Simple Internet Subscriptions with VFP and XML

Session Number34

Ted Roche
Ted Roche & Associates, LLC
Contoocook, NH 03229
Voice: 603-746-5670
Email:tedroche@tedroche.com

Overview

Many news, discussion and weblog sites let users subscribe to change notifications using RSS – the RDF Site Summary or "Really Simple Syndication" standards. Standards? Yes, there are several. However, the good news is that RSS is encoded as XML and can therefore be parsed and produced with any application that's good at manipulating XML – like Visual FoxPro! But RSS is not limited to news organization feeds. Consider using it where many clients need to determine if one publisher has changed data to query. This session will include demonstrations of current uses of RSS in newsreaders, web sites, aggregation sites, and Visual FoxPro applications.

Attendees will learn how to: distinguish and parse the several "standards" in RSS variants; generate an RSS feed in Visual FoxPro; and develop their own innovative solutions, based on real-world examples of RSS in action.

Introduction

Resource Description Framework (RDF) and RDF Site Summary (RSS) have an interesting and mixed history. One of their forebears comes from academia, where technologists and linguists are attempting to parse the meaning of the web. Another root is from within the web community, searching for ways to find, search, and categorize the web to advance it to the next generation, summed up in Tim Berners-Lee's "Semantic Web" (see references). Still others were far more practically just trying to get their jobs done and glommed onto a new format that made sense for what they were trying to do.

Count me in the last camp. I came into blogging as an end-user. I started reading the websites of people who interested me. Some of them kept up journals of what they were doing. Some of them seemed to have some pretty slick content management systems (CMS) to manage all of these log entries they were making. It was remarkable how they kept track of what others were posting. Many were using "news aggregator" programs that somehow detected new postings. Pretty soon I started noticing little orange graphic "XML" tags on their sites, and my curiosity was piqued. What were these things? What were the XML and RSS they referred to? And, of course, could I do that in FoxPro?

It turns out that RSS is a rich format and fairly easy to work with. RSS and the principles behind the push- and pull- subscription model it presents offer developers some great opportunities to move data between producers and consumers without requiring the two to be closely coupled. In this paper, we'll look at what the RSS format is, how to read and write it from Visual FoxPro, and look at some of the innovative uses that others are putting it to, speculating on what other uses it could have over time.

What is RSS?

Here's a simple definition, and an explanation of why you should care: RSS is the language of an XML document you can use to publish and subscribe, transferring text information between two or more sites. These two clients do not have to be publicly exposed, nor do they have to be dealing with publications or subscriptions: they can be exchanging information of any sort.

Depending on whom you ask, RSS might stand for several things or nothing at all. It started life at Netscape Corporation as "RDF Site Summary" where RDF is yet another acronym for Resource Description Framework. RDF is a language to express metadata. "Rich Site Summary" and "Real Simple Syndication" are two other popular acronyms.

RSS was originally adopted by Netscape as a means of expressing the contents of a website or news feed in order to integrate it into a news portal as part of their My.Netscape.com web site. RDF is a key element in the W3C's concept of the "Semantic Web" where web resources are not simply opaque blogs, but have metadata explaining their use, context and purpose.

As is typical of many "standards," many people were responsible for the development. There are several branches claiming legitimacy and ownership of the standard. Dave Winer of Radio Userland picked up the RSS banner when Netscape faltered, and he was responsible for development of versions 0.91 through 0.94, dropping much of the RDF roots as too complex and unnecessary. A separate group developed "version 1.0" and have established the standard with the Internet Engineering Task Force (further resources are at the end of this document). After the release of 1.0, Winer responded with an updated 0.94 version dubbed "2.0"

The RSS 2.0 format and copyright have been turned over by Dave Winer to the Berkman Center for Public Policy at Harvard University. Dave has referred to the format as "frozen," and it is unclear if there will be future development efforts on it. However, RSS 2.0 is the most commonly used format, and is easy to read and parse. On the other hand, the RSS 1.0 group claims their format has greater extensibility and future-proofing through the use of RDF and many more extension modules (namespaces) available. However, no compelling application has yet appeared.

So, what's a poor developer to do? If at all possible, I recommend supporting both standards. This ensures you have the greatest reach with the material you'll be publishing, and hedges your bets on which format, if either, will ultimately succeed. As you'll see in the next section, the formats are not that different, so support for both should not be onerous.

