47

Синдикация

В ЭТОЙ ГЛАВЕ...

- Пространство имен `System.ServiceModel.Syndication`
- Чтение синдицируемых каналов
- Предоставление синдицируемых каналов

Часто возникает желание публиковать какие-то структурированные данные, которые время от времени изменяются. Сегодня на многих веб-сайтах присутствуют значки RSS (Really Simple Syndication — действительно простая синдикация) и Atom (Another Syndication Format — еще один формат синдикации), которые позволяют подписываться на чтение каналов. RSS представляет собой формат XML, позволяющий выполнять синдикацию (объединение) информации. Блоги сделали RSS очень популярным. На эту XML-информацию очень просто подписаться с использованием специальных средств для чтения RSS.

В наши дни RSS применяется не только с блогами, но и со многими другими источниками данных, такими как онлайновые журналы новостей. Любые данные, которые изменяются с течением времени, могут предоставляться посредством RSS или появившегося после него протокола Atom. Продукты Microsoft Internet Explorer и Microsoft Outlook обладают встроенными инструментами для чтения данных RSS и Atom.

Пространство имен `System.ServiceModel.Syndication` в Windows Communication Foundation (WCF) содержит расширение для поддержки средств синдикации. В этом пространстве имен предлагаются классы, служащие для чтения и записи каналов RSS и Atom.

В настоящей главе будет показано, как создавать приложения для чтения и предоставления синдицируемых каналов.

Краткий обзор пространства имен `System.ServiceModel.Syndication`

Для целей синдикации используется пространство имен `System.ServiceModel.Syndication`, в котором определены классы, обеспечивающие синдикацию данных в форматах RSS и Atom.

Технология RSS (Really Simple Syndication — действительно простая синдикация) получила свое нынешнее название лишь после выхода версии RSS 2.0. В предыдущих версиях она называлась Resource Description Framework (RDF) Site Summary и Rich Site Summary. Первая версия RDF была создана компанией Netscape для описания содержимого ее портального сайта. Она прославилась тогда, когда в издательстве *New York Times* в 2002 г. с ее помощью начали предлагать читателям возможность подписываться на RSS-каналы новостей.

Формат синдикации Atom был разработан после RSS, и предложен в качестве стандарта в документе RFC 4287, который доступен по адресу `www.ietf.org/rfc/rfc4287.txt`. Главное отличие между RSS и Atom связано с видом содержимого, которое может определяться вместе с элементом.

В RSS внутри элемента описания может содержаться как простой текст, так и текст в формате HTML, тип которого не играет никакой роли для считывающего приложения. В Atom для содержимого должен быть определен конкретный тип с помощью атрибута типа, а также разрешено использовать XML-содержимое с определенными пространствами имен.

Рис. 47.1. *Логотип, с помощью которого представляются каналы RSS и Atom*

На рис. 47.1 показан логотип, который применяется для представления каналов RSS и Atom. Если на веб-сайте присутствует такой логотип, значит, на нем предлагается какой-нибудь канал RSS или Atom.

В табл. 48.1 перечислены классы и элементы, которые можно применять для создания синдицируемого канала. Эти классы не зависят от типа синдикации, RSS или Atom.

Таблица 47.1. Классы и элементы, доступные для создания каналов RSS и Atom

Класс	Описание
`SyndicationFeed`	Класс `SyndicationFeed` представляет элемент самого высокого уровня в канале. В случае Atom таковым является `<feed>`, а в случае RSS — элемент `<rss>`. Статический метод `Load()` позволяет читать канал с использованием `XmlReader`. Свойства этого класса, такие как `Authors`, `Categories`, `Contributors`, `Copyright`, `Description`, `ImageUrl`, `Links`, `Title` и `Items`, позволяют определять дочерние элементы.
`SyndicationPerson`	Класс `SyndicationPerson` представляет персону с помощью свойств `Name`, `Email` и `Uri`, которая может быть присвоена коллекциям `Authors` и `Contributors`.
`SyndicationItem`	Любой канал состоит из множества элементов. Свойства элемента включают `Authors`, `Contributors`, `Copyright` и `Content`.
`SyndicationLink`	Класс `SyndicationLink` представляет ссылку внутри канала или элемента. Имеет свойства `Title` и `Uri`.
`SyndicationCategory`	Элементы в канале могут группироваться по категориям. Ключевые слова категории могут быть установлены в свойствах `Name` и `Label` класса `SyndicationCategory`.
`SyndicationContent`	`SyndicationContent` — это абстрактный базовый класс, который описывает содержимое элемента. Содержимым может быть HTML, простой текст, XHTML, XML или URL. Содержимое описывается с помощью конкретных классов `TextSyndicationContent`, `UrlSyndicationContent` и `XmlSyndicationContent`.
`SyndicationElementExtension`	С помощью элемента расширения можно добавлять дополнительное содержимое. Класс `SyndicationElementExtension` может использоваться для добавления информации в канал, категорию, объект, персону, ссылку и элемент.

