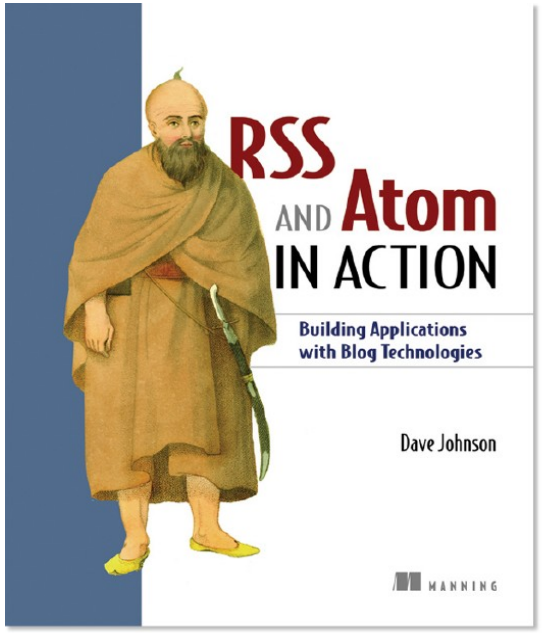
Beyond blogging:
Understanding feeds and publishing protocols

- Dave Johnson
  - Staff Engineer S/W
  - Sun Microsystems, Inc.
  - http://rollerweblogger.org/page/roller



Beyond blogging: Goals

Understand Atom (and RSS) feed formats and the Atom Publishing Protocol

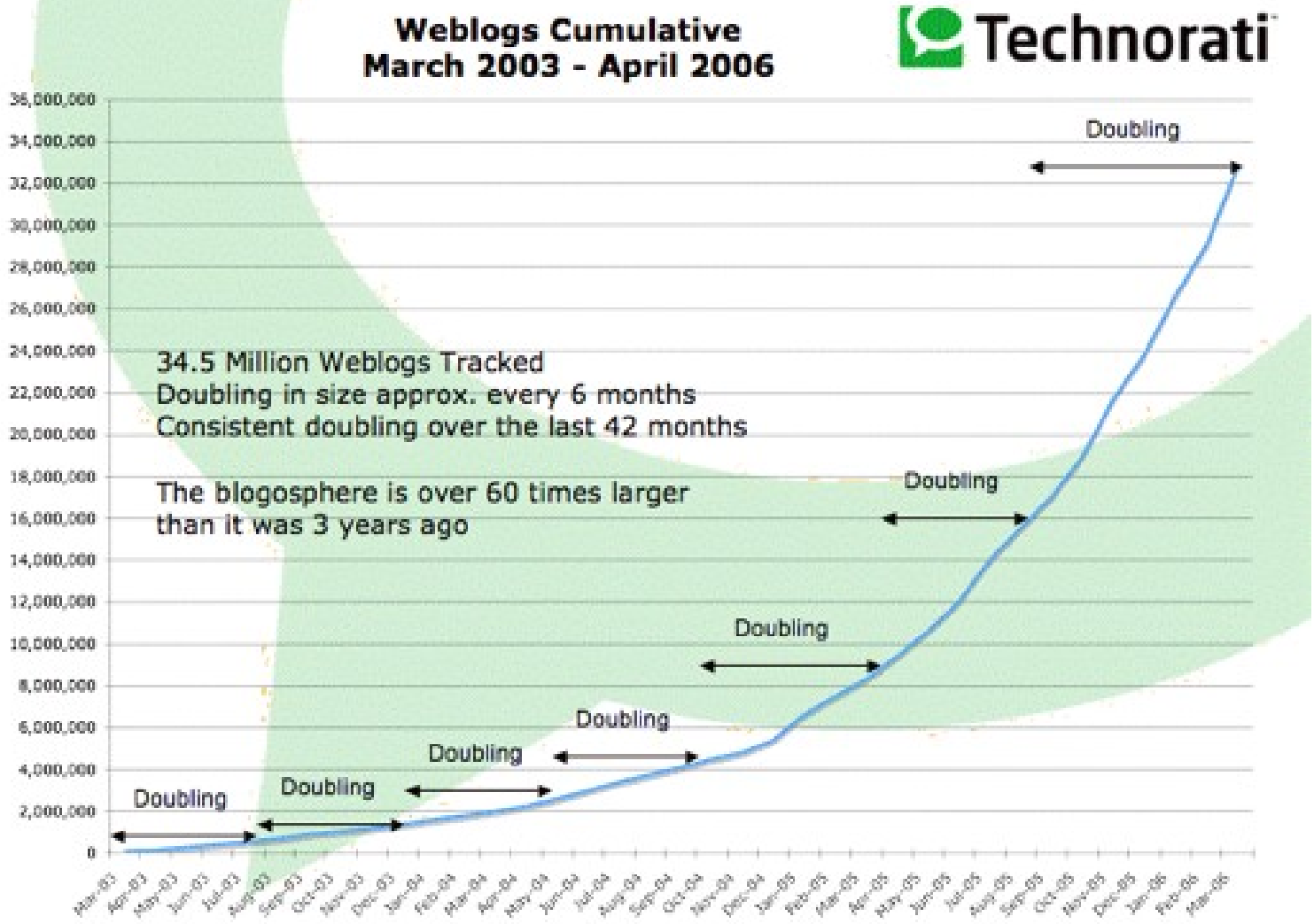
Why talk about blogging?

- Blogs made the web easier
- For writers, readers and *software developers*
- A blog is a feed-enabled, programmable *instantaneous world-wide publishing system*
- With interop provided by HTTP and **XML**

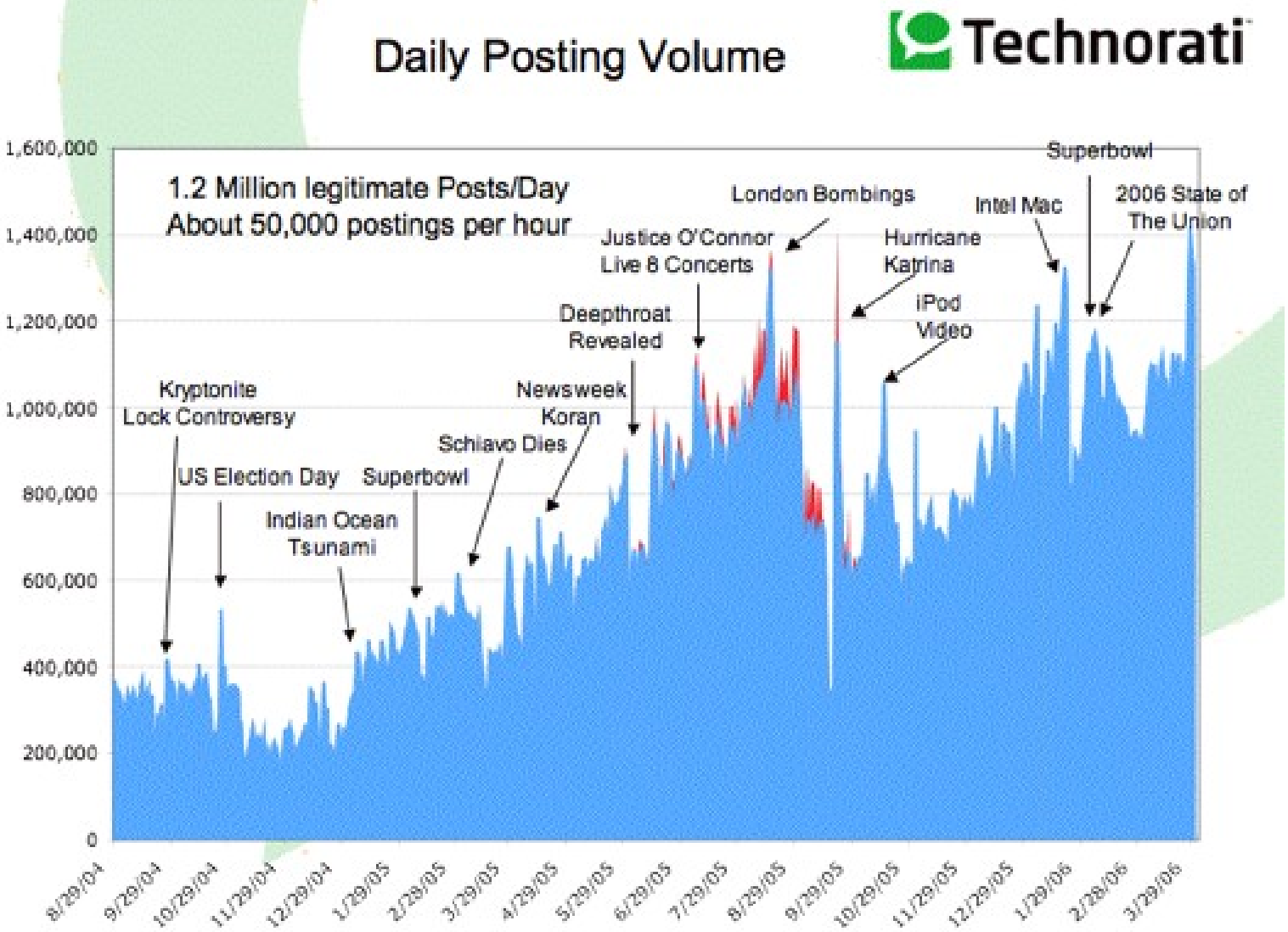
Bloggers didn't invent XML

- 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...

Technorati's state of the blogosphere



Technorati's state of the blogosphere



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 feeds
  - Every web site has an API
- Bloggy?