RSS Data Structures: 1.0 and 2.0

The concepts and terminology of RSS formats are very similar. Since all XML documents can have only one root (top-level) object, RSS 1.0 chooses "RDF" and 2.0 settles for "RSS." Each format has a "channel" element that's the one side of a one-to-many relationship, like an order header, that describes the "news channel" – who it's from, where it is located, characteristics common to all the news items. Then, the RSS files have the news items themselves. Here, RSS 1.0 and 2.0 differ. In RSS 2.0 (see Figure 2), the items are included as sub-elements within the channel element. In RSS 1.0 (see Figure 1), an items collection in the channel list the items included, somewhat like a table of contents. The items are included as separate elements at the same level as the channel element, making them all sub-elements of the RDF element.

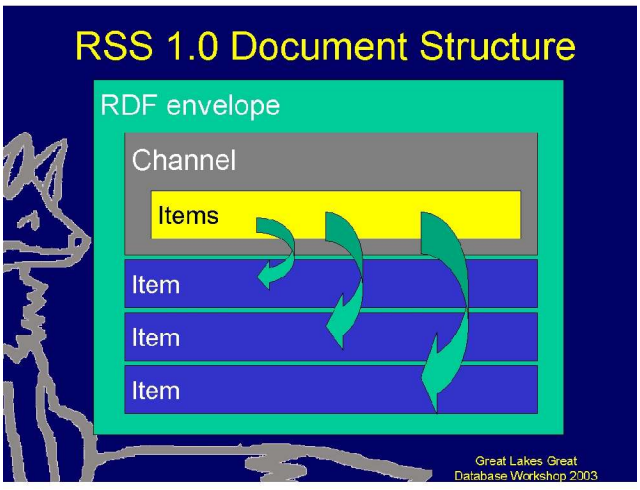

Figure 1: RSS 1.0 Document Structure

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:admin="http://webns.net/mvcb/"
  xmlns:cc="http://web.resource.org/cc/"
  xmlns="http://purl.org/rss/1.0/">

<channel rdf:about="http://www.foxcentral.net/">
<title>FoxCentral</title>
<link>http://www.foxcentral.net/</link>
<description>Visual FoxPro Central, a news site for the Fox Community</description>
<dc:language>en-us</dc:language>
<dc:creator>The FoxPro Community</dc:creator>
<dc:date>2003-08-21T02:14:31-00:00</dc:date>
<admin:generatorAgent rdf:resource="http://msdn.microsoft.com/vfoxpro" />
<admin:errorReportsTo rdf:resource="mailto:tedroche@tedroche.com" />
<items>
<rdf:Seq><rdf:li rdf:resource="http://www.bostonusergroups.com/vfpboston" />
<rdf:li rdf:resource="http://www.foxtoolbox.com/itemgroup.dbx?sku=E32%2D00001" />
<rdf:li rdf:resource="http://www.mwfpug.org" />
<rdf:li rdf:resource="http://microsoft.com/downloads/details.aspx?FamilyId=0F43EB58-7A94-4AE1-
A59E-965869CB3BC9&amp;displaylang=en" />
<rdf:li rdf:resource="http://www.west-wind.com/articles.asp" />
<rdf:li rdf:resource="http://www.lafox.org/home.page.fox" />
</rdf:Seq>
</items>
</channel>


