Implement news syndication using RSS and Atom

A handy reference for using RSS 2.0 and Atom 1.0 in content syndication

Level: Intermediate

Ying Ying Lin (queenielyy@yahoo.com.cn), Software Engineer, Dong Hua University, Shanghai, China

26 Sep 2006

The advent of RSS and Atom technologies brings a bright new era of news syndication. It takes time, however, for Web site administrators to publish the news manually every day and to manage e-mail subscribers. This article shows how to implement a general news publication architecture using RSS and Atom syndication formats to ease the process and minimize human error.

If you have at least a basic understanding of content syndication and the RSS 2.0 format specification, as well as the Atom 1.0 format specification, then you're ready to implement a news syndication system using one or the other or both, and this article is intended to be a handy reference for you.

RSS and Atom syndication

RSS and Atom are similar XML-based document formats that describe lists of related information known as feeds. These feeds are composed of a number of items, each with an extensible set of attached metadata; for example, each item has a title. The primary purpose of these feeds is the syndication of Web content such as Weblogs or news headlines to Web sites and directly to user agents.

Two examples of RSS 2.0 and Atom 1.0

Listing 1. Sample RSS 2.0 feed

```
<?xml version="1.0"?>
<rss version="2.0">

<channel>
   <title>Feed Title</title>
   <link>http://yourwebsite.com/</link>
   <description>Feed Description</description>
   <language>en-us</language>
   <pubDate>Mon, 03 Jan 2005 12:00:00 GMT</pubDate>

<item>
   <title>Article Title</title>
   <link>http://yourwebsite.com/articlelink.html</link>
   <description>Your content included here.</description>
</item>
<item>
   <title>Sports</title>
   <link>http://yourwebsite.com/sportslink.html</link>
   <description>Your content included here.</description>
</item>

</channel>
</rss>
```

Listing 2. Sample Atom 1.0 feed

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

  <title> Feed Title </title>
  <link href=" http://yourwebsite.com/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>Your Name</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Article Title</title>
    <link href=" http://yourwebsite.com/articlelink.html "/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>
  <entry>
    <title>Sports</title>
    <link href=" http://yourwebsite.com/sportslink.html "/>
```

```
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344e45ab90</id>
    <updated>2003-12-14T13:30:55Z</updated>
    <summary>Some text.</summary>
  </entry>