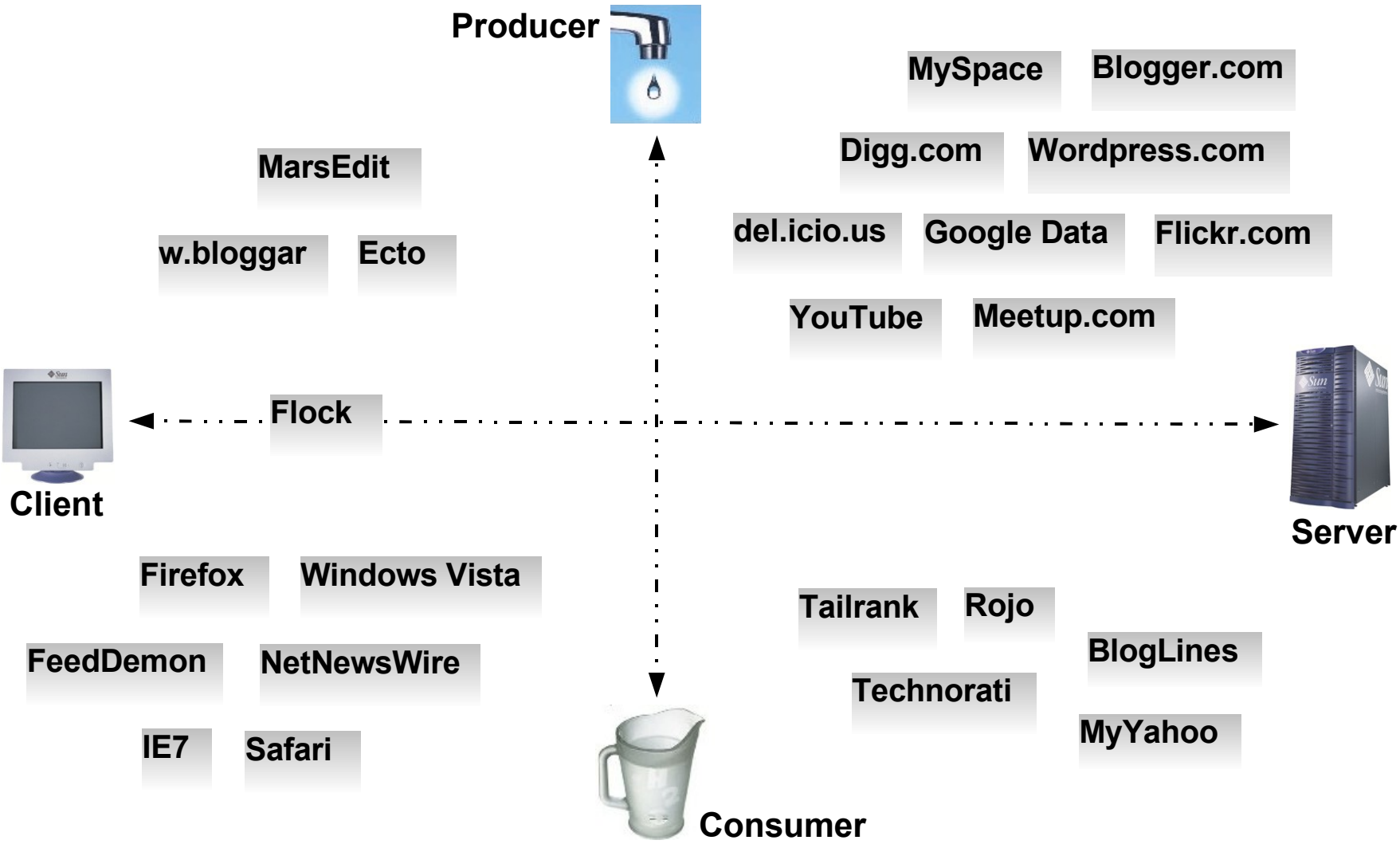
And the web is bloggy

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

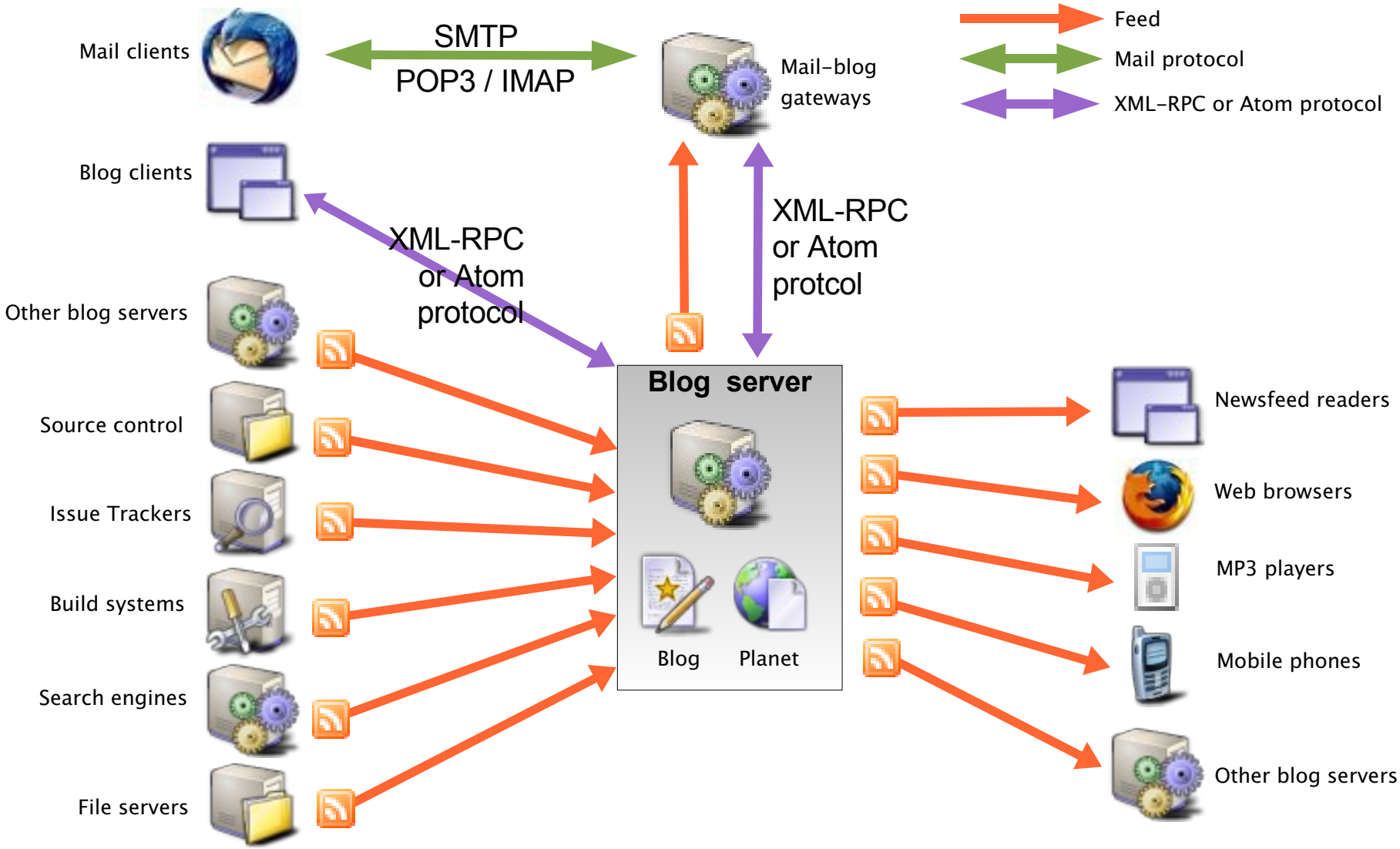
- News stories
- Search results
- Uploaded photos
- Events and meetups
- Podcasts and Vodcasts
- Bug reports
- Wiki changes
- Source code changes
- O/S log messages

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

Feeds in Web 2.0 applications

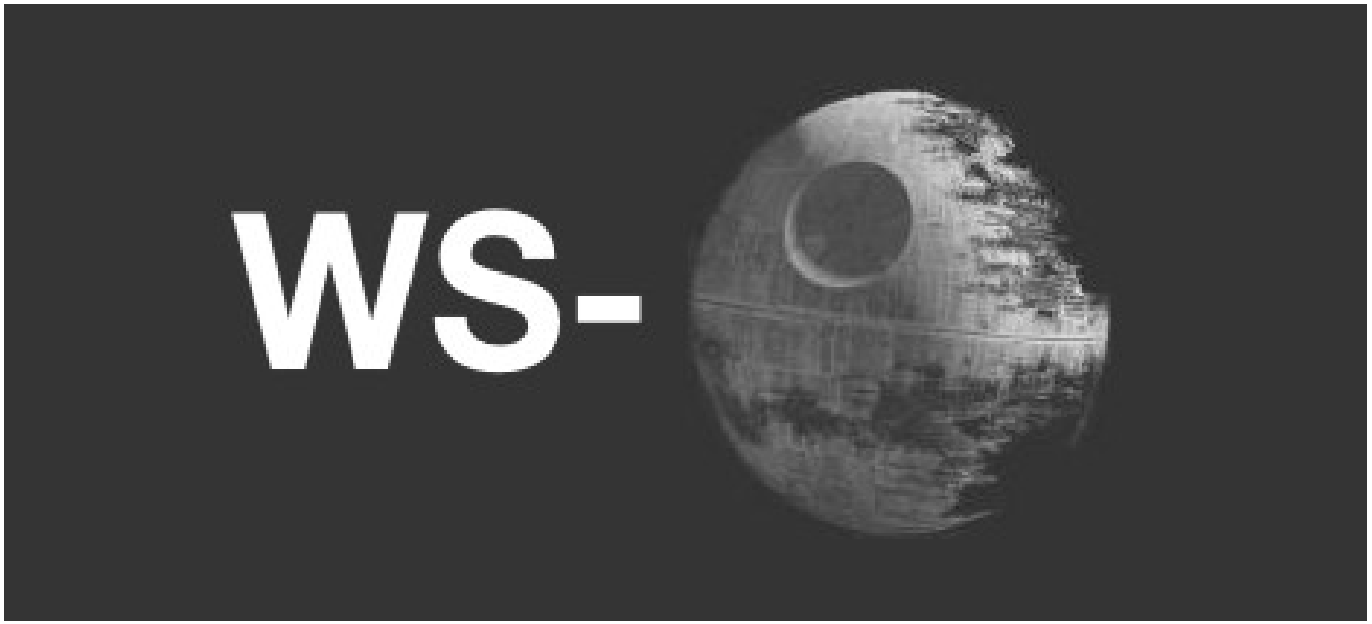


Feeds in enterprise blogging



Meanwhile: web services got uppity

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



Tim Bray's WS-* Page Count

Security	230 pages
Reliable messaging	21 pages
Transactions	39 pages
Metadata	111 pages
Messaging	211 pages
Management	23 pages
Business process	74 pages
Specification profiles	74 pages
Total	**783 pages***

** and that's not counting XML and XML schema specifications, add 599 pages*

But most of us didn't follow

- Developers prefer REST and POX over HTTP
  - *"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), POX is a much better solution than WS-*."*