       <item rdf:about="http://www.bostonusergroups.com/vfpboston">
              <title>Boston FUG, Ken Levy, VFP8 News, Europa, .NET &amp;amp; XML Rescheduled to
August 13th</title>
              <description><![CDATA[Last minute rescheduling: Boston VFP User Group will meet on
Wednesday, August 13, 6:30pm at the Microsoft offices, 6th floor, 201 Jones Rd., Waltham MA,
presenting Ken Levy, Microsoft VS Data Product Manager, showing the latest news for Visual
FoxPro.  Ken will also discuss and demo Visual FoxPro 8.0 along with VFP 8.0 working with VS .NET
2003.  Ken will also show demos and discuss Europa (next version of VFP) and Whidbey (next
version of Visual Studio .NET).  In addition, Ken will show some exciting demos of the new
XML/XSLT editor/debugger for VS .NET.  For more information on the latest news that will be
discussed, refer to http://msdn.microsoft.com/vfoxpro/letters/ and for more UG information and
directions, tune into http://www.bostonusergroups.com/vfpboston]]></description>
              <link><![CDATA[http://www.bostonusergroups.com/vfpboston]]></link>
              <dc:date>2003-08-13T18:52:00-00:00</dc:date>
       </item>
       (more items would appear here…)
</rdf:RDF>
```

Listing 1: An RSS 1.0-formatted feed using additional XML namespaces

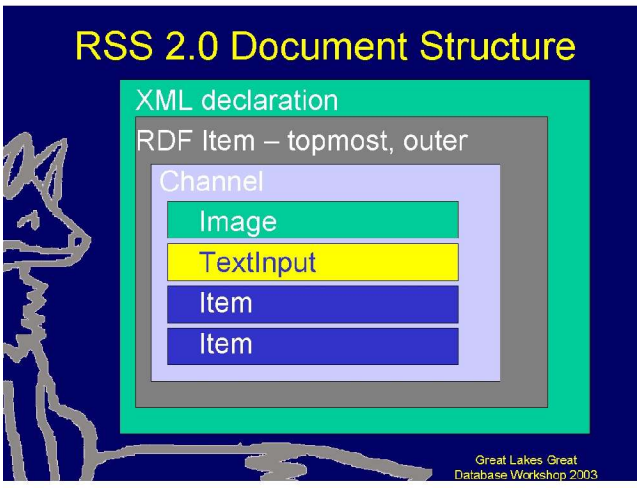

Figure 2: RSS 2.0 Document Structure

```
<?xml version="1.0"?>
<!--  RSS generated by Visual FoxPro 08.00.0000.2521 on Wed, 25 Jun 2003 00:32:45 GMT-->
<rss version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="c:\Development\FoxWikiWS\FoxWikiRSS20.xsd">
<channel>
  <title>FoxForum Wiki</title>
  <link>http://fox.wikis.com/</link>
  <description>KnowledgeBase and Community about Visual FoxPro</description>
  <language>en-us</language>
  <copyright>Copyright 2002 by the authors</copyright>
  <lastBuildDate>Wed, 25 Jun 2003 00:32:45 GMT</lastBuildDate>
  <docs>http://backend.userland.com/rss</docs>
  <generator>Visual FoxPro 08.00.0000.2521 for Windows</generator>
  <category domain="Syndic8">23766</category>
  <ttl>40</ttl>


       <item>
              <title>EuropaFeatures</title>
              <description><![CDATA[MalcolmGreene summarized the various coverages and posted
this list on  ProFox .  Array: - 65K limit on array size has been removed (limited now only by
available memory)   Data engine: - New data type: Timestamp (for clarification, there is no new
Times]]></description>
              <link>http://fox.wikis.com/wc.dll?Wiki~EuropaFeatures</link>
              <guid>http://fox.wikis.com/wc.dll?Wiki~EuropaFeatures</guid>
              <pubDate>Tue, 24 Jun 2003 20:38:59 GMT</pubDate>
       </item>
```

Listing 2: RSS 2.0 formatted subscription data

How does a VFP application consume RSS?

RSS is a specialized form of XML file, and can be consumed by Visual FoxPro in any of the ways VFP can interact with XML. Key elements of RSS can be extracted with the STREXTRACT() function, with VFP's low-level string functions, or by using an XML parser to load the document and interpret it. There are advantages and disadvantages to each technique.

With string extraction, you need to know the exact phrase you are looking for, and how the document is formatted. For example, suppose you wanted to pull all of the item titles out of an RSS feed. With the VFP code in Listing 3, you can find and display all of the item titles. However, there are a few gotchas the code had to work around. First, since element tags can have additional attributes, you have to search for just the open tag and tagname sequences, like "<item", and not the entire tag. This leaves you with an end tag to clean up later, and problems if you have similar tags (the IF…ENDIF block is to avoid the <items> tag in RSS 2.0).

```
* ListItems.PRG
LOCAL lcFilename as String, ;
      lnCount as Integer, ;
      lcXML as String, ;
      lcString as String

lcFileName = GETFILE()
IF NOT FILE(lcFileName)
  RETURN
ENDIF

lcXML = FILETOSTR(lcFileName)
IF "<items" $ lcXML
  lnCount = 2
ELSE
  lnCount = 1
ENDIF

lcString = STREXTRACT(lcXML, "<item", "</item>", lnCount)
DO WHILE NOT EMPTY(lcString)
  ? STREXTRACT(lcString, "<title", "</title>")
```