Для назначения каналу формата RSS и Atom можно использовать классы, унаследованные от `SyndicationFeedFormatter` и `SyndicationItemFormatter`. В .NET 4 для форматирования каналов предназначены классы `Atom10FeedFormatter` и `Rss20FeedFormatter`, а для форматирования содержащихся в них элементов — классы `Atom10ItemFormatter` и `Rss20ItemFormatter`.

Пример чтения синдицируемых каналов

В качестве первого примера рассмотрим приложение для чтения синдицируемых каналов, которое называется `SyndicationReader` и имеет пользовательский интерфейс, разработанный с помощью WPF (Windows Presentation Format). На рис. 47.2 показано, как выглядит интерфейс этого приложения.

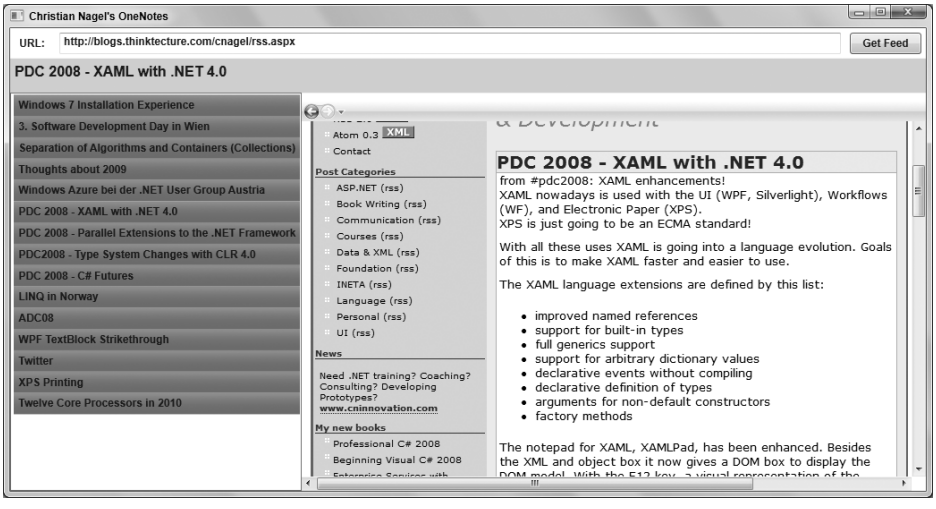

Рис. 47.2. *Интерфейс приложения* `SyndicationReader`

Для использования API-интерфейса Syndication в приложение добавлена ссылка на сборку `System.ServiceModel`. Исполняющему роль обработчика событий методу `OnGetFeed()` назначается событие `Click` кнопки с текстом Get Feed (Получить канал). Код, необходимый приложению для чтения, выглядит действительно очень просто. Сначала производится считывание XML-содержимого из RSS-канала в класс `XmlReader` пространства имен `System.Xml`. Далее с помощью метода `ReadFrom()` объект `XmlReader` передается классу `Rss20FeedFormatter`. После этого для осуществления привязки данных свойство `Feed`, возвращающее `SyndicationFeed`, присваивается свойству `DataContext` объекта `Window`, а свойство `Feed.Items`, возвращающее `IEnumerable<SyndicationItem>`, присваивается свойству `DataContext` контейнера `DockPanel`.

```
private void OnGetFeed(object sender, RoutedEventArgs e)
{
  try
  {
    using (var reader = XmlReader.Create(textUrl.Text))
    {
      var formatter = new Rss20FeedFormatter();
      formatter.ReadFrom(reader);
      this.DataContext = formatter.Feed;
      this.feedContent.DataContext = formatter.Feed.Items;
    }
  }
  catch (WebException ex)
  {
    MessageBox.Show(ex.Message, "Syndication Reader");
  }
}
```