  -- Anne Thomas Mannes

And now Atom is emerging

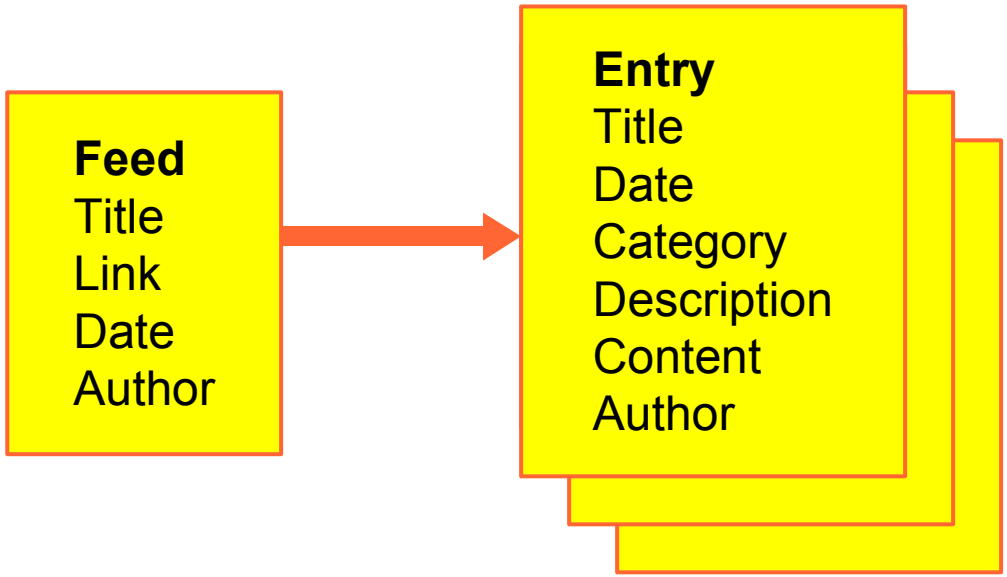
- A new foundation for REST based web services
- e.g. Google Data API
  - Atom protocol
  - Atom and RSS formats
  - A9 Open Search for filtering
- e.g. Lucene Web Services
  - Atom protocol for managing index entries
- Let's return to the topic of feeds

Beyond blogging

Understanding feeds

What Is a Feed?

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



The birth of the RSS feed format

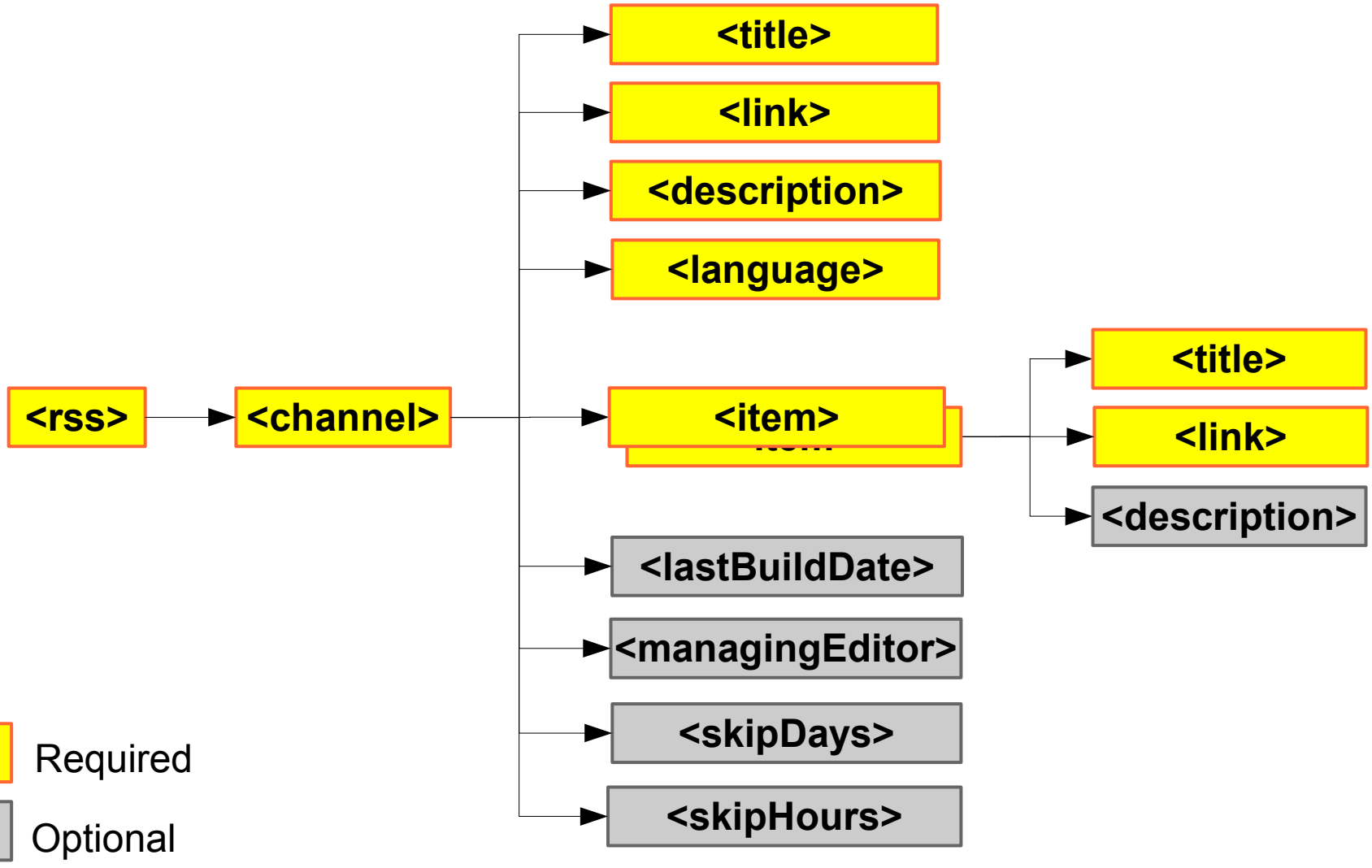
- RSS began life at Netscape

- First spec RSS 0.90 was authored by Dan Libby
- Created for the My Netscape portal
- Known as RDF Site Summary (RSS)

- Version 0.91 dropped RDF
  - Dan Libby released RSS 0.91 spec
  - Dave Winer released his own RSS 0.91 spec

- The 0.9X formats are obsolete but still in use today

Elements of RSS 0.91



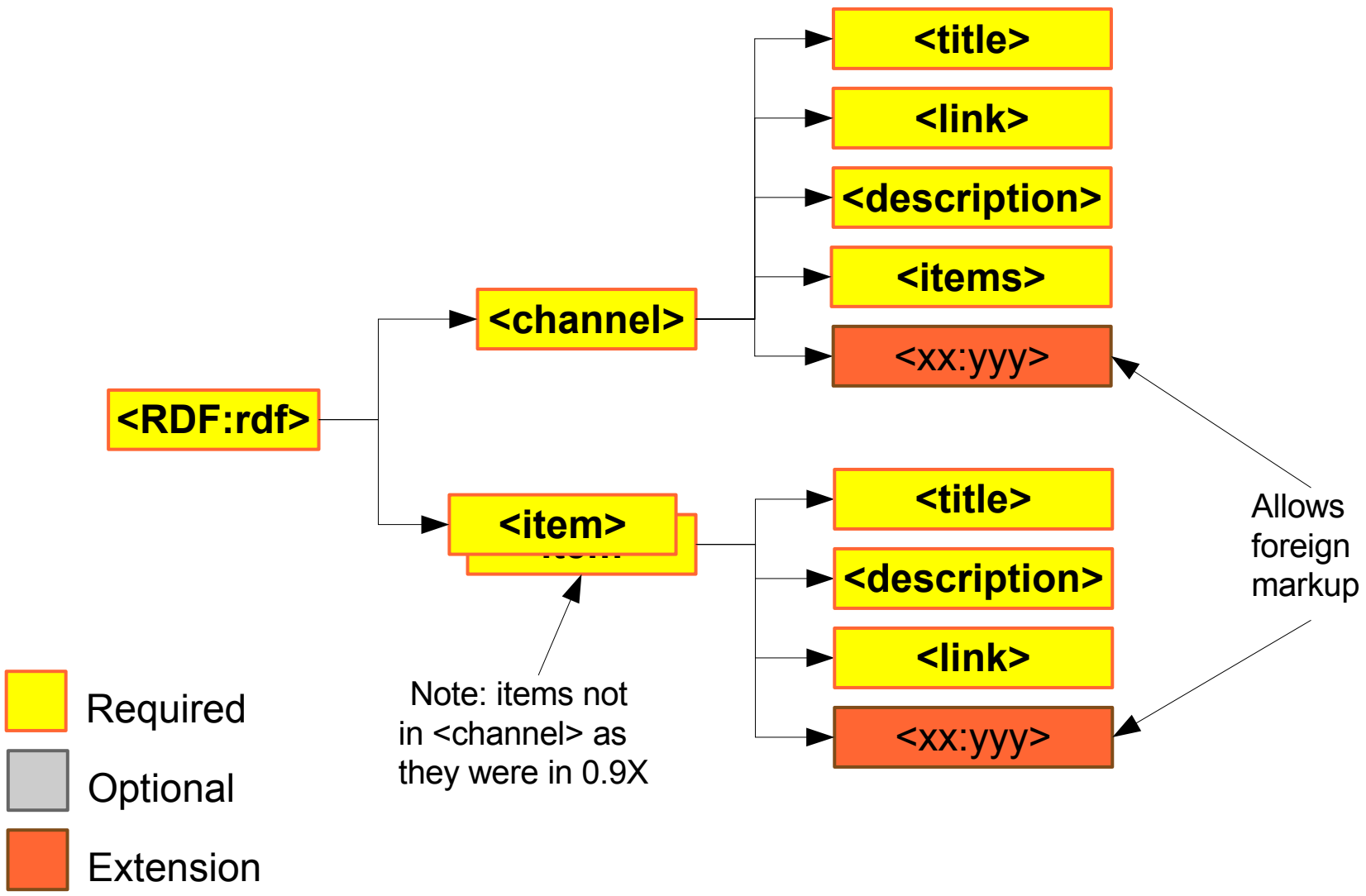
The RDF fork: RSS 1.0

- After RSS 0.91...
- Dave Winer argued for keeping RSS simple
- RDF advocates argued for adding RDF back in

- The RDF side declared victory and released 1.0
  - Based on RDF, incompatible with RSS 0.91
  - 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)



