

# Simple RSS with Perl

*brian d foy, comdog@panix.com*

## Abstract

The Rich Site Summary (RSS) provides a way for web site operators to allow other people to syndicate the web site content. The web site operators publish a *feed* which other people can download, parse, and display on their own site. I present the RSS files that *The Perl Review* provides along with the program we use to parse other site's RSS files. I only show RSS 0.9.

## 1 Introduction

The Rich Site Summary (RSS) is a set of stories, usually from the same web site. The RSS file contains XML data which other programs can parse to create custom presentations.

There are several major versions of RSS. The first version, 0.9, is simple to read and simple to use. It does not contain as much information as later versions which provide various additions to the RSS format—even arbitrary extension through XML namespace magic. I think that is overkill for my purposes, and I do not want to do the extra work to handle the extra features. I use version 0.9, and I only show that in this article.

*The Perl Review* publishes RSS files for each issue and for collections of related articles. A few other web sites actually use them. In the other direction, we use several other sites' RSS files on our web site for "Perl at a Glance".

## 2 Creating RSS

Files in RSS 0.9 typically have the extension `.rdf` for Resource Description Framework. The XML format is very simple. Code listing 1 shows the actual RSS file for the last issue. Line 1 is the required XML header. Line 3 pulls in the appropriate definitions. The RSS data has *channel* and *item* data. The channel is the name of the feed and typically has a title and a link to the original website. Line 7 starts my channel element with a title element with the name of the issue, and a link element to the issue file. The rest of the elements are items, sometimes called *headlines*. Each item has a title and link.

Code Listing 1: RSS file format

```
1 <?xml version="1.0"?>
2
3 <rdf:RDF
4   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5   xmlns="http://my.netscape.com/rdf/simple/0.9/">
6
7 <channel>
8 <title>The Perl Review, v0 i5, September 2002</title>
```

```

 9 <link>http://www.theperlreview.com/Issues/The_Perl_Review_0_5.pdf</link>
10 </channel>
11
12 <item>
13 <title>Extreme Mowing, by Andy Lester</title>
14 <link>http://www.theperlreview.com/Articles/v0i5/extreme_mowing.pdf</link>
15 </item>
16
17 <item>
18 <title>What Perl Programmers Should Know About Java, by Beth Linker</title>
19 <link>http://www.theperlreview.com/Articles/v0i5/perl-java.pdf</link>
20 </item>
21
22 <item>
23 <title>Filehandle Ties, by Robby Walker</title>
24 <link>http://www.theperlreview.com/Articles/v0i5/filehandle_ties.pdf</link>
25 </item>
26
27 <item>
28 <title>The Iterator Design Pattern, by brian d foy</title>
29 <link>http://www.theperlreview.com/Articles/v0i5/iterators.pdf</link>
30 </item>
31
32 </rdf:RDF>

```

Creating this file is very easy. I can write it by hand, but I can also create it with a program, which I want to do since all of the data comes from a database and I want to automate the process.

Code listing 2 shows a simple program to create the RSS data in code listing 1. The XML::RSS module can handle several versions of RSS, so in line 5 I specify version 0.9. I can change the version number to another one that XML::RSS can handle and see the results. Other versions are a bit more complicated, but XML::RSS handles them for me through the same interface so I can decide to switch later and not have to completely rewrite the script.

On line 10, I create the RSS channel from the first two lines from DATA. On line 15, I start a while loop to process the rest of the data. I skip lines without non-whitespace, chomp the line that I read in the while condition, then read another line of data and chomp that. I expect the first line to be the title and the line after that to be the link address. I add these to my RSS object on line 22.

On line 28, I simply print the RSS data as a string. It should look close to the output in code listing 1, although some people may see slight differences for different versions of XML::RSS.

```

_____ Code Listing 2: Create RSS files with XML::RSS _____
 1  #!/usr/bin/perl -w
 2  use strict;
 3
 4  use XML::RSS;
 5  my $rss = XML::RSS->new( version => '0.9' );
 6
 7  chomp( my $channel_title = <DATA> );
 8  chomp( my $channel_link  = <DATA> );
 9

```

```
10 $rss->channel(  
11     title      => $channel_title,  
12     link       => $channel_link,  
13 );  
14  
15 while( defined( my $title = <DATA> ) )  
16 {  
17     next unless $title =~ /\S/;  
18     chomp $title;  
19  
20     chomp( my $link = <DATA> );  
21  
22     $rss->add_item(  
23         title => $title,  
24         link  => $link,  
25     );  
26 }  
27  
28 print $rss->as_string;  
29  
30 __END__  
31 The Perl Review, v0 i5, September 2002  
32 http://www.theperlreview.com/Issues/The_Perl_Review_0_5.pdf  
33  
34 Extreme Mowing, by Andy Lester  
35 http://www.theperlreview.com/Articles/v0i5/extreme_mowing.pdf  
36  
37 What Perl Programmers Should Know About Java, by Beth Linker  
38 http://www.theperlreview.com/Articles/v0i5/perl-java.pdf  
39  
40 Filehandle Ties, by Robby Walker  
41 http://www.theperlreview.com/Articles/v0i5/filehandle_ties.pdf  
42  
43 The Iterator Design Pattern, by brian d foy  
44 http://www.theperlreview.com/Articles/v0i5/iterators.pdf
```

---

### 3 Parsing RSS

The XML::RSS module makes parsing RSS even more easy than creating it. In code listing 3, which shows the actual program we use to generate the HTML for “Perl at a Glance”, most of the work deals with HTML, not RSS. Line 7 defines the RSS files to download. I found those by either visiting the site or asking the author if they had RSS files. For instance, Randal Schwartz has RSS feeds for most of his columns although he does not advertise this on his web site—at least not somewhere I could find.