</feed>
```

Similarities in RSS and Atom feeds

As you've seen in the previous two examples (Listing 1 and Listing 2), RSS and Atom are similar XML-based formats. Their basic constructions are the same with only a small difference between the expressions of the nodes.

Each feed file actually represents a channel. It has a channel title, link, description, author, and so on. The channel information provides the basic information about the feed. Following the channel information are several items. Each item represents a real piece of news or articles you can read from the feed reader. Usually an item contains a title, link, date of update, and summary information.

Differences of RSS and Atom feeds

With RSS 2.0 and Atom 1.0, Compared as a reference, review the differences between RSS and Atom.

Table 1. RSS 2.0 and Atom 1.0 Compared

Difference	RSS 2.0	Atom 1.0
Deployment	RSS 2.0 is widely deployed.	Atom 1.0 is not widely deployed.
Specification	Harvard has copyrighted and frozen the RSS 2.0 specification.	The Atompub Working Group (within the IETF) agreed on the Atom 1.0 specification, and that it could be revised in the future.
Required content	RSS 2.0 has a feed-level title, link, and description required. It doesn't require that any individual item fields be present in a feed.	Atom 1.0 has a title (which can be empty), a unique identifier, and a last-updated time stamp required for both feeds and entries.
Payload	RSS 2.0 can have plain text or escaped HTML, but you can't tell which of the two is provided.	Atom 1.0 has a payload container.
Full or partial content	RSS 2.0 has a `<description>` element that can contain either the full text or a synopsis of an entry. It doesn't have a built-in way to signal whether the contents are complete.	Atom 1.0 offers separate `<summary>` and `<content>` `elements`. The summary is good to use for accessibility reasons if it's nontextual or nonlocal content.
Auto-discovery	RSS 2.0 auto-discovery is implemented in different ways.	Atom 1.0 standardizes auto-discovery.
Extraction and aggregation	RSS 2.0 has only one recognized form: an `<rss>` document.	Atom 1.0 allows stand-alone Atom Entry documents, which can be transferred using any network protocol; for example, XMPP. Atom also has support for aggregated feeds, where entries point to the feed they came from if they will be included in other feeds.

A general architecture for RSS and Atom publication system

With the development and sophistication of the RSS and Atom format specifications, more and more Web applications are being implemented. The most popular and typical implementation using RSS or Atom is a news publication system.

Here is a general architecture for implementing a publication system using both RSS and ATOM feeds. This architecture consists of three parts:

- A feed generator subsystem
- A feed operation subsystem
- An auto-publication subsystem