Фрагмент кода ***SyndicationReader\SyndicationReader.xaml.cs***

Код XAML, необходимый для определения пользовательского интерфейса, показан ниже. Здесь свойство `Title` класса `Window` сначала привязывается к свойству `Title.Text` объекта `SyndicationFeed` для обеспечения отображения заголовка канала.

Далее определяется элемент управления `DockPanel` по имени `heading`, в котором содержится один элемент управления `Label`, привязываемый к `Title.Text`, и один элемент

управления Label, привязываемый к Description.Text. Поскольку эти элементы управления Label содержатся внутри элемента управления DockPanel по имени feedContent, а этот feedContent привязывается к свойству Feed.Items, они будут отвечать за отображение заголовка и описания элемента, выбранного в текущий момент.

Список элементов отображается в элементе управления ListBox, в котором для привязки элементов управления Label к Title применяется ItemTemplate

В элементе DockPanel по имени content содержится элемент Frame, в котором свойство Source связывается с первой ссылкой элемента. Благодаря этому элемент управления Frame использует элемент управления типа веб-браузера для отображения содержимого из этой ссылки.

```
<Window x:Class="Wrox.ProCSharp.Syndication.SyndicationReaderWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="{Binding Path=Title.Text}" Height="300" Width="450">
    <DockPanel x:Name="feedContent">
     <Grid DockPanel.Dock="Top">
      <Grid.ColumnDefinitions>
       <ColumnDefinition Width="50" />
       <ColumnDefinition Width=" * " />
       <ColumnDefinition Width="90" />
      </Grid.ColumnDefinitions>
      <Label Grid.Column="0" Margin="5">URL:</Label>
      <TextBox Grid.Column="1" x:Name="textUrl" MinWidth="150"
        Margin="5">http://blogs.thinktecture.com/cnagel/rss.aspx
      </TextBox>
      <Button Grid.Column="2" Margin="5" MinWidth="80"
        Click="OnGetFeed">Get Feed</Button>
     </Grid>
     <StackPanel Orientation="Vertical" Background="LightGreen"
      DockPanel.Dock="Top" x:Name="heading">
      <Label DockPanel.Dock="Top" Content="{Binding Path=Title.Text}" FontSize="16" />
      <Label DockPanel.Dock="Top" Content="{Binding Path=Description.Text}" />
     </StackPanel>
     <ListBox DockPanel.Dock="Left" ItemsSource="{Binding}"
      Style="{StaticResource listTitleStyle}"
      IsSynchronizedWithCurrentItem="True"
      HorizontalContentAlignment="Stretch" />
     <DockPanel x:Name="content">
      <Label DockPanel.Dock="Top" Content="{Binding Path=Description.Text}" />
      <Frame Source="{Binding Path=Links[0].Uri}" />
     </DockPanel>
    </DockPanel>
</Window>
```

Фрагмент кода ***SyndicationReader\SyndicationReader.xaml***

Пример предоставления синдицируемых каналов

Чтение синдицируемых каналов является лишь одним из случаев применения API-интерфейса Syndication. Другой случай связан с предоставлением синдицируемых каналов клиентам RSS и Atom. Для этого в Visual Studio 2010 предлагается шаблон Syndication Service Library (Библиотека службы синдикации), который послужит хорошей отправной точкой. Этот шаблон содержит ссылку на библиотеку System.ServiceModel и предусматривает возможность добавления конфигурационного файла приложения для определения конечной точки WCF.

Чтобы предоставить данные для синдицируемого канала, удобно применять ADO.NET Entity Framework. В предлагаемом здесь примере приложении используется база данных Формулы-1, входящая в состав кода для этой книги. Сначала в проект добавляется элемент ADO.NET Entity Data Model по имени `Formula1`. Таблицы `Racers`, `RaceResults`, `Races` и `Circuits` отображаются на классы сущностей `Racer`, `RaceResult`, `Race` и `Circuit`, как показано на рис. 47.3.