```
  lnCount = lnCount + 1
  lcString = STREXTRACT(lcXML, "<item", "</item>", lnCount)
ENDDO
```

Listing 3: ListItems.PRG lists the titles of all items within an RSS feed using native VFP string functions.

Using Visual FoxPro's native string functions has the advantage of using the raw speed of VFP, but they do not allow for as much flexibility as you will find with delegating the responsibility for parsing the document to a 3rd-party XML parser. Microsoft offers one such parser in their XMLDOM object (many other parsers are available from other software manufacturers). Listing 4 and Listing 5 illustrate two techniques of parsing out values from the RSS XML file using the DOM document. In Listing 4, from the root document, you can traverse through the collection of nodes in the document and, for each of the nodes named "item" traverse their child nodes to extract the text from the title elements. In Listing 5, a different technique is used, using the query language XPath to extract all elements named "item" (that are located one level of the hierarchy down from the root) and then directly iterating through that collection.

```
loXML = CREATEOBJECT("MSXML2.DOMDocument.4.0")
loXML.load(lcFileName)

loRoot = loXML.documentElement
lnNodeCount = loRoot.childNodes.Length
FOR m.i = 0 TO lnNodeCount -1
  IF loRoot.childNodes(m.i).baseName = "item"
    FOR m.j = 0 TO loRoot.childNodes(m.i).childNodes.length - 1
      IF loRoot.childNodes(m.i).childNodes(m.j).basename = "title"
        ? loRoot.childNodes(m.i).childNodes(m.j).text
      ENDIF
    NEXT && m.j
  ENDIF  && baseName = "item"
NEXT  && m.i
```

Listing 4: ListTitle.prg (repetitive code removed, but in the samples) loads the XML into the DOM and walks down the hierarchy to extract the titles

```
loXML = CREATEOBJECT("MSXML2.DOMDocument.4.0")
loXML.load(lcFileName)