Line 18 defines the location the program stores output files. Each feed has an associated output file which contains just its portion of HTML that another program collates into the final web page.

Line 20 starts the foreach loop which cycles through all of the RSS files. On line 22, I copy the URL to \$file so I can manipulate \$file and use it as the file name in the open on line 26. If I cannot open the file, I skip to the next feed. On line 34, I select the file handle I just opened so I do not have to specify it in all of the print statements.

On line 36, I create a new RSS object. I do not have to specify the version I want to use because XML::RSS figures it out based on the data I feed it in line 38. Once the module parses the data, I simply access the parts that I need. On lines 40 and 41 I get the channel title and image, which are hash references I store in \$channel and \$image.

On line 43, I start the HTML output. At some point I will change this program to use Text::Template, but for now something is better than nothing, even if this is horribly wrong. Next issue I will convert this program to store configuration and template data apart from the code to make up for it.

On line 49, I check if \$image has a url key. The feed might not have included a logo and I do not want a broken image icon to show up if it did not. With an image, I use the image location to form the link back to the original site, and without an image, I use the channel title.

On line 67, I iterate through the items in the feed and print a link for each one. Once I finish with the items I finish the HTML output and close the filehandle.

Code Listing 3: Fetch and parse RSS feeds

```
1  #!/usr/bin/perl -w
2  use strict;
3
4  use LWP::Simple;
5  use XML::RSS;
6
7  my @files = qw(
8  http://use.perl.org/useperl.rss
9  http://search.cpan.org/rss/search.rss
10 http://jobs.perl.org/rss/standard.rss
11 http://www.perl.com/pace/perlnews.rdf
12 http://www.perlfoundation.org/perl-foundation.rdf
13 http://www.stonehenge.com/merlyn/UnixReview/ur.rss
14 http://www.stonehenge.com/merlyn/WebTechniques/wt.rss
15 http://www.stonehenge.com/merlyn/LinuxMag/lm.rss
16 );
17
18 my $base = '/usr/home/comdog/TPR/rss-html';
19
20 foreach my $url ( @files )
21 {
22     my $file = $url;
23
24     $file =~ s|\.*/||;
25
26     my $result = open my $fh, "> $base/$file.html";
27
28     unless( $result )
29     {
30         warn "Could not open [$file] for writing! $!";
31         next;
32     }
33
34     select $fh;
35
36     my $rss = XML::RSS->new();
37     my $data = get( $url );
```

```

38     $rss->parse( $data );
39
40     my $channel = $rss->{channel};
41     my $image   = $rss->{image};
42
43     print <<"HTML";
44     <table cellpadding=1><tr><td bgcolor="#000000">
45     <table cellpadding=5>
46         <tr><td bgcolor="#aaaaaa" align="center">
47 HTML
48
49         if( $image->{url} )
50             {
51                 my $img = qq||;
52
53                 print qq|<a href="$$channel{link}">$img</a><br>\n|.
54             }
55         else
56             {
57                 print qq|<a href="$$channel{link}">$$channel{title}</a><br>\n|.
58             }
59
60         print qq|<font size="-1">$$channel{description}</font>\n|;
61
62         print <<"HTML";
63         </td></tr>
64     <tr><td bgcolor="#bbbbff" width=200><font size="-1">
65 HTML
66
67         foreach my $item ( @{ $rss->{items} } )
68             {
69                 print qq|<b>&gt;</b><a href="$$item{link}">$$item{title}</a><br><br>\n|;
70             }
71
72         print <<"HTML";
73             </font></td></tr>
74     </td></tr></table>
75 </td></tr></table>
76 HTML
77
78     close $fh;
79     }

```

### 3.1 Updating RSS feeds automatically

Once I decide which feeds I want to process and how I want to present them, I want to fetch them automatically. I can use crontab to schedule the program to run at certain times (cron comes with unix-like platforms and is available as third-party tools for windows). The frequency that I run this program depends on how often the sites update their feeds. I typically update things hourly, but not on the hour when everyone else is probably updating their feeds. I update at 17 minutes past the hour.

```
17 * * * * /usr/home/comdog/bin/rss2html.pl
```

### 3.2 Some things I do not cover

The RSS format handles a lot more than what I have shown, but once I have XML::RSS doing the hard work, everything else is easy. Other commonly-used features include search forms linking to the original site, channel descriptions, item descriptions, channel meta-data for caching, fetching, and more.

### 3.3 Some places offering RSS feeds

None of this parsing magic is much use to anyone unless people offer RSS feeds to parse.

#### *The Perl Review*

This journal offers several different RSS feeds to choose from. <http://www.theperlreview.com/rss/>

#### **“All the Perl that’s Practical to Extract and Report”**

use.Perl has a feed of its stories - <http://use.perl.org/useperl.rss>

#### **Recent modules**

CPAN Search has a feed of the latest 10 modules

## 4 Conclusion

With a minimum of effort, I can create or parse RSS files. The XML::RSS module handles the details of different versions for me so I can know as little or as much RSS as I care to know. When I parse an RSS file, I can access its parts through familiar Perl data structures.

## 5 References

*The Perl Review* - <http://www.theperlreview.com>

Perl at a Glance - [http://www.theperlreview.com/at\\_a\\_glance.shtml](http://www.theperlreview.com/at_a_glance.shtml)

O’Reilly Network RSS DevCenter - <http://www.oreillynet.com/rss/>

RSS News - <http://blogspace.com/rss/>

## 6 About the Author

brian d foy is the publisher of *The Perl Review*.