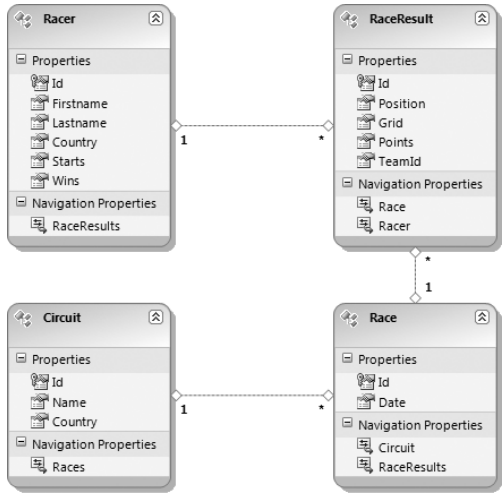

Рис. 47.3. *Отображение таблиц на классы сущностей*

Более подробные сведения об ADO.NET Entity Framework можно найти в главе 31.

Затем в шаблоне проекта создается файл `IService1.cs`, в котором размещается контракт службы WCF. В интерфейсе содержится метод `CreateFeed()`, который возвращает `SyndicationFeedFormatter`. Поскольку `SyndicationFeedFormatter` представляет собой абстрактный класс, а в реальности возвращается класс либо типа `Atom10FeedFormatter`, либо типа `Rss20FeedFormatter`, эти типы перечисляются в `ServiceKnownTypeAttribute`, чтобы тип был известен и могла выполняться сериализация.

Атрибут `WebGet` указывает, что операция может вызываться из простого HTTP-запроса `GET`, который может применяться для запроса синдицируемых каналов. `WebMessageBodyStyle.Bare` указывает, что результат (XML из синдицируемого канала) должен отправляться в таком, как он есть виде, без добавления вокруг него элемента-оболочки XML.

```
using System.ServiceModel;
using System.ServiceModel.Syndication;
using System.ServiceModel.Web;
namespace Wrox.ProCSharp.Syndication
{
  [ServiceContract]
  [ServiceKnownType(typeof(Atom10FeedFormatter))]
  [ServiceKnownType(typeof(Rss20FeedFormatter))]
  public interface IFormula1Feed
  {
    [OperationContract]
    [WebGet(UriTemplate = "*", BodyStyle = WebMessageBodyStyle.Bare)]
    SyndicationFeedFormatter CreateFeed();
  }
}
```

Фрагмент кода ***SyndicationService\IFormula1Feed.cs***

Реализация службы выполняется в классе `Formula1Feed`. Здесь создается элемент `SyndicationFeed` и производится присваивание различных свойств этого класса, таких как `Generator`, `Language`, `Title`, `Categories` и `Authors`. Свойство `Items` заполняется результатами из LINQ-запроса, который предусматривает извлечение информации о победителях гонок Формулы-1 за определенную дату. В конструкции `select` этого запроса создается новый анонимный тип, который заполняется несколькими свойствами. Эти свойства затем используется методом `Select()` для создания объектов `SyndicationItem`,

представляющих победителей. В SyndicationItem свойству Title присваивается простой текст, содержащий информацию о стране, в которой проводилась гонка. Свойство Content заполняется с помощью LINQ to XML. Классы XElement применяются для создания кода на языке XHTML, который может интерпретироваться браузером. В результате всего этого в содержимом отображается дата, когда проходила гонка, страна, в которой она проходила, и имя участника, который в ней победил.

