

在 Web 站点中创建和使用 Rss 源

Creating & Consuming Rss Feeds on Your Web Site

Bipin Joshi

编译: 张子阳

www.tracefact.net

jimmy_dev@163.com

主要参考:

[Creating Rss Feeds For Your Web Site](#)

[Consuming Rss Feeds On Your Web Site](#)

术语表

tag: 标签

markup: 标记

node: 结点

item: 条目

entry: 入口

Rss Reed: Rss 源

介绍

Rss是将你Web站点的内容与其他人分享的标准方式。Rss代表着：**Really Simple Syndication**。它不过是一个标准化的XML标记，用于描述你想要分享的内容。因此Rss是一个在你的内容准备好被其他用户所消费时被广泛接受的格式。一些使用Rss的范例站点有：www.asp.net、weblogs.asp.net 和 www.dotnetbips.com。Dotnetbips.com 通过 Rss 发布新添内容的列表，这个列表可能会被其他的站长放置在他们的站点或目录中。

Rss 的格式

如同我之前提到的，Rss 不过是有着一些特殊标签的 XML 标记。下面的标记展示了这样一个文档：

```
<rss version="2.0">
  <channel>
    <title>DotNetBips.com Latest Articles</title>
    <link>www.dotnetbips.com</link>
    <description>DotNetBips.com Latest Articles</description>
    <copyright>Copyright (C) DotNetBips.com. All rights reserved.</copyright>
    <generator>www.dotnetbips.com RSS Generator</generator>
    <item>
      <author>Bipin Joshi</author>
      <title>Using WebRequest and WebResponse</title>
      <link>http://www.dotnetbips.com/displayarticle.aspx?id=239</link>
      <description>Description here</description>
      <pubDate>Sun, 25 Jan 2004 12:00:00 AM GMT</pubDate>
    </item>
  </channel>
</rss>
```

让我们仔细看看每一个标记：

- <rss>: 根结点, 拥有一个 version(版本)属性, 最新版本是 2.0
- <channel>: rss 下的根结点, 可以再次包含<channel>结点。<channel>结点可以进一步包含<title>, <link>, <item>结点。
- <title>: 代表 Rss 源的标题。
- <link>: 代表着提供 Rss 源的站点的 URL。
- <description>: 关于这个 Rss 源的更多详细信息。
- <copyright>: 详细说明版权信息。
- <generator>: 说明产生这个 Rss 源的应用程序。

除了上面的这些标签, 还可以有一个或多个<item>标签。Item 标签代表着你想要分享的实际条目。比如, 文章、博客入口。每个<item>标记进一步包含下面这些子结点。

- <title>: 代表着这个条目的标题。比如: 文章标题。
- <author>: 代表着这个条目的作者。比如: 文章作者。
- <link>: 代表这个条目的 URL。比如: 文章的 URL。
- <description>: 包含着这个条目的描述信息。比如: 文章的摘要。
- <pubDate>: 这个标签包含着这个条目的发布日期。典型的日期格式是: Sun 28 Dec 2003 12:00:00 AM GMT.

采用的方法

OK, 我们已经对 Rss 的格式做了了解, 但是如何使用 .Net 生成 Rss 源? .Net 有许多 XML 相关的类。我们将从这些类中使用 XML Text Writer 来生成 Rss 源。但是我们应该开发出一个通用的解决方案以便在任何的 web 站点中都可以使用。这就意味着我们的代码必须独立于特定的数据库领域或者表。为了达到这个目的, 我们将要在 VS.NET 中创建一个类库。我们 Rss 的<item>标记的数据源将采用一个 Dataset, 这个 Dataset 通常填充自数据库表。这个类将有下面的属性和方法。

属性

- Outputstream: 一个源所投递到的 stream 对象。
- RssTitle: 代表<channel>标签下的<title>的特定值。
- PublisherUrl: 代表<channel>标签下的<link>标签。
- Description: 代表<channel>标签下的<description>值。
- Copyright: 代表<channel>标签下的<copyright>值。
- Generator: 代表<channel>标签下的<generator>值。
- ItemSource: 指定一个包含着 item 行的 Dataset 对象。
- ItemTitleField: 数据列, 代表<item>标签下的<title>标签
- ItemUrlField: 数据列, 代表<item>标签下的<link>标签。
- ItemDescriptionField: 数据列, 代表<item>标签下的<description>标签。
- ItemPublicationDateFiled: 数据列, 代表<item>标签下的<pubDate>标签。
- ItemAuthor: 数据列, 代表<item>标签下的<author>标签。

