C

Syndication

Syndication is a mechanism that allows the data modifications done in one website to be reflected in all other websites subscribed to it. An example of a syndicated website is the news website where information is regularly updated to provide its visitors with the latest updates. The updated information is provided to the user in the form of syndication feeds, which are subscribed to the website providing syndication. A feed can be defined as a small chunk of data that is connected to a syndicated website and reflects the changes made in that website. In a feed, items are usually displayed on the basis of reverse time order, which means the item (usually news) on the top of a series of feed items is the latest one. The metadata of feed consists of title, author, and Uniform Resource Locator (URL). The website user subscribed to a syndicated website does not need to check the updates regularly as the syndicated website automatically updates the information, such as title, link, and brief extracts of the contents available on the website.

.NET Framework allows you to expose syndication feeds using Windows Communication Foundation (WCF) services. WCF supports two types of syndication feeds, Really Simple Syndication (RSS) and Atom. The content presented in these formats is called an RSS feed or Atom feed. In addition, the WCF service provides an extension for the syndication feature in the `System.ServiceModel.Syndication namespace`. The classes provided by this namespace are used to read and write the RSS and Atom feeds.

In this appendix, you first learn about the types of syndication formats—RSS and Atom. Next, you learn to implement syndication in .NET Framework. You also learn to read a syndication feed from a website. Finally, you learn to create a syndication feed.

Understanding Types of Feed Formats

As already discussed, in .NET, RSS and Atom are the two most popular types of syndication feed formats supported by WCF. Both RSS and Atom are XML-based format types that are widely used to publish information between different websites on the Web. XML stands for Extensible Markup Language. Now, let's discuss about these two types of feeds in detail.

RSS

RSS is one of the most commonly-used feed type to publish the data that needs to be updated or changed frequently, such as blog articles and news. RSS type feed is published in an XML format that can read from many websites with the help of the aggregator program. The various versions of RSS are as follows:

- **RSS 0.9**—Refers to the version that is first released by Netscape in March 1999 for their My.Netscape.Com portal. In this version, RSS stands for Resource Description Framework (RDF) Site Summary.
- **RSS 0.91**—Refers to the version, released by Netscape in July 1999, in which the RSS format is simplified by removing the RDF elements. In this version, RSS is renamed as Rich Site Summary (RSS).
- **RSS 1.0**—Refers to the version that reclaimed the name of RSS as RDF Site Summary and reintroduced the RDF support in addition with the XML namespaces. This version was released in December 2000.

- **RSS 2.0**—Refers to the current version of RSS. It is released in September 2002 and is further improved by various organizations and teams.

The RSS 2.0 version introduced various new elements, such as <cloud> and <guid>. In addition, the restriction imposed on the <link> and <url> elements in the previous versions of RSS are removed in this version. The restriction was that the data of these elements must start with only http or ftp that is removed in RSS 2.0.

Atom

Atom, another type of feed, was developed as an alternative to the RSS feed due to some limitations and flaws in RSS, such as its necessity to remain compatible with the previous version and lack of on-going innovation. Atom is an the Internet Engineering Task Force (IETF) standard that develops and promotes Internet standards in coordination with other standards bodies, such as World Wide Web Consortium (W3C) and International Organization for Standardization (ISO)/ International Electrotechnical Commission (IEC). The main difference between an Atom feed and an RSS feed is that you need to explicitly declare the format in which the content should be displayed; whereas, the content in an RSS feed can contain both plain text or Hypertext Markup Language (HTML) tags.

Let's now learn about implementing syndication in .NET.

Implementing Syndication in .NET

In .NET, WCF provides the `System.ServiceModel.Syndication` namespace that is used to implement the feature of syndication in an application. The `System.ServiceModel.Syndication` namespace contains various classes that can be used to retrieve the syndication feeds from other websites. In addition, you can also use these classes to create syndication feeds and offer them to the readers.