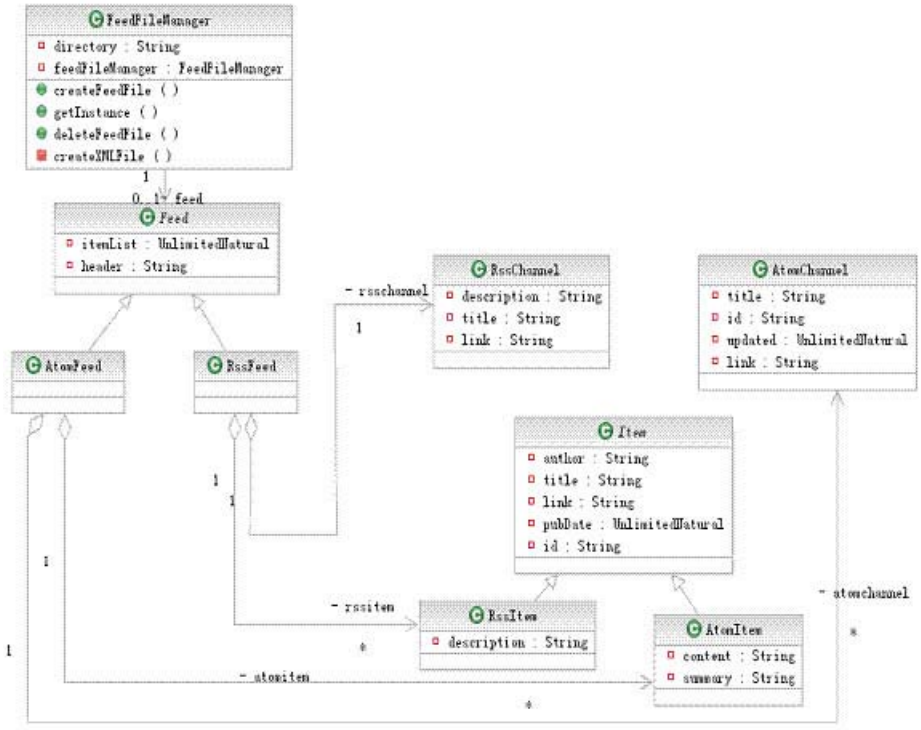
Now view the details of the three subsystems individually.

A feed generator

A feed generator takes charge of generating XML-based feed files. The class diagram of the feed generator is as

follows:

Figure 1. Feed generator class diagram

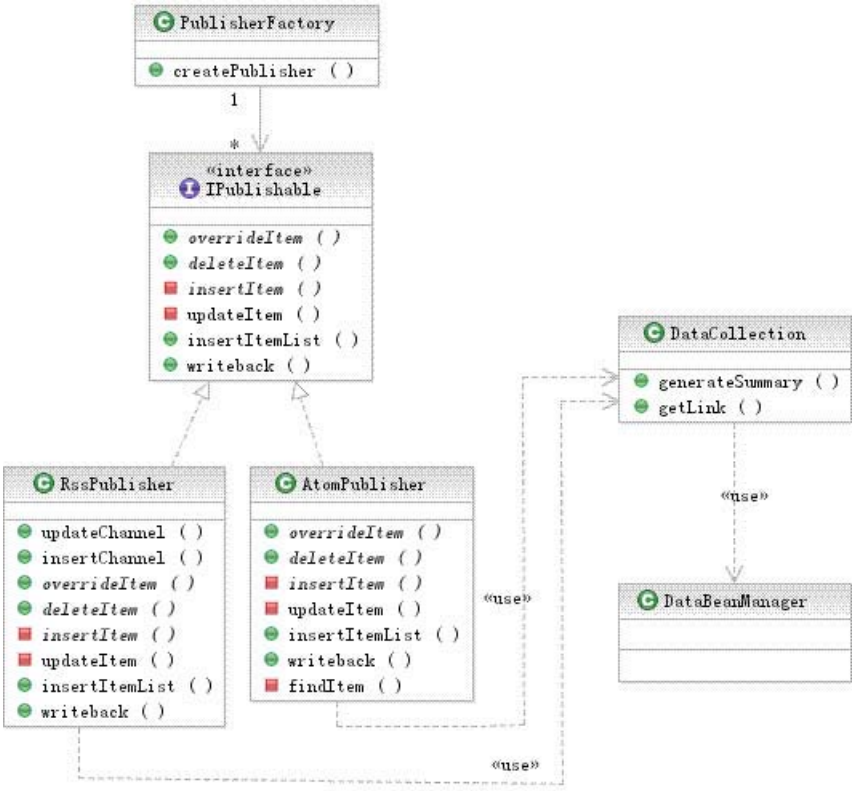


The core of the feed generator is class `FeedFileManager`. It generates the XML file related to a particular feed. The feed has two specific types: `AtomFeed` and `RssFeed`. Both of the two feed classes contain a feed channel class and a feed item class. The realizations of the feed channel class and feed item class will vary due to their different constructions.

A feed operation subsystem

A feed operation subsystem provides several feed operation functions to operate these two types of feed files. For example, `insertItem()`, `deleteItem()`, `updateItem()` functions. The class diagram of the feed operation subsystem is as follows:

Figure 2. Feed operation subsystem class diagram

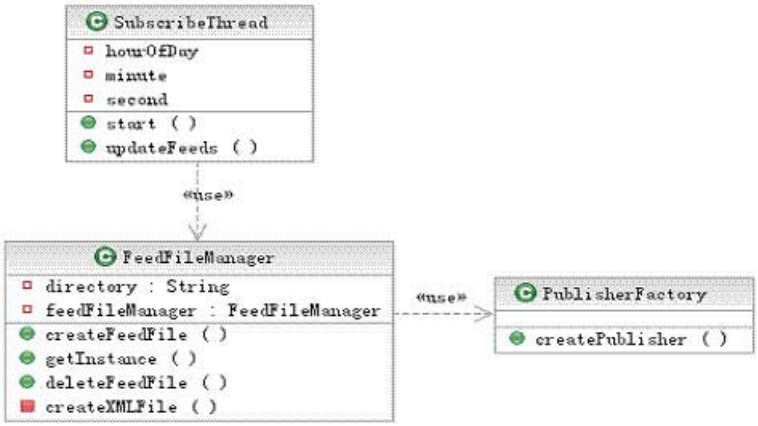


First, you need an abstract factory pattern `PublisherFactory` to generate either one of two feed publishers. Then the publisher pulls out the data from the database through a `DataCollection` class.

An auto-publication subsystem

An auto-publication subsystem is a timer for updating the feed files in a fixed time. This is the real part of utilizing the previous two subsystems and makes the publication system work.

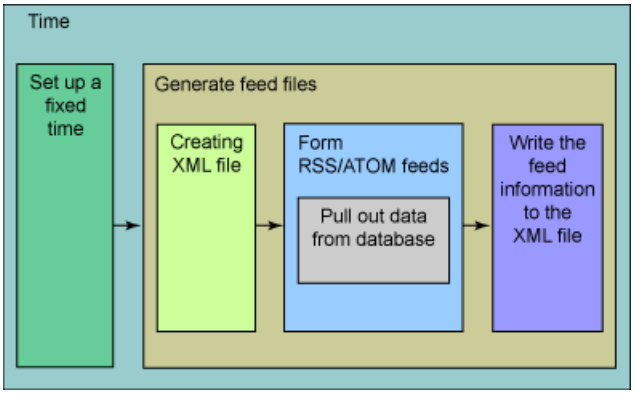
Figure 3. Auto-publication subsystem class diagram



Step-by-step implementation of how to generate RSS and ATOM feeds automatically

Now, look at the detailed implementation procedure, step by step. First, Figure 4 provides a general implementation procedure diagram. It shows several main building blocks and the relationship between them.

Figure 4. Overview of implementation procedure



The whole application is a big timer, so you can set a fixed launch time inside it. When time is up, the timer starts to do the scheduled work -- generate feed files. First, the application generates an XML file, then forms RSS and Atom feeds, and finally writes these feeds to the created XML file. The source of the feeds can be a database, a file or various other resources.

Timer

A timer is the core of the publication system. It triggers the system to generate feed files according to the user's appointment or in a fixed time per day. For a Java application, you can usually use the Java standard scheduling classes: `Timer` and `TimerTask` to implement a repeated execution schedule task.

It's more common to compose a scheduled recurring task in this publication system. You can refer to Scheduling recurring tasks in Java applications for detailed information on how to compose a scheduled recurring task.

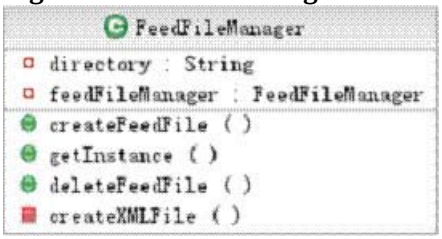
Listing 3. Sample Timer start method