RSS 1.0 extension modules

- You can add your own XML elements
- As long as they are properly name-spaced

- Allows use of Dublin Core elements

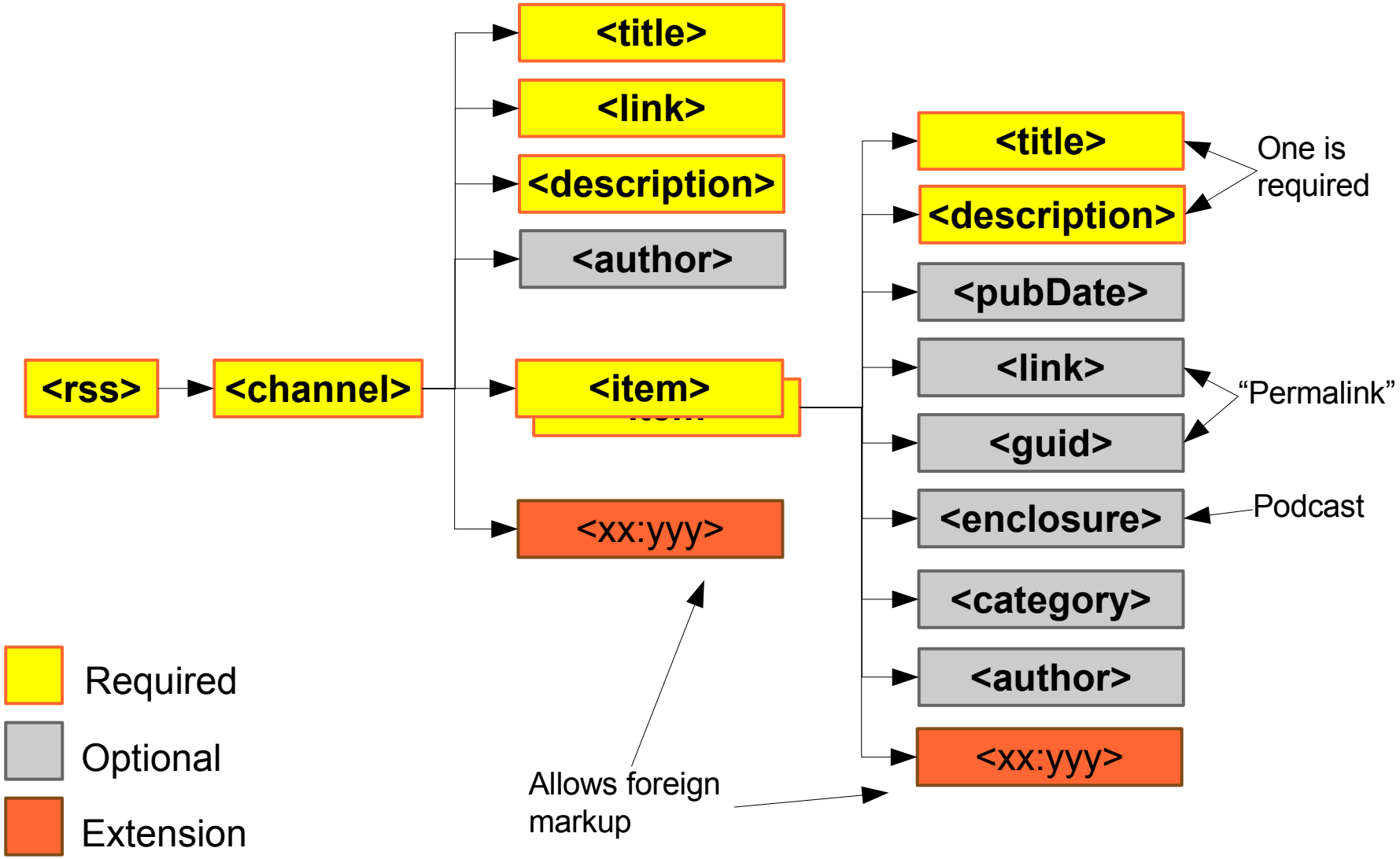
- Modules exist for for Syndication, Annotation, Taxonomy, iTunes, GeoRSS, Slashdot etc.

- Can be used w/other feed formats too (i.e. Atom and RSS 2.0 now support extensions too)

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 2.0 Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0">
<channel>
<title>Example Blog</title>
<link>http://example.com/blog</link>
<item>
   <title>Hello World!</title>
   <description>
      Welcome to &lt;b&gt;my blog&lt;b&gt;.
   </description>
   <pubDate>Wed, 20 Apr 2005 17:41:04 EDT</pubDate>
   <link>http://example.com/blog/20050420?id=132</link>
   <enclosure url="http://example.com/casts/file1.mpg"
      type="audio/mpeg3" length="13456170"/>
</item>
</rss>
```

RSS 2.0 extensions lead to Funky RSS

- Elements under **`<item>`** are optional
- Extensions elements are allowed

- So, folks use extension elements in place of standard RSS elements and thus we have **Funky RSS**

- But why?
  - Support both content and summary
  - Use author name instead of email address
  - Don't like RFC-822 dates

Funky RSS

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
<title>Example Blog</title>
<link>http://example.com/blog</link>
<item>
   <title>Hello World!</title>
   <description>Welcome to &lt;b&gt;my blog&lt;b&gt;.
   </description>
   <link>http://example.com/blog/20050420?id=132</link>
   <enclosure url="http://example.com/casts/file1.mpg"
      type="audio/mpeg3" length="13456170"/>
   <dc:date>Wed, 20 Apr 2005 17:41:04 EDT</dc:date>
   <dc:creator>Dave Johnson</dc:creator>
</item>
</rss>
```

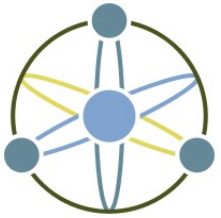
RSS limitations

- Spec is too loose and unclear
  - What fields can be escaped HTML?
  - How many enclosures are allowed per element?
- Content model is weak
  - No support for summary and content
  - Content-type and escaping not specified
- Spec can't be clarified
  - RSS Board not allowed to clarify specification

Beyond blogging

Atom

What is Atom?



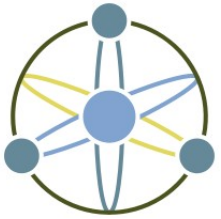
- From the IETF Atom WG charter:

