RSS Feeds

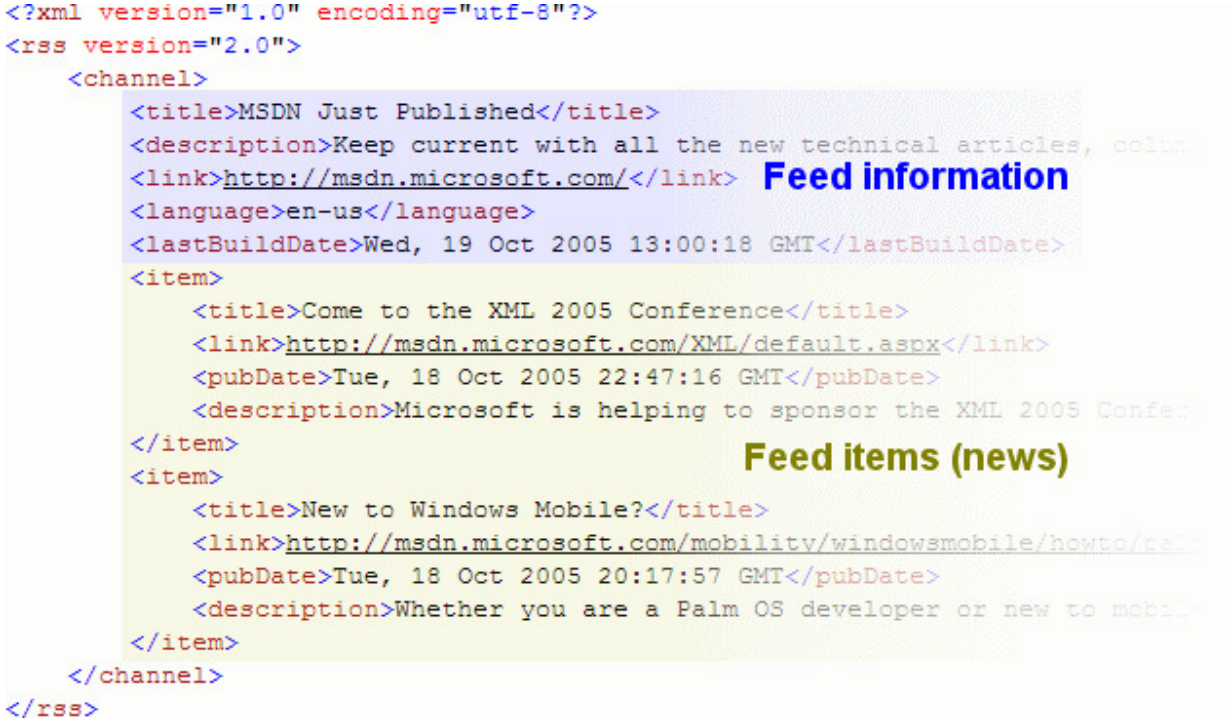
RSS (Really Simple Syndication) provides a convenient way to syndicate information from a variety of sources, including news stories, updates to a web site or even source code check-ins for a development project. Regardless of the purpose for which the RSS file is being used, by watching this XML file, you can quickly and easily see whenever an update has occurred.

Creating an RSS feed reader in C#

This tutorial will show you how to create your own RSS reader using the XmlTextReader in .NET. You'll learn how to read information about the RSS feed, retrieve its main content (the items) and output it in a ListView. The content of this tutorial was adapted from the http://www.geekpedia.com/tutorial147_Creating-an-RSS-feed-reader-in-Csharp.html .

The structure of an RSS feed

RSS feeds are everywhere, almost any website with dynamic content has at least one RSS feed for the visitors to subscribe to, so there's no question why would you need to know how to build an RSS feed reader. I don't think this is news to you, but an RSS feed is actually an XML file that follows a certain structure.

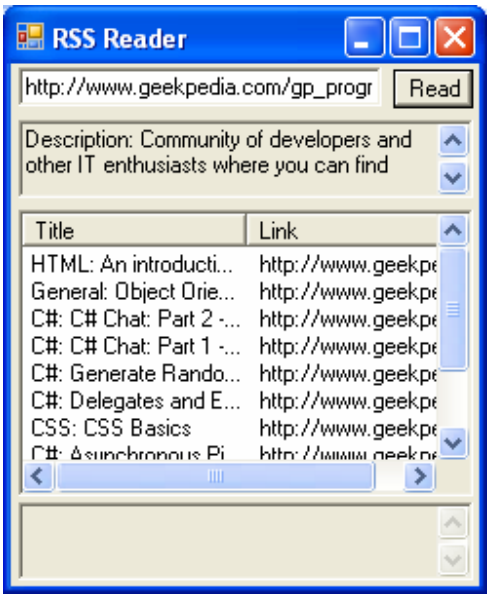



```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
    <channel>
        <title>MSDN Just Published</title>
        <description>Keep current with all the new technical articles,
        <link>http://msdn.microsoft.com/</link>
        <language>en-us</language>
        <lastBuildDate>Wed, 19 Oct 2005 13:00:18 GMT</lastBuildDate>
        <item>
            <title>Come to the XML 2005 Conference</title>
            <link>http://msdn.microsoft.com/XML/default.aspx</link>
            <pubDate>Tue, 18 Oct 2005 22:47:16 GMT</pubDate>
            <description>Microsoft is helping to sponsor the XML 2005
        </item>
        <item>
            <title>New to Windows Mobile?</title>
            <link>http://msdn.microsoft.com/mobility/windowsmobile/howto/
            <pubDate>Tue, 18 Oct 2005 20:17:57 GMT</pubDate>
            <description>Whether you are a Palm OS developer or new to
        </item>
    </channel>
</rss>
```

RSS feeds can have their own custom tags, or lack some of the tags, but the main structure should be the same: start with an **xml** tag, followed by an **xml-stylesheet** tag (if applicable) and then the **rss** tag which tells us it's an RSS feed. The nodes under the **channel** tag give information about the feed, such as its title and description. Then the **item** tags which have children tags which represent the main content of the feed: the **title**, **link** and **description** tags.