Table C.1 lists some of the classes available in the System.ServiceModel.Syndication namespace:

Table: C.1: Describing the classes available in the System.ServiceModel.Syndication Namespace

Class	Description
Atom10FeedFormatter	Serializes a SyndicationFeed instance to and from Atom 1.0 format
Atom10ItemFormatter	Serializes a SyndicationItem instance to and from RSS 2.0 format
Rss20FeedFormatter	Serializes a SyndicationFeed instance to and from RSS 2.0 format
Rss20ItemFormatter	Serializes a SyndicationItem instance to and from Atom 1.0 format
SyndicationCategory	Specifies a class that represents the category of a syndication feed
SyndicationFeed	Specifies a top-level feed object, <feed>, in Atom 1.0 and <rss> in RSS 2.0
SyndicationFeedFormatter	Represents an abstract class used as a base class for other formatters, for example Atom10FeedFormatter
SyndicationItem	Specifies a feed item, for example an RSS <item> or an Atom <entry>
SyndicationItemFormatter	Represents an abstract class used as a base class for other formatters, for example, Atom10ItemFormatter
SyndicationLink	Specifies a feed item, for example an RSS <item> or an Atom <entry>
SyndicationPerson	Specifies an author or contributor of syndication content
SyndicationVersions	Specifies a class that represents the syndication versions

Lets' now learn about reading the feeds.

Creating an Application to Retrieve the RSS Feeds

In this section, you learn to create a WPF application to retrieve the syndication feeds from a website and display it in the application.

Perform the following steps to create a WPF application, named FeedReader (also available on CD-ROM):

1. Open the `MainWindow.xaml` file of the FeedReader application by double-clicking the file in Solution Explorer.
2. Add the code, shown in Listing C.1, in the `MainWindow.xaml` file to design the interface of the application:

Listing C.1: Showing the Code for the `MainWindow.xaml` File

```
<Window x:Class="FeedReader.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{Binding Path=Title.Text}" Height="427" Width="650">
    <DockPanel x:Name="Content">
        <Grid  DockPanel.Dock="Top">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="60" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="100" />
            </Grid.ColumnDefinitions>
            <Label Grid.Column="0" Margin="5">Enter Feed URL:</Label>
            <TextBox Grid.Column="1" x:Name="FeedUrl" MinWidth="150" Margin="5">
            </TextBox>
            <Button Grid.Column="2" Margin="5" MinWidth="80"
                    Click="retrieveFeed">Retrieve Feed</Button>
        </Grid>
        <!--<StackPanel Orientation="Vertical" DockPanel.Dock="Top"
         Background="LightGreen" x:Name="heading"></StackPanel>-->
        <ListBox MinWidth="120"  DockPanel.Dock="Left" ItemsSource="{Binding}"
         Style="{StaticResource lstStyle}"
           IsSynchronizedWithCurrentItem="True" HorizontalContentAlignment="Stretch"  />
        <DockPanel x:Name="pagecontent">
            <Frame Source="{Binding Path=Links[0].Uri}" />
        </DockPanel>
    </DockPanel>
</Window>
```

The design view of the `MainWindow.xaml` file after adding the code of Listing C.1 is shown in Figure C.1:

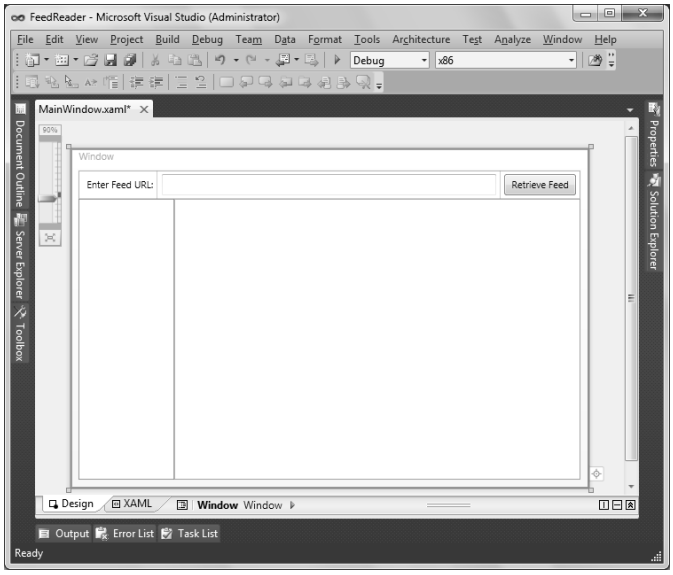

Figure C.1: Showing the Design View of the MainWindow.xaml File

3. Right-click the name of the application in Solution Explorer and select the Add Reference option from the context menu that appears. This opens the Add Reference dialog box, as shown in Figure C.2:

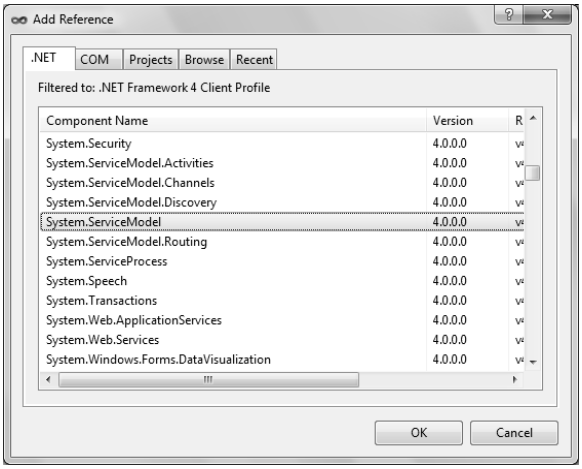

Figure C.2: Showing the Add Reference Dialog Box

4. Select the `System.ServiceModel` component from the .NET tab (Figure C.2) and click the OK button. This adds the `System.ServiceModel.dll` library in your application.
5. Open the code-behind file of the `MainWindow.xaml` file (`MainWindow.xaml.cs`), and add the code shown in Listing C.2:

Listing C.2: Showing the Code for the `MainWindow.xaml.cs` File

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Xml;
using System.Net;
using System.ServiceModel.Syndication;
namespace FeedReader
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        private void retrieveFeed(object sender, RoutedEventArgs e)
        {
            try
            {
                using (XmlReader reader = XmlReader.Create(FeedUrl.Text))
                {
                    var formatter = new Rss20FeedFormatter();
```

```
                    formatter.ReadFrom(reader);
                    this.DataContext = formatter.Feed;
                    this.Content.DataContext = formatter.Feed.Items;
                }
            }
            catch (WebException ex)
            {
                MessageBox.Show(ex.Message, "Syndication Reader");
            }
        }
    }
}
```

6. Open the `App.xaml` file of the FeedReader application and add the code, shown in Listing C.3, in the `App.xaml` file:

Listing C.3: Showing the Code for the `App.xaml` File