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

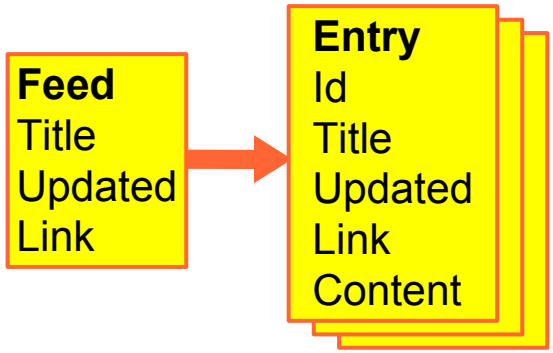
- Feed format is now IETF RFC-4287

- Protocol will be finalized in 2006

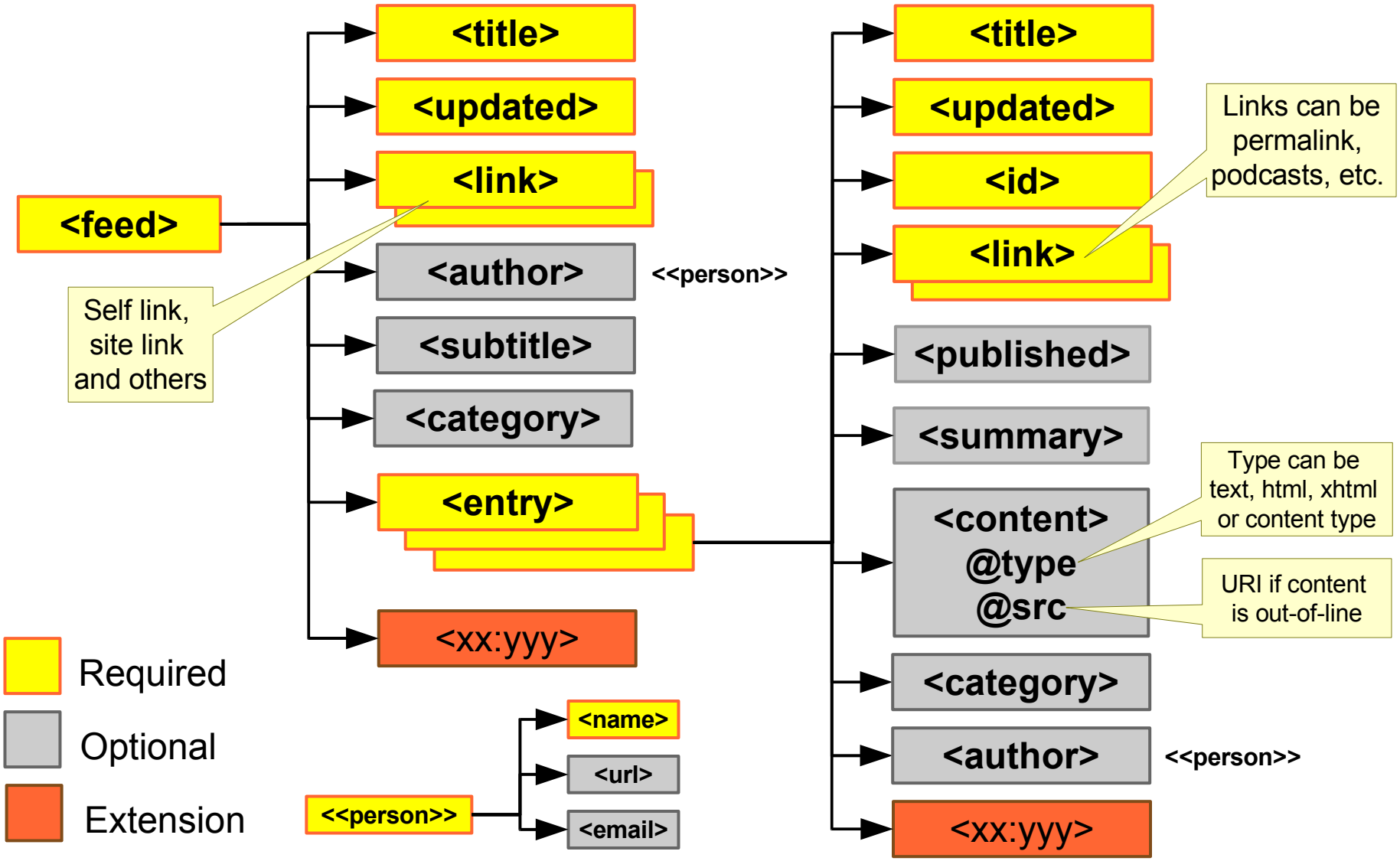
Atom Publishing Format



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



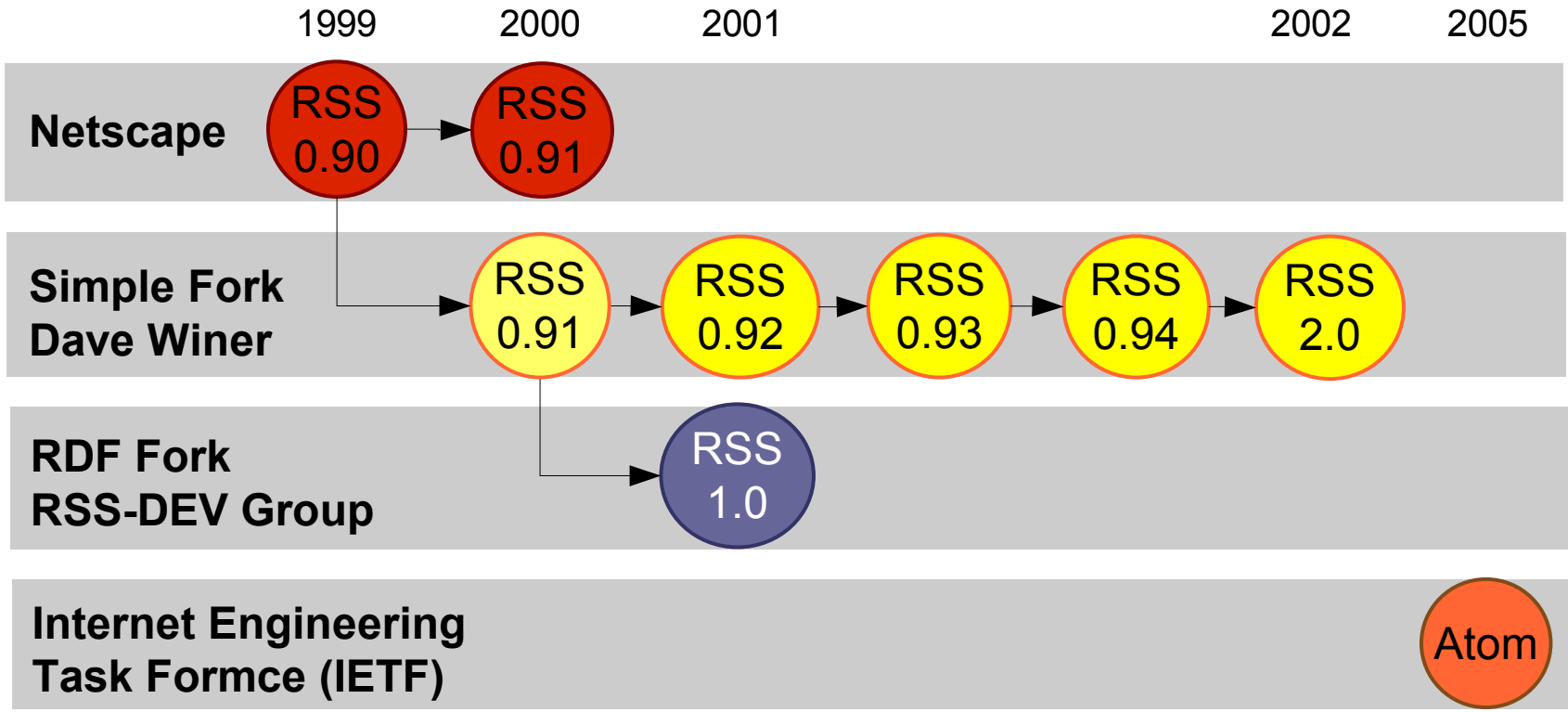
Elements of Atom (abridged)



Atom <feed> with one <entry>

```
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns='http://purl.org/atom/ns#' xml:lang='en-us'>
   <title>Oh no, Mr. Bill</title>
   <link href='http://nbc.com/sluggo/' />
   <link rel='self' href='http://nbc.com/sluggo/index.atom' />
   <updated>2005-04-06T20:25:05-08:00</updated>
   <author><name>Mr. Bill</name></author>
   <entry>
      <title>A post about stuff</title>
      <link href='http://nbc.com/sluggo/20050420?id=321' />
      <id>http://nbc.com/sluggo/20050420?id=321</id>
      <updated>2005-04-06T20:25:05-08:00</updated>
      <content type='xhtml'>
         <!-- xhtml content -->
      </content>
   </entry>
</feed>
```

RSS and Atom feed family feud



Beyond blogging

Parsing feeds

Parsing RSS and Atom newsfeeds

- Use your favorite XML parsing technique
- Better yet, use a parser library
  - **Universal Feed Parser** (Python)
  - **ROME**: DOM based parser / generator (Java)
  - **Tailrank Feed Parser**: SAX based parser (Java)
  - **Windows Feeds API**: parser in IE7 and Vista
  - **Abdera**: STAX based Atom parser (Java)

Fetching newsfeeds

- Be nice and conserve bandwidth
  - Don't poll too often
  - Use HTTP conditional GET or ETags
  - Obey provider skip hours / days settings
- Your parser library might do the work for you
  - ROME's Fetcher provides a caching feed-store
  - Windows RSS does too

Universal Feed Parser

- In Python, but it's the best in any language

- Parses all forms of RSS and Atom
- *Ultra-liberal*: Parses anything, even invalid XML
  - Uses an SGML parser, falls back to regex
- Parses to simple hash-table of values

- Supported by thousands of test cases
- Free and open source (GPL licence)
- Widely used

Universal Feed Parser example


```
import feedparser
import sys

feed = feedparser.parse(url)

for item in feed["items"]:
   print "Title: " + item["title"];
   print "Link: " + item["link"];
   print "Date: " + item["date"];
   print "Desc: " + item["description"] + "\n";
```

ROME Feed Utilities

- Most capable Java based toolkit

- Parses and generates all forms of RSS and Atom
- Highly pluggable/extensible, based on JDOM
- Parses to Atom, RSS or abstract object model

- Free and open source

ROME example

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

Iterator entries =
  feed.getEntries().iterator();
while (entries.hasNext()) {
  SyndEntry entry = (SyndEntry)entry.next();
  System.out.println("Title: " + entry.getTitle());
  System.out.println("Link: " + entry.getLink());
  System.out.println("Date: " + entry.getPublishedDate());
  System.out.println("Desc: " + entry.getDescription());
  System.out.println("\n");
}
```

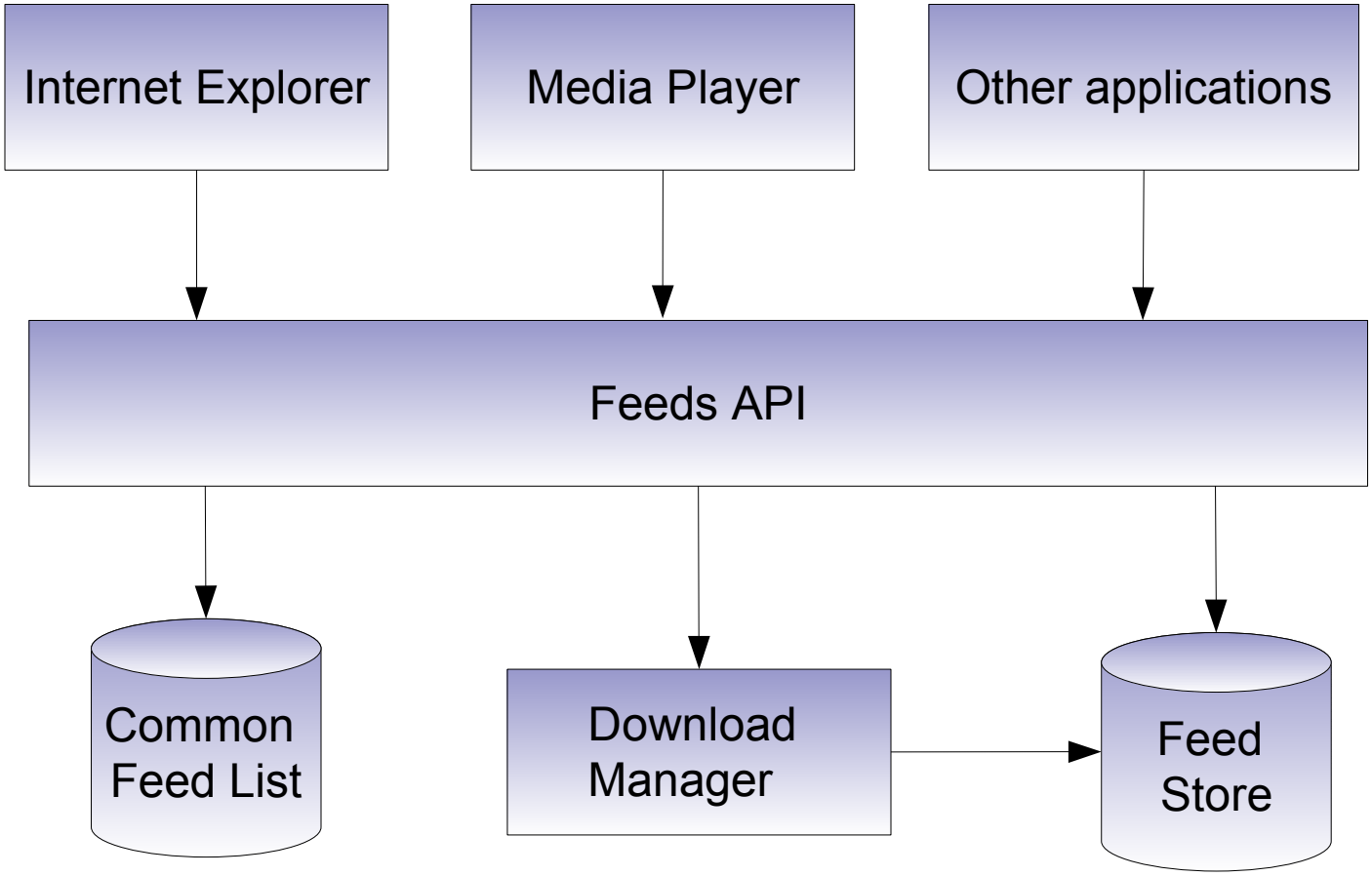
Windows RSS Platform

- Feeds API included in IE7 and Windows Vista
- Parses all forms of RSS and Atom

- Manages user subscription list
  - Via per-user feed subscription list / feed store
  - Manages read / unread state of all feed items
  - All applications have access to user's feeds

- No support for feed generation
- Callable from .NET and un-managed code

Windows RSS platform



Windows RSS platform example


```
IFeedsManager fm = new FeedsManagerClass();