loRoot = loXML.documentElement

loItems = loRoot.selectNodes("//item")
FOR m.i = 0 TO loItems.length - 1
  ? loItems.item(m.i).text
NEXT  && m.i = 0 TO loItems.length - 1
```

Listing 5: ListXPath.PRG uses the XPath querying language to extract the items of interest without having to traverse the document.

How can a VFP application generate RSS?

Visual FoxPro can generate RSS in the same ways it can generate XML output: you can assemble the file manually using string functions, you can generate XML using the XMLDOM parser, or you can do a combination of the two. Each technique has advantages and disadvantages. The main advantage of using FoxPro functions is the speed: raw text assembly in Visual FoxPro is wickedly fast, and orders of magnitude faster than assembling an XML document via the COM interfaces of XMLDOM. XMLDOM, on the other hand, is a validating component that will ensure that you are creating a legitimate XML document.

Over the past eight months, I've been working to create RSS feeds for two FoxPro-focused web sites, the FoxForum Wiki, http://fox.wikis.com, and the FoxCentral news site, http://www.foxcentral.net. I was disappointed at the lack of presence of VFP news in the blogging world, and am working to alleviate that. Both of these feeds, and various beta

variations, are available for examination or subscription from http://www.tedroche.com/RSSFeeds.html. The initial proof-of-concept versions of these applications were written in pure FoxPro code, using the Monkey-See-Monkey-Do (MSMD) rapid application development methodology of copying other people's stuff that worked.

Three betas have been completed and deployed to date. Each provided lessons and has some valuable code to learn from. Beta One was purely a proof of concept, a "See, Ma, no hands!" demo to prove it could be done, written in a day. Beta Two was an attempt to refine the process a bit, particularly when it came to using Web Services to obtain the data. Finally, Beta Three was the first serious attempt at refactoring the code into a reliable and maintainable production system. While there is a still a way to go to bulletproof code, the lessons learned here can start you on your way.

FoxWikiRSSBetaOne

The project in the "FoxWikiRSSBetaOne" directory contains the four files used for that process, and they are included for your enjoyment and hopefully reuse:

- **XMLDatetime.PRG**: a UDF to produce strings from dates in the format of "Wed, 15 Sept 2003 12:34:56 GMT" with proper time zone corrections, thanks to Andrew Coates.
- **ReadURL.PRG**: a UDF to return the HTML from a given URL, using the WinInet functionality directly.
- **HTMLText.PRG**: a UDF to strip a supplied HTML string of all markup by removing all less-than and greater-than signs and the markup within them.
- **MakeFoxWikiRSS.PRG**: the core program. Calls the other functions.

MakeFoxWikiRSS.PRG is the driver program. It reads the "Most Recent Changes" page from the FoxForum Wiki, requesting it in XML format. It converts that format to a cursor to drive the remainder of the program. Each topic is requested from the Wiki and a new RSS topic is generated from it. To save load on the FoxForum site, the topic is only read at its first change, and the local copy is used to present the synopsis of the site in the RSS feed after that. Finally, the program generates the output RSS, using FoxPro's textmerge functionality.

The program makes no effort to schedule itself; that is taken care of as part of the installation. The Windows Scheduler is used to set up the program to run on a regular frequency (I have it updating each hour from 5 AM to 11 PM EST) and the program quits when it is done.

Error handling is fairly simple, since this is not a particularly mission-critical task. The main logic is wrapped in a TRY…CATCH block, and any errors are logged and the application is shut down. An external batch program, also scheduled via Windows, emails the log to the administrator should an error occur. Since minimal code was used for this demonstration-of-technology, a few "normal" conditions would trip an error. For example, if no topics were recently changed, an empty result set would occur, and the XMLToCursor() function call would fail. Obviously, in a mission-critical application, you will want a more capable error handling system.

FoxWikiRSS20.xml is included as a sample output file.

FoxWikiRSSBetaTwo

The second revision to the FoxWiki RSS generation tried to take on a couple of new design ideas:

1. Take advantage of the Web Service interfaces Steve Black exposed at http://fox.wikis.com to return XML directly from the web site.
2. Generate an RSS 1.0 compliant feed, rather than the 2.0 feed generated from Beta One. This was to ensure that I could supply the needs of some consumers, who could only read one or the other.

There are three main programs included, and three other files already available on your VFP 8 distribution required to recompile the source:

- **HTMLText2.PRG** is a variant of the original, looking for explicit greater-than and less-than signs, rather than their encoded (>) versions, since we are reading a different source format now.
- **Iso8601DateTime.PRG** returns a file in the format specified in by the W3C and ISO, something like "2002-10-02T10:00:00-05:00"
- **FoxWikiRSS10.PRG** is the main program logic.
- **SetObjRef.PRG** is a Fox Foundation Class (FFC) program.
- **_Base.VCX** is an FFC base class library.
- **_Ws3Client.VCX** is an FFC class library for clients consuming Web Services.

The program logic is pretty similar, with a couple of notable exceptions. First, since the failure to find any changes would fail to create a cursor, the entire main logic is wrapped in an extended IF…ENDIF to support the TRY…CATCH outer block. There are more elegant ways to handle that. Second, since RSS 1.0 generates an "items" collection within the channel element, there needs to be two passes over the items collection. I chose to do that by creating an item cursor and then using CursorToXML() for the first pass. Since CursorToXML() can only create element-based or attribute-based XML, but not a mixture of the two, this leads to an ugly kludge shown in Listing 6. The CursorToXML() creates two elements, <item> and <about>, and this code modifies the text so it becomes <item about="…" >. Not a pretty fix, but another point for the power of VFP.