```
<Application x:Class="FeedReader.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <Style x:Key="lstStyle" TargetType="{x:Type ListBox}">
            <Setter Property="ItemTemplate">
                <Setter.Value>
                    <DataTemplate>
                            <Label Content="{Binding Title.Text}" />
                    </DataTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Application.Resources>
</Application>
```

6. Press the F5 Key to run the FeedReader application. The output of the application appears, as shown in Figure C.3:

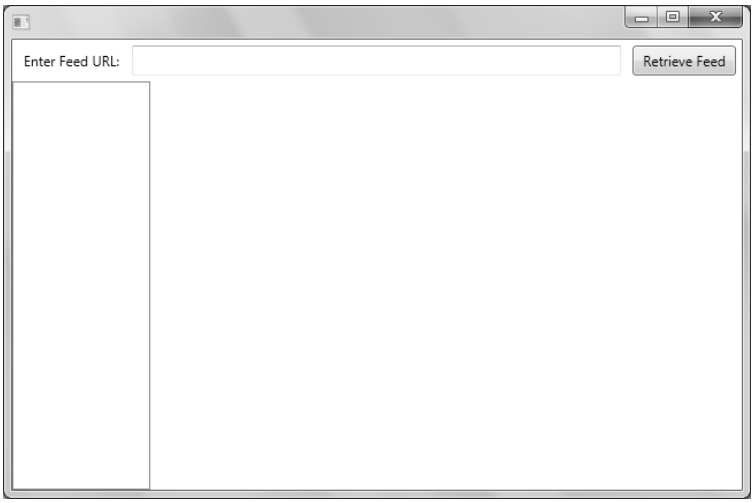

Figure C.3: Showing the Output of the FeedReader Application

7. Enter the URL from where you want to retrieve the feed in the Enter Feed URL text box and click the Retrieve Feed button (Figure C.3). In this case, we have retrieved the news feeds provided by the Yahoo website, as shown in Figure C.4:

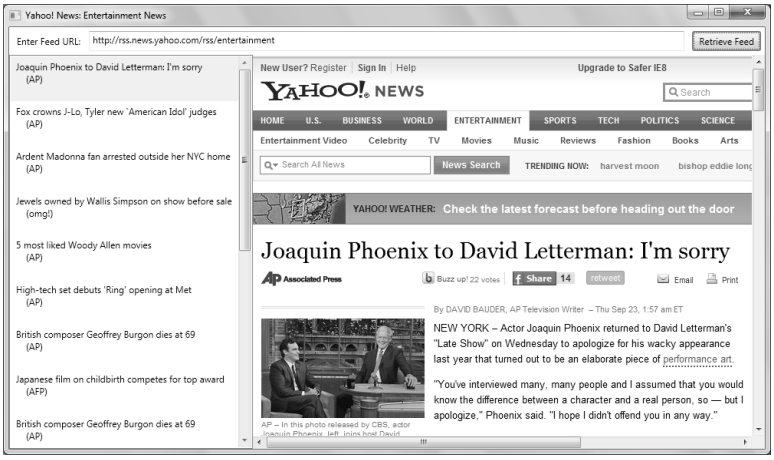

Figure C.4: Showing the Retrieved Feeds

Using this application, you can retrieve the RSS feeds of any website by entering its URL in the Enter Feed URL text box.

Now, let's learn to create feeds in the next section.

Creating an Application to Create the Syndication Feeds

In this section, we create an application, named SyndicationServiceApplication (also available on CD-ROM), in which we create syndication feed to offer them to the users. The Syndication Service Library template is provided by the Visual Studio 2010 to create the syndication feeds. Now, perform the following steps to create a syndication feed:

1. Open Visual Studio 2010 and select the File → New → Project option from the main menu. This opens the New Project dialog box (Figure C.5).
2. Select Visual C# → WCF node in the Installed Templates section and select the Syndication Service Library template from the list of templates, as shown in Figure C.5:

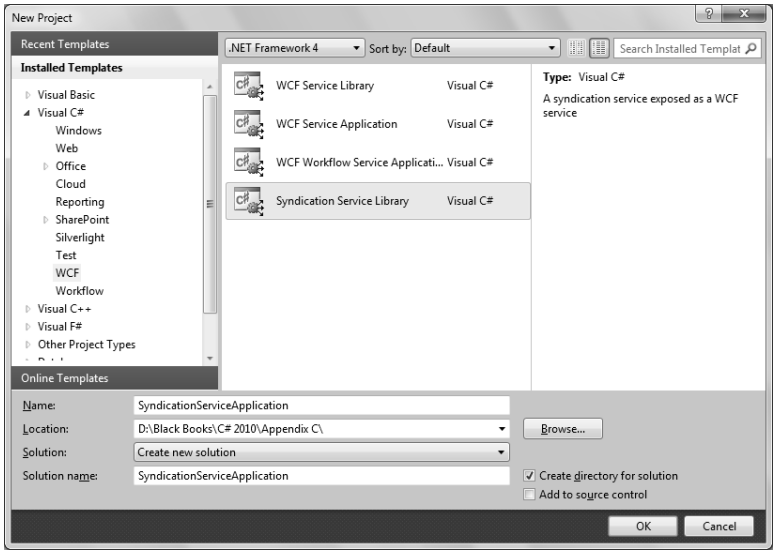

Figure C.5: Showing the New Project Dialog Box

3. Enter the name of the application, as SyndicationServiceApplication, in the Name text box and the location in the Location combo box (Figure C.5).
4. Click the OK button of the New Project dialog box (Figure C.5) to create the application.
5. Replace the existing code of the `Feed1.cs` file, created automatically at the time of creating the application, with the code shown in Listing C.4:

Listing C.4: Showing the Code for the `Feed1.cs` File

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Syndication;
using System.ServiceModel.Web;
using System.Text;

namespace SyndicationServiceApplication
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class
         name "Feed1" in both code and config file together.
    public class Feed1 : IFeed1
    {
        public SyndicationFeedFormatter CreateFeed()
        {
            // Create a new Syndication Feed.
            SyndicationFeed feed = new SyndicationFeed("Books Feed", "This is a WCF
                 Syndication Feed", null);
            List<SyndicationItem> items = new List<SyndicationItem>();

            // Create a new Syndication Item.
            SyndicationItem item = new SyndicationItem("C# 2010 Black Book", "This book
                 contains all the concepts related to C#.", null);
            items.Add(item);

            SyndicationItem item1 = new SyndicationItem("ASP.NET 4.0 Black Book", "This
                 book contains all the concepts related to ASP.NET 4.0.", null);
            items.Add(item1);

            SyndicationItem item2 = new SyndicationItem("ASP.NET 4.0 Project Book", "This
                 book contains various enterprise level projects created using ASP.NET
                 4.0.", null);
            items.Add(item2);
            feed.Items = items;

            // Return ATOM or RSS based on query string
            // rss ->
         http://localhost:8732/Design_Time_Addresses/SyndicationServiceApplication/Feed1/
            // atom ->
http://localhost:8732/Design_Time_Addresses/SyndicationServiceApplication/Feed1/?format=atom
            string query =
    WebOperationContext.Current.IncomingRequest.UriTemplateMatch.QueryParameters["format"];
            SyndicationFeedFormatter formatter = null;
            if (query == "atom")
            {
                formatter = new Atom10FeedFormatter(feed);
            }
            else
            {
                formatter = new Rss20FeedFormatter(feed);
            }
```

```
                return formatter;
            }
        }
    }
```