IFeed feed = null;
if (!fm.IsSubscribed(url)) {
  IFeedFolder rootFolder =
     (IfeedFolder)fm.RootFolder;
  feed = (IFeed)rootFolder.CreateFeed(url, url);
} else {
  feed = (IFeed)fm.GetFeedByUrl(url);
}
feed.Download();

foreach (IFeedItem item in (IFeedsEnum)feed.Items) {
  Console.Out.WriteLine("item.Title:  " + item.Title);
  Console.Out.WriteLine("item.pubDate:" + item.PubDate);
  Console.Out.WriteLine("item.Desc: " + item.Description);
}
```

MS Common Feed Format

- Good old embrace and extend...

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0"
   xmlns:atom="http://www.w3.org/2005/Atom"
   xmlns:cf="http://www.microsoft.com/schemas/rss/core/2005"
   xmlns:dc="http://purl.org/dc/elements/1.1/" >

<channel>
   <title cf:type="text">funky-both</title>
   <description cf:type="text">
      Core RSS elements vs. funky RSS</description>
   <language>en-us</language>
   <copyright cf:type="text">Copyright 2006</copyright>
   <lastBuildDate>
      Mon, 20 Mar 2006 20:43:55 GMT</lastBuildDate>
   <link>http://example.com/blog/link1</link>
```

MS Common Feed Format (continued)

```
    <item>
        <title cf:type="text">Item title</title>
        <author>David M Johnson</author>
        <atom:author><atom:name>
            David M Johnson</atom:name></atom:author>
        <pubDate> Mon, 20 Mar 2006 21:43:55 GMT</pubDate>
        <atom:summary type="html">
            RSS &lt;b&gt;description&lt;/b&gt;</atom:summary>
        <description cf:type="html">
            Funky RSS &lt;b&gt;content:encoded&lt;/b&gt;,
        </description>
        <category>Item category1</category>
        <link>http://example.com/blog/link1</link>
        <guid isPermaLink="false">
            http://example.com/blog/link1</guid>
        <cf:id>0</cf:id>
        <cf:read>false</cf:read>
    </item>
</channel>
</rss>
```

Tailrank Feed Parser

- SAX based RSS and Atom parser in Java
- Fast and lean
- Parses all RSS and Atom feed formats
- Battle-tested at Rojo.com and Tailrank.com
- An updated fork of the Jakarta Feed Parser

Apache Abdera (incubating)

- Java implementation of Atom format *and* protocol

- Donated to Apache by IBM
- STAX based parser, fast and lean
- Parses Atom 1.0 only

- Free and open source

Beyond blogging

Serving feeds

Serving feeds

- **Generate the XML**
  - Using templates:
    - PHP, JSP, Servlets, ASP.Net
  - Using a feed wrangler like ROME
  - Or your favorite XML generation technology

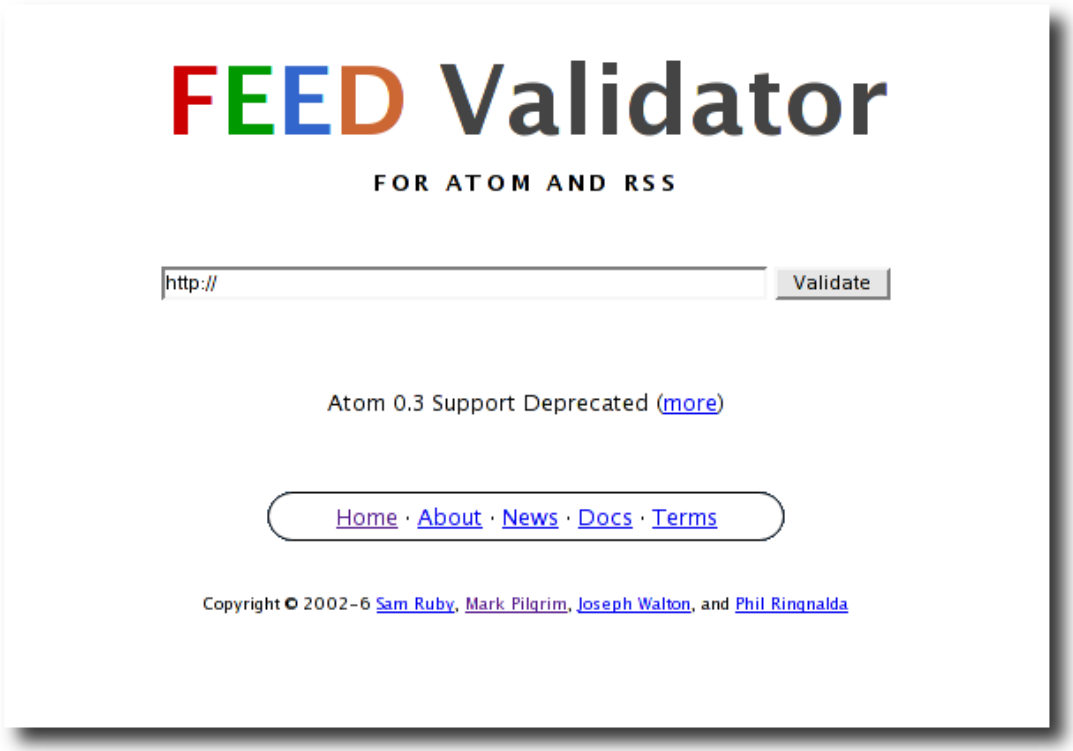
- **Serve it up**
  - Set the correct content-type

    `application/rss+xml` or `application/atom+xml`
  - Support HTTP conditional GET
  - Cache cache cache!

Serving valid feeds

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

- Validate!
- feedvalidator.org



Feed autodiscovery

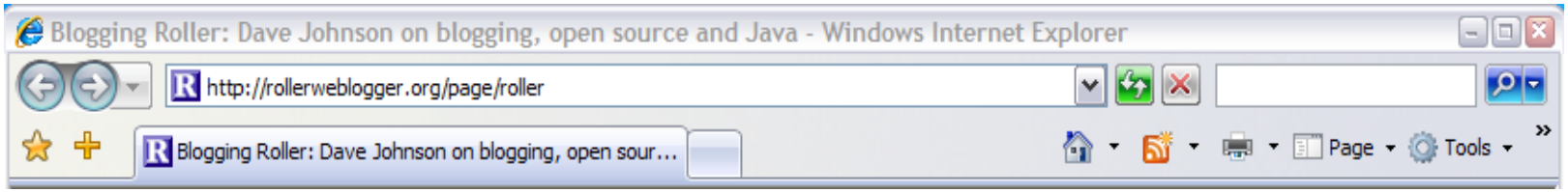
```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html" />

  <link rel="alternate"
    type="application/rss+xml" title="RSS"
    href="http://rollerweblogger.org/rss/roller" />

  <link rel="alternate"
    type="application/atom+xml" title="Atom"
    href="http://rollerweblogger.org/atom/roller" />

. . .
```