Create a .NET project

Create a C# device application project and drag controls from toolbox (in order to seem ike the following figure).



Coding the RSS reader

A bunch of variable we'll use:

```
XmlTextReader rssReader;
XmlDocument rssDoc;
XmlNode nodeRss;
XmlNode nodeChannel;
XmlNode nodeItem;
ListViewItem rowNews;
```

The nodes (*nodeRss*, *nodeChannel*, *nodeItem*) will be set to the actual nodes inside the XML file (*<rss>*,*<channel>* and *<item>*).
Most of the action takes place in the *click* event of the Read button (*button1*):

```
private void button1_Click(object sender, EventArgs e)
        {
            listView1.Items.Clear();
            // Create a new XmlTextReader from the specified URL (RSS feed)
            rssReader = new XmlTextReader(textBox1.Text);
            rssDoc = new XmlDocument();
            // Load the XML content into a XmlDocument
            rssDoc.Load(rssReader);
            // Loop for the <rss> tag
            for (int i = 0; i < rssDoc.ChildNodes.Count; i++)
            {
                // If it is the rss tag
                if (rssDoc.ChildNodes[i].Name == "rss")
                {
                    // <rss> tag found
                    nodeRss = rssDoc.ChildNodes[i];
                }
            }
            // Loop for the <channel> tag
            for (int i = 0; i < nodeRss.ChildNodes.Count; i++)
            {
```

```
                // If it is the channel tag
                if (nodeRss.ChildNodes[i].Name == "channel")
                {
                    // <channel> tag found
                    nodeChannel = nodeRss.ChildNodes[i];
                }
            }
            textBox2.Text= "Title: " + nodeChannel["title"].InnerText
                + "\r\nLanguage: " + nodeChannel["language"].InnerText
                + "\r\nLink: " + nodeChannel["link"].InnerText
                + "\r\nDescription: " + nodeChannel["description"].InnerText;

            // Loop for the <title>, <link>, <description> and all the other tags
            for (int i = 0; i < nodeChannel.ChildNodes.Count; i++)
            {
                // If it is the item tag, then it has children tags which we will add as
items to the ListView
                if (nodeChannel.ChildNodes[i].Name == "item")
                {
                    nodeItem = nodeChannel.ChildNodes[i];

                    // Create a new row in the ListView containing information from
inside the nodes
                    rowNews = new ListViewItem();
                    rowNews.Text = nodeItem["title"].InnerText;
                    rowNews.SubItems.Add(nodeItem["link"].InnerText);
                    listView1.Items.Add(rowNews);
                }
            }
        }
```

You can see here how we load the content of the supplied RSS feed (via *txtBox1*) into an **XmlDocument**.
Then we simply star looking for tags by looping through the nodes in search of the *<rss>* node. We never know how many nodes there are before the *<rss>* node, that's why we have to take each and every node and see if it's the one we want (*<rss>* in our case). Once we found it, we store its content (child nodes) within *nodeRss*. Now it would be the time to get information such as the RSS version the feed is using from the attributes of the *<rss>* tag, however we'll skip that since we want to keep this tutorial simple.
Now we loop again, this time in search of the *<channel>* tag.
We proceed with the same technique we used for the *<rss>* tag: store it as a **XmlNode**. Next thing you know, we are actually retrieving data from the RSS feed and outputing it using the labels on the form. More exactly, we retrieve information about the RSS feed from the first tags inside *<channel>*. We retrieve the Title, Language, Link and Description of the RSS feed, but we could retrieve much more such as the last update date or the feed generator, depending on the feed.
The next big step follows, loop through the child nodes (*<child>*), get the information inside them (*<title>* and *<link>*) and add it to the ListView.
This is enough for the RSS reader to work. Running the application, entering a valid feed isnide *textBox1* and pressing the Read Feed button will read the feed. It will take a few seconds to retrieve the feed's content from the Internet, and putting this on a different thread would be a good idea.
Now we have one more thing to do. When a row is selected inside the ListView, we'll want to give more details about that item in the feed, such as the description. So let's do that inside the *SelectedIndexChanged* event of the ListView:

```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
        {
            // When an item is selected
           if (listView1.SelectedIndices.Count == 1)
            {
                // Loop through all the nodes under <channel>
                for (int i = 0; i < nodeChannel.ChildNodes.Count; i++)
```

```
                {
                    // Until you find the <item> node
                    if (nodeChannel.ChildNodes[i].Name == "item")
                    {
                        // Store the item as a node
                        nodeItem = nodeChannel.ChildNodes[i];
                        // If the <title> tag matches the current selected item
                        for (int j = 0; j < listView1.Items.Count; j++)
                        {
                            if (nodeItem["title"].InnerText ==
listView1.Items[listView1.SelectedIndices[0]].SubItems[0].Text)
                            {
                                // It's the item we were looking for, get the description
                                textBox3.Text = nodeItem["description"].InnerText;
                                // We don't need to loop anymore
                                break;
                            }
                        }
                    }
                }
            }
        }
```

What we're doing here is loop through all the *<item>* tags and check to see if their *<title>* tag inside matches the value of the currently selected item in the ListView. So basically we search the XML file for the title of the selected item to find its *<description>* tag and show it inside the *textBox3*.