方法

- **PublishRss**: 这个静态方法将 Rss 标记写入到 outputStream(输出流)中。

下面列出的是上面这些属性和方法的完整程序清单。为了简单和快速地作个示范,我使用了公用字段,而没有使用属性。在实际的应用程序中,应该使用属性。

```
using System;
using System.IO;
using System.Data;
using System.Xml;
using System.Collections.Generic;
using System.Text;

public class Rss {
    public Stream OutputStream;
    public string RssTitle;
    public string PublisherUrl;
    public string Description;
    public string Copyright;
    public string Generator;
    public DataSet ItemSource;
    public string ItemTitleField;
    public string ItemUrlField;
    public string ItemDescriptionField;
    public string ItemPublicationDateField;
    public string ItemAuthor;

    public static void PublishRss(Rss r){
        XmlTextWriter writer = new XmlTextWriter(r.OutputStream, Encoding.UTF8);
writer.WriteStartDocument();

        writer.WriteStartElement("rss");
writer.WriteAttributeString("version", "2.0");
writer.WriteStartElement("channel");
writer.WriteElementString("title", r.RssTitle);
writer.WriteElementString("link", r.PublisherUrl);
writer.WriteElementString("description", r.Description);
writer.WriteElementString("copyright", r.Copyright);
writer.WriteElementString("generator", r.Generator);

        foreach (DataRow row in r.ItemSource.Tables[0].Rows) {
            writer.WriteStartElement("item");
writer.WriteElementString("author", row[r.ItemAuthor].ToString());
writer.WriteElementString("title", row[r.ItemTitleField].ToString());
writer.WriteElementString("link", row[r.ItemUrlField].ToString());
```

```

        writer.WriteElementString("description",
row[r.ItemDescriptionField].ToString());
        writer.WriteElementString("pubDate",
Convert.ToDateTime(row[r.ItemPublicationDateField]).ToString("dd MMM yyyy hh:mm:00
"));
        writer.WriteEndElement();
    }

    writer.WriteEndElement();
    writer.WriteEndElement();
    writer.Flush();
}
}

```

NOTE: 这里 pubDate 的日期格式很重要, 当你按上面的代码对 pubDate 进行格式转换的时候, 假如数据库中是 2007-9-14 9:58, 那么在英文操作系统下, 会转换成 “14 Sep 2007 9:58”, 这个是没有问题的。但在中文操作系统下, 就变成了 “14 九月 2007 9:58”。导致的结果就是在 IE7 中点开 rss 源的时候, 发现日期没有显示。如果你不进行数据格式转换, 简单的使用一个 ToString(), 结果仍是如此。这里, 我是写了一个方法, 对它进行了格式转换:

```

// ... 省略 ...
writer.WriteElementString("pubDate", GetRssDate(row[r.ItemPublicationDateField]));
// ... 省略 ...

public static string GetRssDate(Object date) {
    DateTime rssDate = Convert.ToDateTime(date);
    string[] monthName = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
"Sep", "Oct", "Nov", "Dec" };

    StringBuilder sb = new StringBuilder();
    sb.Append(rssDate.Day);
    sb.Append(" ");
    sb.Append(monthName[Convert.ToInt32(rssDate.Month) - 1]);
    sb.Append(" ");
    sb.Append(rssDate.Year);
    sb.Append(" ");
    sb.Append(rssDate.ToLongTimeString());

    return sb.ToString();
}

```

我看到 ToString() 有一个重载了的方法, 接受一个 IFormatProvider 类型的参数, 这个应该是进行格式转换的标准方式, MSDN 的范例有点长了, 就没有研究下去。如果哪位朋友对这里有好的解决方法, 可以在回复在评论中, Thanks。

创建属性就像上面显示的那样容易。PublishRss() 方法是我们这里关心的核心内容。我们创

建了 System.Xml.XmlTextWriter 类的实例。这个类是撰写 XML 文档的快速方式。在这个例子中，我们传递进一个 OutputStream 对象，并确定编码(UTF-8)。然后我们开始写入这个文档的不同部分。我们使用 XmlTextWriter 类的下面这些方法。