Feed autodiscovery

- Allows applications to find your feeds
- Firefox can do it

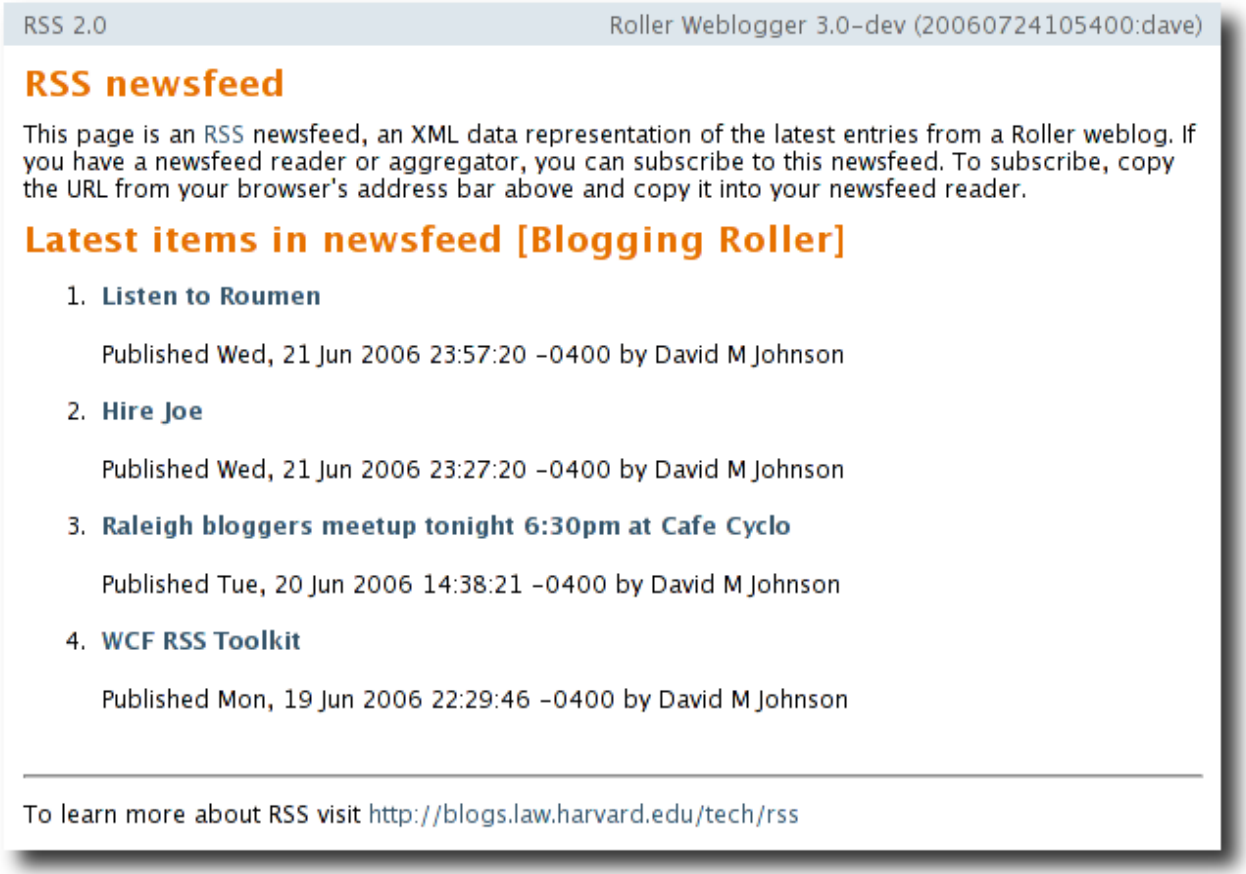
- Safari can too

- And even IE someday

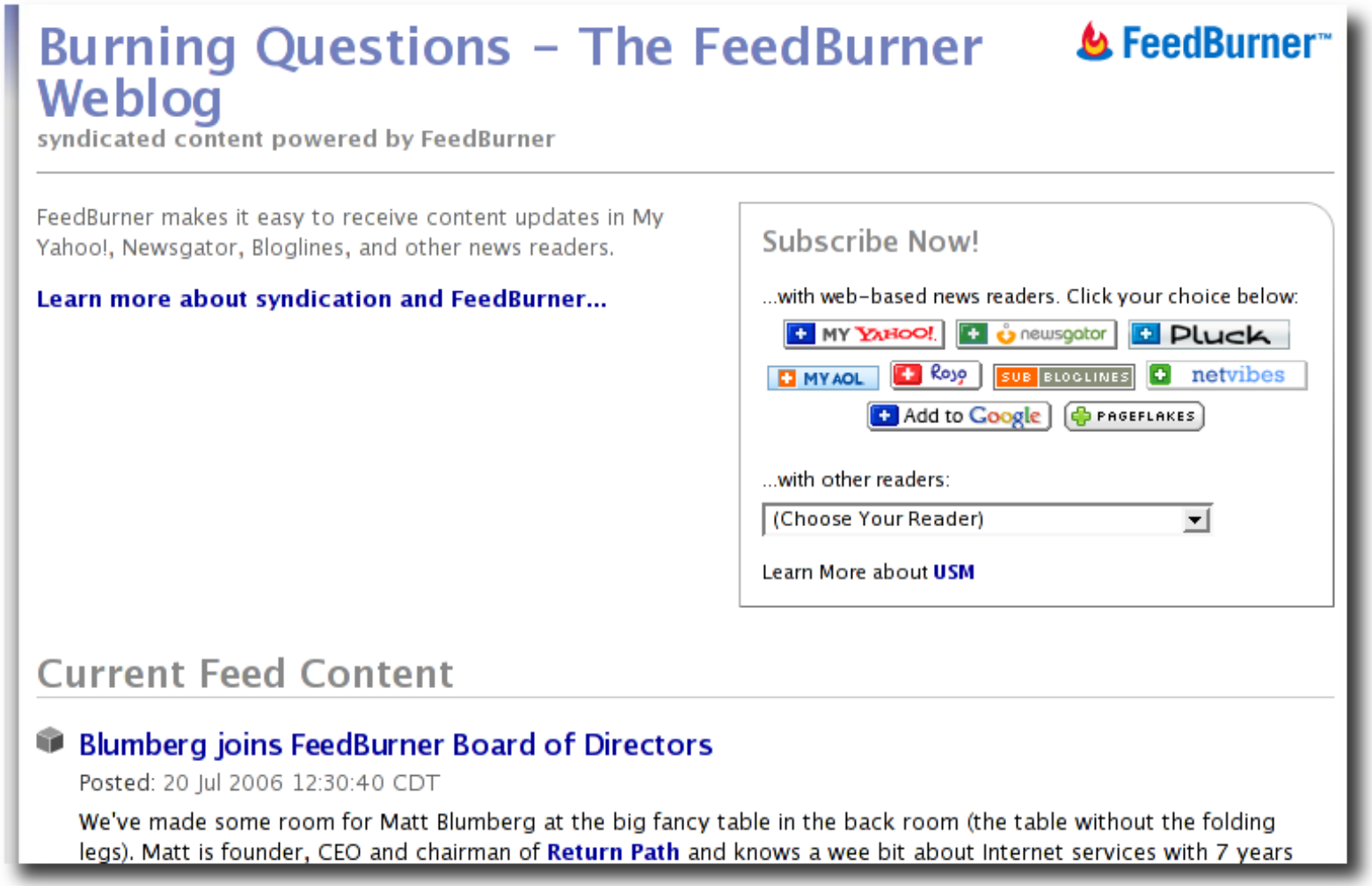


Styled feeds

- Use CSS or XSL for browser-friendly feeds that explain how to subscribe.



Feedburner's styled feed



Beyond blogging

The Atom protocol

The Atom Publishing Protocol (APP)

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

- Based on Atom Publishing Format
- Began as a replacement for the old XML-RPC based blog APIs

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

** A hash-table where (most) keys correspond to RSS element names*

What does Atom protocol do?

- Atom WG charter says protocol must enable:
  - Creating, editing, and deleting feed entries
  - Multiple authors for a feed
  - Multiple subjects or categories in a feed
  - User authentication
  - Creating, getting and setting related resources* ~~such as comments, templates, etc.~~
  - ~~Adding, editing, and deleting users~~
  - ~~Setting and getting user preferences~~

** Grey items won't be in first version of specification*

How does it do all that?

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

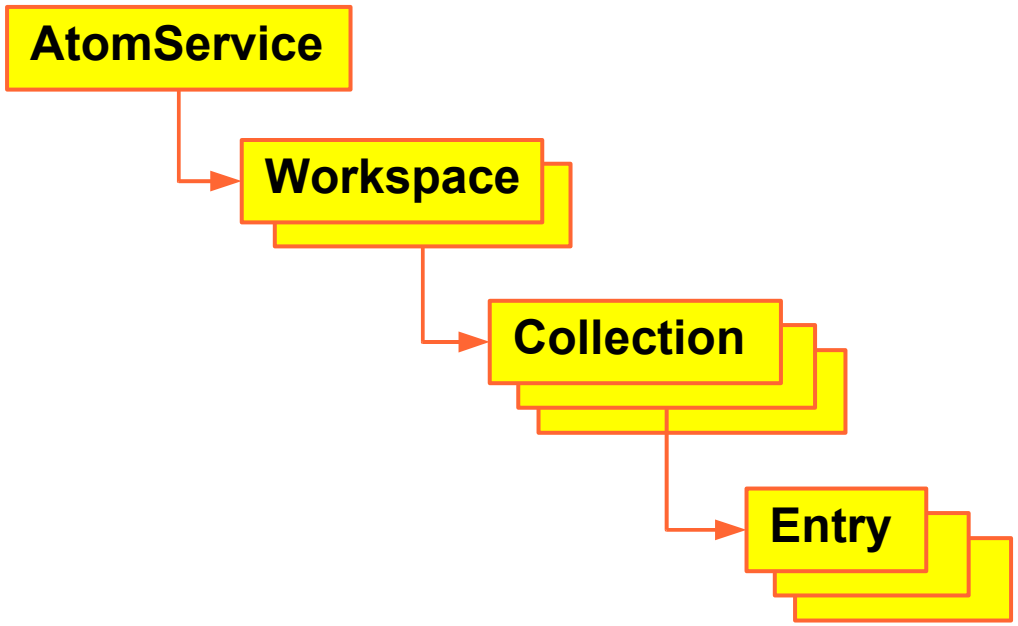
APP: everything's a resource

- Introspection document
- Collections (represented as Atom feeds)
- Entries (represented as Atom entries)
- Media files

APP: HTTP defines operations

- POST to create entries and media files
- GET to retrieve
  - Introspection document
  - Collections
  - Media files
- PUT to update entries and media files
- DELETE to delete entries and media files

APP containment



e.g. Blog user account

e.g. User's blogs

e.g. Blog's collections

e.g. Blog posts & uploads

Atom introspection document

```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://purl.org/atom/app#">
    <workspace title="My blog" >
        <collection title="My blog entries"
            href="http://example.org/reilly/main" >
            <accept>entry</accept>
        </collection>
        <collection title="Pictures"
            href="http://example.org/reilly/pic" >
            <accept>image/*</accept>
        </collection>
    </workspace>
</service>
```

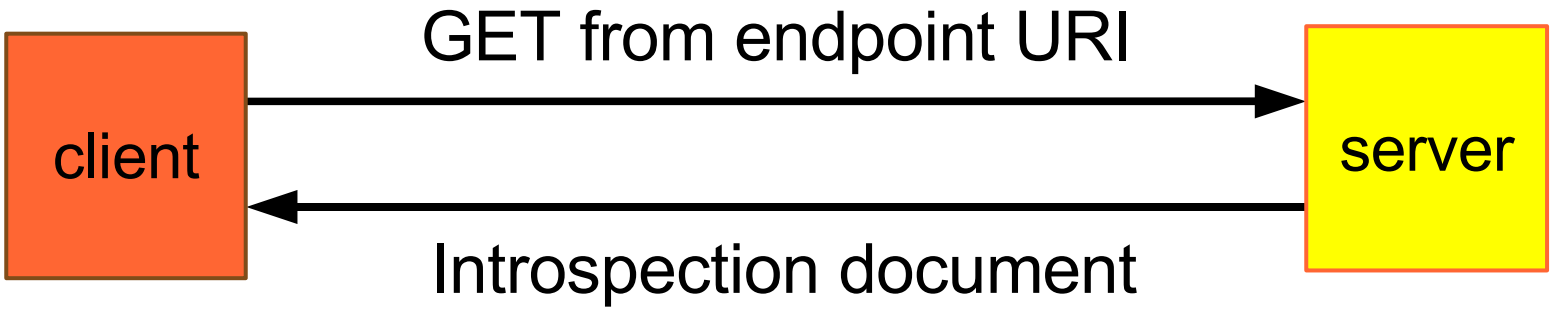
Extending the APP

- Servers can add XML elements via namespaces
- Clients that understand can benefit
- Clients that don't understand foreign markup
  - **SHOULD** preserve and return