```
lcXML = STRTRAN(lcXML ,"<item>"+CHR(13)+CHR(10)+CHR(9)+CHR(9)+"<about>","<item rdf:about="+["])
lcXML = STRTRAN(lcXML ,"</about>", [">])
```

Listing 6: A quick and dirty cheat to turn two elements into an element with an attribute. I am not proud.

Beta Two is also deployed, at the web site above, generating RSS 1.0 code. A sample is included in the file **FoxWiki.rdf**.

The RSS 1.0 generator code works, and creates topics pretty quickly. But the architecture leaves a great deal to be desired. A single program specifies the technique to read the source, parse it, format it and output it, leading to a long and fragile chain of control. Any modification in the code could perturb the behavior of the rest of it. In short, it's monolithic, procedural, rigid, tightly-coupled and fragile. On to Beta Three.

FoxWikiRSSBetaThree

Beta one and two served as excellent proofs of concept. They got RSS content about FoxPro out into the blogging world (the "blogosphere") and into search engines. However, they lacked a great deal of flexibility. For this conference, I wanted to present something that was easier for you to work with for your own applications. Beta three is that product. The primary goals with beta three were:

1. Separation of the tasks of raw data retrieval from RSS generation, and
2. Use of the XMLDOM for document creation.

The first goal answers the objections raised in the last section, where any change to the code could disturb all layers of the monolithic application. In addition, it was clear that there could be use cases for the generation of several different kinds of output from the same source. Also, there might be situations where more than one input method made sense. Separating these processes and allowing them to communicate through a well-defined interface makes for greater flexibility to meet those needs with the least customization.

The second goal is to use Microsoft's XML Document Object Model COM control to generate the XML document. This goal has several motivations. It is a learning experience to use the XMLDOM. It's also a way to ensure the XML is well-formed. Also, hooking into the DOM can open up some interesting possibilities of going further with pure XML manipulation: verifying the document against a schema, for example, or using an XSL stylesheet to modify the output.

The components of this solution are located in a FoxWikiRSSBetaThree directory and consist of:

- **Driver.PRG** – a simple main program to drive the demo and display the results. In a real-life application, it's likely that this program would control the scheduling, present an interface to display status and logs, and be driven from metadata/process data tables to read a variety of data sources into a variety of back-ends at specified times and periodicities.
- **FoxWikiWS.PRG** – the front-end process of retrieving data from the data source, in this case the FoxForum Wiki via Web Services, and delivering it to the rendering engine in a predictable format.
- **ISO8601DateTime.PRG** – an even more simplified routine, with a clever TRANSFORM() replacing a bunch of string manipulation code
- **RSS20gen.PRG** – a generic program for generating RSS from the cursors created by FoxWikiWS.PRG or similar front-end programs. Note that there are still a few values, such as dc:language that are hard-coded and probably should be driven off additional columns in the curHeader cursor.

It's finally in Beta Three that we start to see the evolution of the code from get-it-done procedural code into reusable and maintainable modules. Particularly in RSS20gen.prg, we see the introduction of "classes," an object-oriented concept first introduced in Visual FoxPro 3.0 in 1995. Obviously, this code needs additional work before it is industrial-strength, but the foundation and the key elements you will need to understand to generate RSS are in place.

Lessons and Conclusions from Beta One, Two and Three

Straight Visual FoxPro code and raw HTML access is faster than document creation via XMLDOM and loading XML via Web Services by far. Running against the same data source, Beta one would completely load, do its processing and complete in four seconds. Beta Three took 23 seconds to simply read all the files via Web Services. However, unless you are planning to process thousands of sites, the difference is insignificant.

The size of the executable is noticeably smaller with plain XML, due to the inclusion of VCXes with the Web Services client. Again, however the differences aren't that significant.

In the Visual FoxPro-focused Beta One, there are fewer dependencies on other files (XMLDOM and Web Services). Deploying the first executable that used Web Services or XMLDOM to a plain Windows 2000 Server was difficult. Web Services required rebuilding a VFP install in InstallShield and selecting the correct merge modules. With XMLDOM, on the other hand, it was not clear if there were redistributables available that could be integrated into an install package. In the interests of getting things installed, the XML 4.0 Parser SDK was installed on the server, but that is surely not the proper technique!

This does also raise a concern that future updates to these files or that installation on other nearly-compatible operating systems will also be problematic. This problem is not unique to this situation, but is always a concern when you have increasing dependence on components not completely under your control. If your application is likely to be used in situations where you do not have a lot of control over the components installed, you may want to consider sticking with the more FoxPro-centric solutions in Beta One and Beta Two.

What comes next?