- WriteStartDocument(): 这个方法写入 XML 1.0 版本的声明。也就是：
<?xml version="1.0" encoding="utf-8"?>。当不写这个声明的时候，在 FireFox 中虽然可以订阅，但是看不到任何条目，也不能进行更新。IE7 下正常。
- WriteStartElement: 这个方法写入指定标签的起始标记。
- WriteAttributeString: 这个方法为当前打开的标签写入属性。
- WriteElementString: 这个方法写入一个起始标记和一个结束标记，以及起始和结束标记之间的文本。
- WriteEndElement: 这个方法写入当前打开标记的结束标记。不需要在这里指明结束标记的名字，因为在每次嵌套的时候都会在内部(NOTE: 属于底层机制)设定。
- Flush: 这个方法将所有缓存的 output 清出到目的位置。

注意，你必须恰当地调用 WriteStartElement() 和 WriteEndElement() 方法以生成格式良好的(well formed)XML 文档。

创建 Asp.Net Web 窗体

现在我们已经创建好了一个通用类，我们可以在我们的 Web 窗体中使用它。假设我们将以 Rss 源形式发布的数据存储在一个表格(Article)中，这个表格的结构如下：

- Title - Varchar(255)
- Description - Varchar(1000)
- Url - Varchar(255)
- Author - Varchar(50)
- Pubdate - DateTime

以 DataSet 形式获取 Table 内容

我们将在 Asp.Net Web 应用程序中创建一个 Rss.aspx 文件，在 CodeBehind 中创建一个 GetDataSet() 方法。这个方法使用 DataAdapter 来填充一个 Dataset。

```
public DataSet GetDataSet() {  
    SqlConnection conn = new SqlConnection("你的连接字符串");  
    string sql = "Select * From Article Order By ArticleId Desc";  
    SqlDataAdapter da = new SqlDataAdapter(sql,conn);  
    DataSet ds = new DataSet();  
  
    da.Fill(ds, "Article");  
    return ds;  
}
```

接着，我们创建一个 Rss 类的实例，设置它的各个属性，然后调用 GetDataSet() 方法获取

DataSet 对象。

```
protected void Page_Load(object sender, EventArgs e){
    DataSet ds = GetDataSet();
    Rss rss = new Rss();
    rss.OutputStream = Response.OutputStream;
    rss.RssTitle = "DotNetBips.com Latest Articles";
    rss.PublisherUrl = Request.Url.Host;
    rss.Description = "DotNetBips.com - Applying .NET";
    rss.Copyright = "Copyright (C) DotNetBips.com.";
    rss.Generator = "DotNetBips.com RSS Generator";
    rss.ItemSource = ds;
    rss.ItemTitleField = "Title";
    rss.ItemDescriptionField = "Description";
    rss.ItemPublicationDateField = "Pubdate";
    rss.ItemUrlField = "Url";
    rss.ItemAuthor = "Author";
    Response.ContentEncoding = System.Text.Encoding.UTF8;
    Response.ContentType = "text/xml";
    Rss.PublishRss(rss);
    Response.End();
}
```

当我们获得 DataSet 后,将 ItemSource 属性设置为这个 DataSet。另外,我们再设置 Response 对象的 ContentEncoding 和 ContentType 属性。然后,调用 PublishRss() 方法,将这个 Rss 类的实例传递进去。

OK, 现在我们在 IE 中浏览 Rss.aspx 页面, 应该可以看到如下图所示的画面:



NOTE: 注意, 此时 创建 Rss 源的类名和 Rss.aspx CodeBehind 中的 Page 类名将会一样, 都是 Rss, 所以需要手动修改 Rss.aspx.cs 中的 Page 类名, 比如, 改为_Rss, 则代码如下:

```
public partial class _Rss : System.Web.UI.Page
```

同时, 修改 HTML 页面的 inherits :

```
Inherits="_Rss"
```

消费 Rss 源

创建了Rss源以后, 其他的站点可以消费这个Rss源。我会以创建一个显示 www.asp.net 最新文章Web窗体来作为范例。

NOTE: 我想应该是因为对于发布 Rss 源的站点来说, 使用源的站点是消费者 Consumer, 所以英文技术文章中使用源通常都用 Consume 这个词, 而不用 Use。

为了能通过一个URL来读取XML的数据, 我们当然可以使用 WebRequest 和 WebResponse 对象(参考我的文章 [Using WebRequest and WebResponse](#))。然而, 有一个更简单的方法 - DataSet。