Beyond blogging

Atom protocol in action

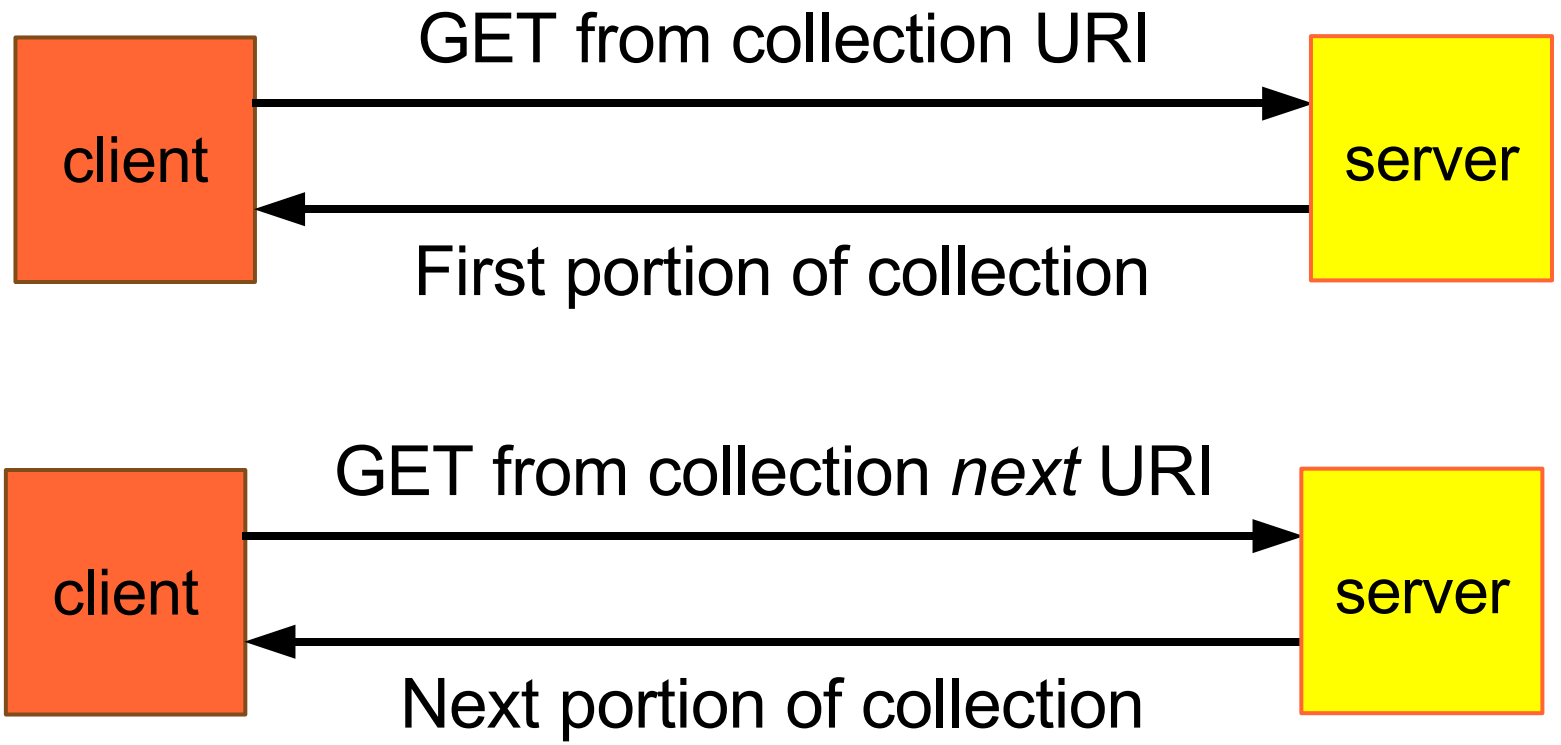
APP Introspection



APP introspection document (again)

```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://purl.org/atom/app#">
    <workspace title="My blog" >
        <collection title="My blog entries"
            href="http://example.org/reilly/main" >
            <accept>entry</accept>
        </collection>
        <collection title="Pictures"
            href="http://example.org/reilly/pic" >
            <accept>image/*</accept>
        </collection>
    </workspace>
</service>
```

Getting a collection - with paging



An Atom collection <feed>

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

  <link rel="next"
        href="http://example.org/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

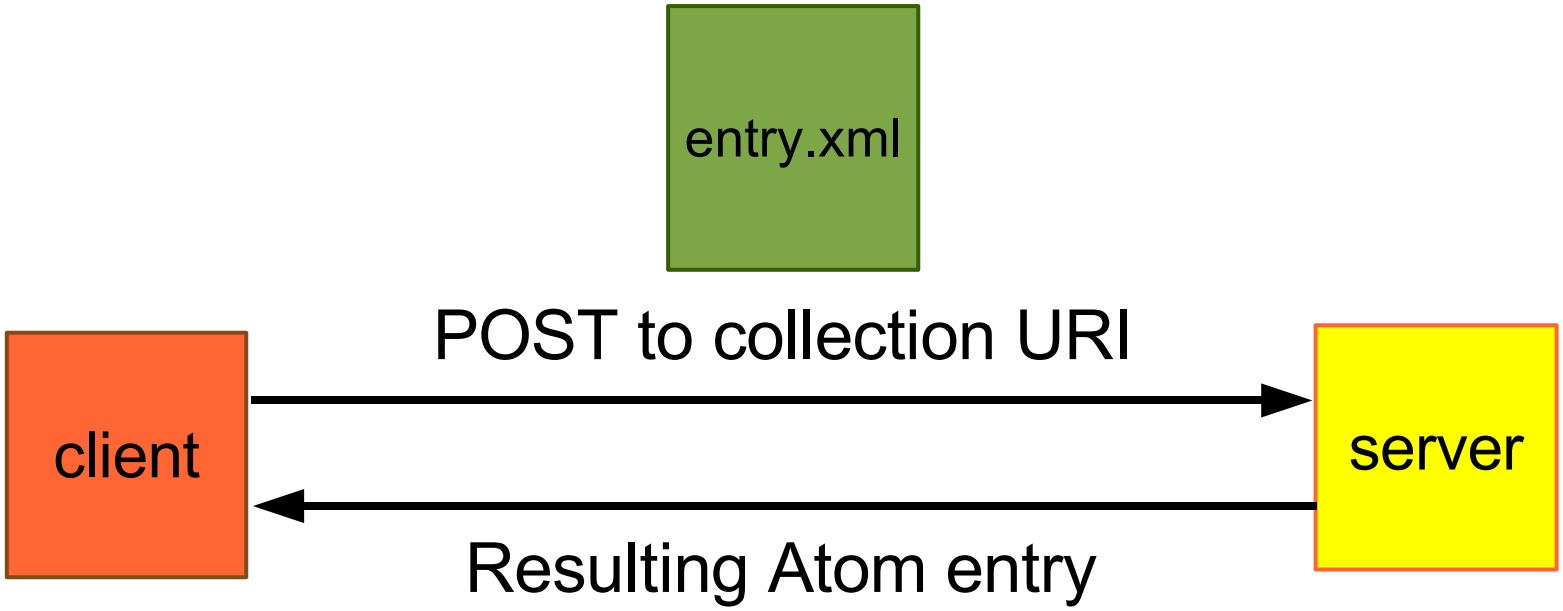
<entry> in a collection

```
  <entry>
    <title>First post!</title>
    <link rel="alternate"
       href="http://localhost/roller/page/bill?entry=post1 />
    <link rel="edit"
       href="http://localhost/roller/app/bill/entry/757" />
    <category term="/Sun" />
    <id>http://localhost:8080/roller/page/bill?entry=post1</id>
    <updated>2005-12-27T22:08:03Z</updated>
    <published>2004-10-13T01:07:59Z</published>
    <content type="html">I'm so blogging this</content>
    <app:control>
      <app:draft>no</app:draft>
    </app:control>
  </entry>
</feed>
```

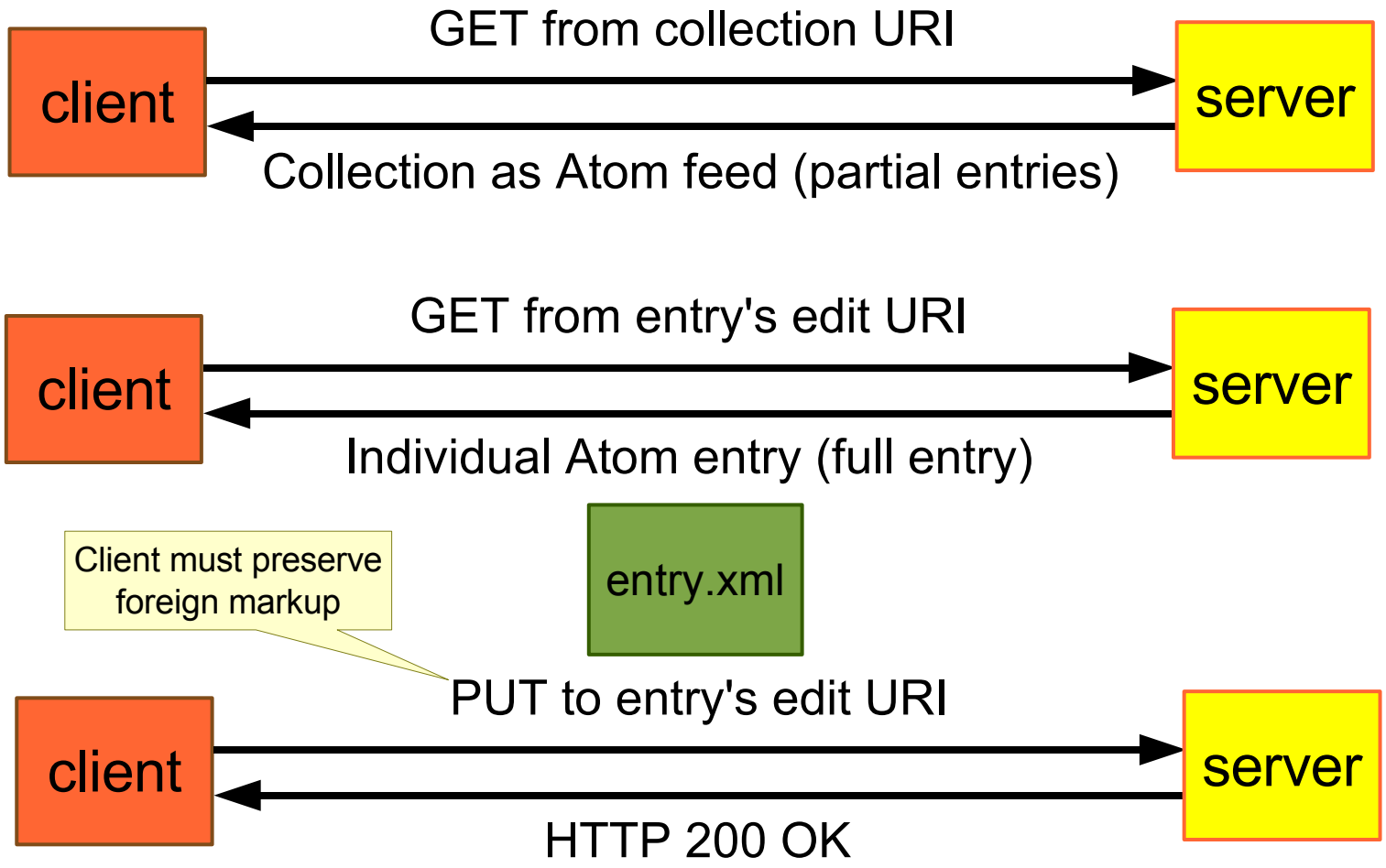
Edit URI for entry

APP namespace
to indicate draft status

Creating an entry



Updating an entry



Beyond blogging

APP as universal web glue

APP as universal web glue

- Not just for blogs anymore
- Entries can carry any type of data
- You can do a lot with collections + CRUD

- Consider Atom Publishing Protocol as the basis for your next web service

- What's missing
  - Hierarchical collections
  - Collection queries

For More Information

- Atom Enabled
  - http://www.atomenabled.org/
- RSS and Atom in Action
  - http://manning.com/dmjohnson
- Blogapps project
  - http://blogapps.dev.java.net
- My blog
  - http://rollerweblogger.org/page/roller