

# An XML Feed Reader the easy way

---

So you want to read some XML from a site in c# maybe iTunes or a feed, but it seems like a lot of coding, I am going to show you the easy way.

You will need a program so go off to this location and get the REST starter kit for Visual Studio. <http://aspnet.codeplex.com/releases/view/24644> install it and you will see on the “edit” menu a new option “Paste XML as Types”. You are going to use this later.

The first thing to do is choose a site find the URL of your favorite RSS feed and put it in the browser. This will give you an XML output. Now go to the source and select and copy it.

Open up a new project in Visual Studio for a class library. In the new class get rid of the class itself just leaving the namespace. Select between the curly braces and go to “edit/Paste Xml as Types”. All of the classes will be created to interpret XML automatically.

So now you have a class that can interpret RSS feed all you need now is a way to pull them in and insert them into the class or Deserialize XML to a class. For this we are going to use the XMLTextReader and the XMLSerializer classes.

Taking the classes that have just been created, find the constructor (which if it is an RSS feed it will be the rss class constructor) and put in this code.

```
/// <summary>
/// Constructor
/// </summary>
/// <param name="url">Url of the Feed</param>
public rss(string url)
{
    using (XmlTextReader reader = new XmlTextReader(url))
    {
        XmlSerializer serializer = new XmlSerializer(typeof(rss));
        rss buffer = (rss)serializer.Deserialize(reader);
        this.Channel = buffer.Channel;
        this.Version = buffer.Version;
        reader.Close();
    }
}
```

So now you can populate the class from a URL

What you need now is a way of doing the reverse turning a class into XML for this we can use the XMLSerializer again but this time Serialize to XML

To Read XML from the class add a property to your class

```
/// <summary>
/// Xml
/// </summary>
/// <returns>Xml String</returns>
[XmlIgnore]
public string Xml
{
    get
    {
        string buffer = string.Empty;
        using (MemoryStream stream = new MemoryStream())
        {
            XmlSerializerNamespaces ns = new XmlSerializerNamespaces();
            ns.Add("", "");
            XmlSerializer serializer = new XmlSerializer(typeof(CallXml));
            serializer.Serialize(stream, this, ns);
            buffer = Encoding.UTF8.GetString(stream.ToArray());
            stream.Close();
        }
        return buffer;
    }
}
```

This will take the contents of your class and output is as an XML string.

That's it you are done you have just created an RSS Reader. One thing that I like to do with my classes is to change the generated output for example arrays I replace with List eg:

```
/// <summary>
/// Channel Items
/// </summary>
private List[] item;
[XmlElement("item")]
public List[] Item
{
    get { return this.item; }
    set { this.item = value; }
}

/// <summary>
/// Channel Items
/// </summary>
private List<Item> item = new List<Item>();
[XmlElement("item")]
public List<Item> Item
{
    get { return this.item; }
    set { this.item = value; }
}
```

The reason for this becomes apparent when you want to use the class.

```
public void Test()
{
    Rss rss = new Rss("http://www.Feeds.com/Myfeed");
    Item item = new Item();
    item.Description = "My Description";
    rss.Channel.Item.Add(item);
    Console.Write(rss.Xml);
}
```

You would not need to instantiate the array just .Add() the item.

I also use attributes to rename the Elements and Attributes which makes for a cleaner class.

The only problem to watch out for is dates, sometimes the Paste XML as Type will set the date fields as Date type. This is OK until someone enters the date in their feed in the wrong format, which is quite common. The easiest solution is change it to a string type and it will always work.

Happy Coding