Beta Four will include a set of classes refactored from the simple one-level classes to abstract classes with the common routines and template methods in them, and specialized classes for the different specific behaviors. In particular, the use of namespaces is an ideal application of the "Hooks and Anchor" pattern advocated by Steven Black (see references at the end of this paper for details). However, Beta Three code is perfectly usable for the few sites I expect to be generating RSS from, so there's little pressure to refactor until there is a demand.

How is RSS used now?

Blogging is probably the most common use of RSS now, but news aggregation and other forms of aggregation are up and coming. Like many new-fangled, just-getting-off-the-ground technologies, most of the buzz is among the people excited about the technology, and pretty self-referential. I don't think this will die down to a tempest in a teapot, but I've been wrong before. Here are some of the intriguing sites I'm following:

- BlogStreet, Technorati and BlogDex categorizes and track blogging activity (each at their respective)
- Task Pane Central (http://taskpane.leafe.com/) a web site managed by Ed Leafe, has a new (September 2003) task pane for reading the MSDN RSS Feeds – the latest news from the Microsoft Developer Network.

- Meerkat is an online news aggregator devoted to the software development industry hosted by O'Reilly and Associates. http://www.oreillynet.com/meerkat/
- Other online aggregators include http://www.bloglines.com/, http://www.syndic8.com/, http://www.newsisfree.com/
- Desktop news aggregators are growing so quickly that posting a list in this document is pointless, obsolete before the pixels appear. Check your favorite search engine for "RSS News Aggregator." However, there are a couple of special interest. AmphetaDesk (http://www.disobey.com/amphetadesk/) (free) is a slick and mature package. NewsGator (http://www.newsgator.com/) ($29) integrates directly into Outlook.

Conclusion

RSS is a format we can use to post "news" and rich information from news producers to their consumers. This exercise also serves as a good example of practical use of XML as a data transport mechanism. I hope I've raised your interest in the development of RSS and the Semantic Web, and that you see some opportunities to use it in your business.

About the Speaker

Since 1987, Ted has worked full time as a software developer using Fox software. He has worked for state agencies, insurance companies and consulting firms, where he has built dozens of applications using nearly every feature of Visual FoxPro. He established Ted Roche & Associates in 2000. Ted Roche & Associates, LLC develops Web, client-server and LAN-based applications using Microsoft Visual FoxPro and other best-of-breed tools. Based in New Hampshire, his company offers consulting, training and mentoring, on-site and long-distance, as well as software development services. Ted is author of *Essential SourceSafe,* co-author of the award-winning *Hacker's Guide to Visual FoxPro* series, and a contributor to five other FoxPro books. In addition to numerous magazine articles, he is a popular speaker at conferences worldwide. Ted is a Microsoft Certified Solution Developer, Microsoft Certified System Engineer, and eight-time winner of the Microsoft Support Most Valuable Professional award.

References: Books and articles

Berners-Lee, Tim, James Hendler and Ora Lassila, "The Semantic Web," *Scientific American,* May 2001, also available at: http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2

Black, Steven, *The Hooks and Anchor Design Pattern*, http://www.stevenblack.com/HooksAndAnchorsDesignPattern.ASP

Blood, Rebecca, History of Weblogs, http://www.rebeccablood.net/essays/weblog_history.html

Bray, Tim. Tim's personal history of RDF: http://www.tbray.org/ongoing/When/200x/2003/05/21/RDFNet

Egger, Markus President, EPS Software Corporation, *XML Basics* whitepaper

Hammersley, Ben, *Content Syndication with RSS*, O'Reilly & Associates, 2003. See also http://www.oreilly.com/catalog/consynrss/

Pilgrim, Mark, *What is RSS?*, http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

Strahl, Rick, *Building distributed applications with XML messaging*, February 2001, http://www.west-wind.com/presentations/xmlmessaging/xmlmessaging.htm

Winer, Dave, catalog of RSS versions: http://backend.userland.com/rss

Tools

While nearly all of the functionality can be achieved with Visual FoxPro alone, additional tools can make the job easier. Here are a few I use:

Microsoft XML 4.0 Parser SDK (http://www.microsoft.com) documents the Microsoft XML DOM and the properties and methods to use.

Stylus Studio, http://www.stylusstudio.com, is a commercial application (MSRP ~$395) useful for editing and manipulating XML, XSLT, XQuery and Web Services. A built-in debugger allows you to trace execution of XSL. A WYSIWYG editor lets you design HTML from XML and then generates appropriate XSL transformations.

XMLEditPro v.2.0 (http://www.daveswebsite.com) is a shareware ($15) editor that lets you compose and examine XML, run XSL against it, and view the XML in color-coded text, tree, or browser views. Think Notepad vs. a full application. Fast and light.

Validators

You must test to make sure you can generate valid RSS. Even if you are only using it in-house, consider copying a file to a public web server and submitting it to these validators to ensure that it fits the standard:

http://feeds.archive.org/validator/

http://aggregator.userland.com/validator

http://www.ldodds.com/rss_validator/1.0/validator.html

Standards

W3C Date and time formats: http://www.w3.org/TR/NOTE-datetime

RDF: http://www.w3c.org/RDF/

RSS 1.0: http://web.resource.org/rss/1.0/modules/standard.html

RDF Site Summary 1.0 Modules; http://web.resource.org/rss/1.0/modules/

RSS 2.0: http://backend.userland.com/rss

RSS 0.9: http://backend.userland.com/rss091