DataSet 类有一个叫做 ReadXml () 的方法, 可以从硬盘的文件或者 URL 中读取 XML 数据。这个方法读取数据并自动为我们生成所需的 DataTable。

```
DataSet ds = new DataSet();  
ds.ReadXml("http://127.0.1.1/rss.aspx");
```

这里, 我们创建了一个 DataSet 的实例, 并且通过传递 URL 参数来调用 ReadXml () 方法。可以根据需要来改变传进去的 URL 参数。

Note: 我房子暂时没有上网, 所以就使用前面所创建的本地 Rss 源作为演示。

ReadXml()方法生成的表格

如果你期望ReadXml () 方法会生成一个DataTable, 其中包含着链接的列表, 你会惊奇地发现实际上会生成三个DataTable。在 [RSS 的格式](#) 这一小节的XML标记中, 我们看到标记是嵌套的, DataSet会在读取数据的时候自动创建相关表。它也会为每个DataTable创建ID字段以便他们可以相互链接。

在这个例子中, 你会得到下面构架的 DataTable:

RSS

- Rss_Id
- Version

Channel

- Title

- Link
- Description
- Language
- Generator
- Channel_Id
- Rss_Id

Item

- Creator
- Title
- Link
- PubDate
- Guid
- Description
- Channel_Id

注意一些字段，比如 Creator 和 Guid 并没有在 Rss 标记中出现。另外注意 DataSet 是如何添加 Rss_Id 和 Channel_Id 这样的字段来关联 DataTable 的。

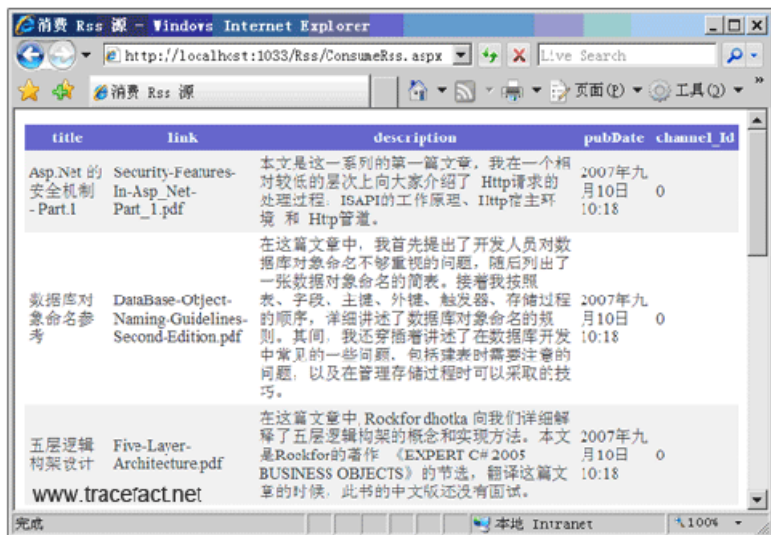
现在我们已经知道了表的结构，让我们编写一些代码来在 GridView 中显示这些数据。

在 GridView 中显示数据

从 Rss 源中获得的详细信息来看，第三张表是最重要的，因为它包含了实际的链接数据。这里我们将我们的 GridView 与第三个 DataTable 进行绑定。

```
GridView1.DataSource = ds.Tables[2].DefaultView;
GridView1.DataBind();
```

一旦你在 PageLoad 事件中调用这段代码，可以看到类似下面这样的屏幕截图：



加入浏览器支持

IE7 和 FireFox 都提供了对 Rss 的支持，为了使浏览器提供 Rss 支持，你必须先告诉它们你的站点创建了 Rss 源。只需要在<head></head>标签中加入如下代码即可：

```
<link rel="alternate" type="application/rss+xml" title="Your Web Site's RSS Feed Title " href="http://www.yourdomain.net/rss.aspx" />
```

这时再打开加入上面<link>的页面，会发现 IE7 的 RSS 图标由灰色变成了明亮的桔红色。

总结

在这篇文章中，我们了解了什么是 RSS，以及如何为你的站点生成 RSS 源。我们使用 XmlTextWriter 类来创建 Rss 标记。我们创建了一个通用类，以便它可以在任何 Web 应用程序中使用。

随后，我们了解了如何使用 DataSet 来消费 Rss 源。Rss 源是一个嵌套的 XML 标记，DataSet 自动创建彼此相关的 DataTable。第三张表 (ITem) 包含了 Rss 源的核心数据。

希望这篇文章能给你带来帮助。