```
public void start() {
    SchedulerTask st = new SchedulerTask(){
    public void run() {
    try {
        updateNewsFeeds(); //concrete method of update feed files.
        updateBooksFeeds();
         ...
        } catch (Exception e) {
            e.printStackTrace();
             }
       }
     };
```

Generate XML-based feed files

Both RSS and ATOM feeds are XML based, so generating these XML-based files is important during the implementation procedures. The `FeedFileManager` class in the feed generator subsystem is responsible for this task.

Figure 5. FeedFileManager class



Generally this class should contain a `createFeedFile()` method. This method needs to do three things:

- Create an XML file. You can create a static method createXMLFile() to achieve this goal.
- Interact with "Feed Operation Subsystem" to create the related channel and item information.
- Write all of that information back to the XML file.

Listing 4. Sample createFeedFile method

```
public String createFeedFile(String channelid, String name, String type)
```

```
                throws Exception {
        ...
        createXMLFile(file, type);

        IPublishable publisher = PublisherFactory.createPublisher(type, file);
        Feed feed = getFeed(type);

        Channel channel = feed.getChannel();
        ArrayList itemlist = feed.getItemList();

        publisher.insertChannel(channel);
        publisher.insertItemList(itemlist);
        publisher.writeback(file);

        return file;
    }
```

Form RSS/Atom feeds

In the last procedure, the publish factory creates a publisher and inserts related channel information and itemlist to the certain feed.

Listing 5. Sample code to insert channel and itemlist

```
        publisher.insertChannel(channel);
        publisher.insertItemList(itemlist);
```

To form the exact channel and itemlist, pull out the data from the database and input pulled-out data to the feed pattern. The following demonstrates how to pull out the data and then input the information to the related databean:

Listing 6. Sample code for feeds

```
    ...
    NewsBeanManager newsmanager = new NewsBeanManager();
    newslist = newsmanager.getAllNews(); //interact with database
    ...
    for (Iterator it = newslist.iterator(); it.hasNext(); it.next()) {