6. Press the F5 key to run the application. The output of the application displays the items of the syndication feeds, as shown in Figure C.6:

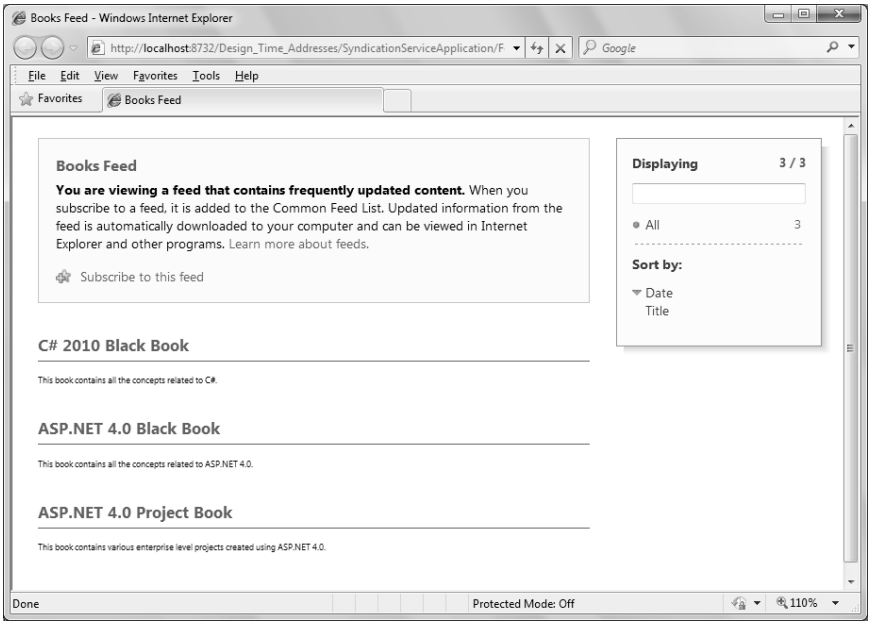

Figure C.6: Showing the Output of the SyndicationServiceApplication Application

With this, we have come to the end of this appendix.