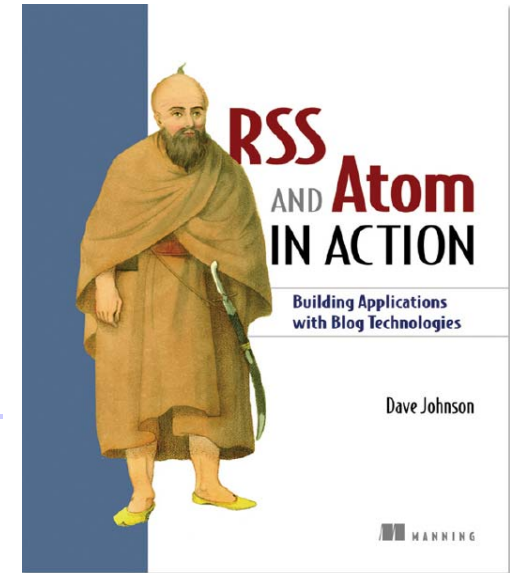
Publishing with ROME Propono

The future…

java.sun.com/javaone

RSS/Atom Trends

- **Better RSS/Atom support in Java platform**
  - Thanks to ROME and Abdera; time for a Java™ Specification Request?

- **More REST-based web services in general**
  - Made easy by REST API, Restlets, 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
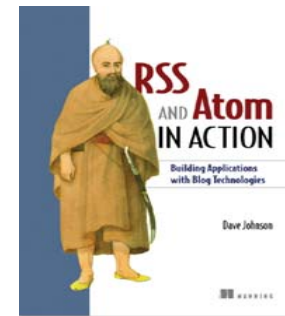 - http://manning.com/dmjohnson

java.sun.com/javaone

Summary

- RSS and Atom: not just for blogs anymore

- Feeds should be part of every developers toolkit

- 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

Beyond Blogging: Feeds in Action

Dave Johnson

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

TS-6029

java.sun.com/javaone