        title = "News " + i + ": " + ((NewsBean) newslist.get(i)).getTitle();
        link = ((NewsBean) newslist.get(i)).getLink();
        author = ((NewsBean) newslist.get(i)).getAuthor();
        timestamp = ((NewsBean) newslist.get(i)).getPublishTime();
        id = String.valueOf(((NewsBean) newslist.get(i)).getNewsID());
        description = ((NewsBean) newslist.get(i)).getSummary();
        content = ((NewsBean) newslist.get(i)).getContent();
        rssitem = new RssItem(title, link, author, timestamp, id,
                              description);
        rssitemlist.add(i, rssitem);
        atomitem = new AtomItem(title, link, author, timestamp, id,
                              description, content);
        atomitemlist.add(i, atomitem);
        i++;
        }
```

Implementation tips

After you implement a news publication system successfully, you might benefit from the following tips for deployment.

Implementing RSS or Atom feeds for most popular feed readers

To meet the needs of most popular feed readers, pay attention to the following when you generate those feed files.

- Add the RSS and Atom version information to the head of the XML file, as in Listing 7:

Listing 7. Sample code to add RSS and Atom version information

```
 <?xml version="1.0" encoding="utf-8"?>
   <rss version="2.0">
   ...
 <?xml version="1.0" encoding="utf-8"?>
```

```
<atom version="1.0">
```

- Provide link information for the RSS and Atom channel. In RSS 2.0, the link information appears as a node, but in the Atom 1.0 link, the information appears as an attribute. It's a good idea to give the link information for the channel part, although the link isn't a mandatory item for Atom (see Listing 8).

Listing 8. Sample code to provide link information for the channels

```
<?xml version="1.0" encoding="utf-8" ??>
<rss?>
<channel?>
   <title?>News Syndication</title?>
   <link?>http://www.newssyndication.com</link?>
   <description?>this is a news feed</description?>
   ...

<?xml version="1.0" encoding="utf-8" ??>
<feed xmlns="http://www.w3.org/2005/Atom"?>
   <title?>News Syndication</title>
   <link href="http://www.newssyndication.com" /?>
   <updated?>2006-07-06T10:26:30Z</updated?>
   <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id?>
```

Support globalization when generating XML files

Add encoding information to the header of the XML file.

Listing 9. Sample code to ad encoding information in XML file header

```
<?xml version="1.0" encoding="utf-8"?>
```

When you implement the function to write the XML file, use `OutputStreamWriter` instead of `FileWriter` to set encoding as utf-8. The default encoding of `FileWriter` is "GBK."

Listing 10. Sample code to set encoding as utf-8 with OutputStreamWriter

```
OutputStreamWriter writer = new OutputStreamWriter(outputstream,"utf-8");
```

In conclusion

In this article, you first reviewed the similarities and differences between RSS and Atom feeds, and then saw the architecture for implementing a publication system using both types of feeds. This architecture consists of three subsystems, which work in correlation with each other. After the step-by-step demonstration, you saw several tips for the implementation procedures. Enjoy your streamlined approach to news feeds.

Resources

Learn

- What is Atom?: Get a quick start in Atom .
- RSS 2.0 and ATOM 1.0, Compared: Find detailed information regarding the differences between RSS 2.0 and Atom 1.0.
- The RSS 2.0 Specification: Get everything you need in the specs.
- RSS and Atom feeds: Build your own RSS and Atom feeds after you dig into developerWorks' comprehensive

site.

- RSS Tutorial: Learn what RSS is, how it started, and how to do it yourself.
- Workplace RSS feeds and blogs: Stay up-to-date with the latest content from the Workplace discussion forums, the Workplace product support pages, developerWorks Workplace articles, and developerWorks Workplace tutorials.
- Scheduling recurring tasks in Java applications (Tom White, developerWorks, November 2003): Read this summary of the Java language's `Timer` class.
- developerWorks Web development zone: Expand your site development skills with articles and tutorials that specialize in Web technologies.
- developerWorks technical events and webcasts: Stay current with jam-packed technical sessions that shorten your learning curve, and improve the quality and results of your most difficult software projects.

Get products and technologies

- IBM trial software: Build your next development project with software available for download directly from developerWorks.

Discuss

- developerWorks blogs: Get involved in the developerWorks community!

About the author

Ying Ying Lin is a graduate student from Dong Hua University and currently an intern at IBM China's Globalization Lab in Shanghai. She is interested in Java and workflow application technologies. You can contact her at queenielyy@yahoo.com.cn.