В зависимости от того, какая строка запроса применяется для запроса синдикации, SyndicationFeed форматируется с помощью либо Atom10FeedFormatter, либо Rss20FeedFormatter.

```
using System;
using System.Linq;
using System.ServiceModel.Syndication;
using System.ServiceModel.Web;
using System.Xml.Linq;

namespace Wrox.ProCSharp.Syndication
{
  public class Formula1Feed: IFormula1Feed
  {
    public SyndicationFeedFormatter CreateFeed()
    {
      DateTime fromDate = DateTime.Today - TimeSpan.FromDays(365);
      DateTime toDate = DateTime.Today;
      string from = WebOperationContext.Current.IncomingRequest.UriTemplateMatch
       .QueryParameters["from"];
      string to = WebOperationContext.Current.IncomingRequest.UriTempateMatch
       .QueryParameters["to"];
      if (from != null && to != null)
      {
        try
        {
          fromDate = DateTime.Parse(from);
          toDate = DateTime.Parse(to);
        }
        catch (FormatException)
        {
          // Сохранение дат по умолчанию
        }
      }
    // Создание нового синдицируемого канала (SyndicationFeed).
    var feed = new SyndicationFeed();
    feed.Generator = "Pro C# 4.0 Sample Feed Generator";
    feed.Language = "en-us";
    feed.LastUpdatedTime = new DateTimeOffset(DateTime.Now);
    feed.Title = SyndicationContent.CreatePlaintextContent("Formula1 results");
    feed.Categories.Add(new SyndicationCategory("Formula1"));
    feed.Authors.Add(new SyndicationPerson("web@christiannagel.com",
              "Christian Nagel", "http://www.christiannagel.com"));
    feed.Description = SyndicationContent.CreatePlaintextContent(
              "Sample Formula 1");
    using (var data = new Formula1Entities())
    {
        var races from racer in data.Racers
                from raceResult in racer.RaceResults
                where raceResult.Race.Date > fromDate &&
                     raceResult.Race.Date < toDate &&
                     raceResult.Position == 1
                orderby raceResult.Race.Date
```

```
                select new                {
                 {
                    Country = raceResult.Race.Circuit.Country,
                    Date = raceResult.Race.Date,
                    Winner = racer.Firstname + " " + racer.Lastname
                 }).ToArray();
        feed.Items = races.Select(race =>
        {
          return new SyndicationItem
          {
            Title = SyndicationContent.CreatePlaintextContent(
                   String.Format("G.P. {0}", race.Country)),
            Content = SyndicationContent.CreateXhtmlContent(
                     new XElement("p",
                       new XElement("h3", String.Format("{0}, {1}",
                         race.Country,
                         race.Date.ToShortDateString())),
                       new XElement("b", String.Format("Winner: {0}",
                         race.Winner))).ToString())
          };
        });
// Возврат ATOM или RSS в зависимости от строки запроса
// rss -> http://localhost:8732/Design_Time_Addresses/SyndicationService/Feed1/
// atom -> http://localhost:8732/Design_Time_Addresses/SyndicationService/
//         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;
    }
  }
}
```

Фрагмент кода ***SyndicationService\Formula1Feed.cs***

При запуске этой службы в среде Visual Studio 2010 для ее обслуживания запустится WCF Service Host и в Internet Explorer появится отформатированный результат канала с URL-параметром `?from=1970/1/1&to=1971/1/1`, как показано на рис. 47.4.

В случае отправки стандартного запроса этой службе вернется канал RSS. Ниже показан фрагмент этого канала с корневым элементом `rss`. Здесь видно, что в случае RSS-канала свойство `Title` преобразуется в элемент `title`, а свойство `Description` — в элемент `description`. Для свойства `Authors` объекта `SyndicationFeed`, в котором содержится объект `SyndicationPerson`, применяется адрес электронной почты и создается элемент `managingEditor`. В случае RSS-канала содержимое элемента с описанием `item` кодируется.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:a10="http://www.w3.org/2005/Atom">
 <channel>
  <title>Formula1 results</title>
  <description>Sample Formula 1</description>
  <language>en-us</language>
  <managingEditor>web@christiannagel.com</managingEditor>
```

```
  <lastBuildDate>Sat, 25 Jul 2009 11:25:31+0200</lastBuildDate>
  <category>Formula1</category>
  <generator>Pro C# 4.0 Sample Feed Generator</generator>
  <item>
    <title>G.P. South Africa</title>
    <description>&lt;p&gt;&#xD; &lt;h3&gt;South Africa, 07.03.1970&lt;/h3&gt;
      &#xD;&lt;b&gt;Winner: Jack Brabham&lt;/b&gt;&#xD;&lt;/p&gt;
     </description>
    </item>
    <item>
      <!-- ... -->
  </channel>
</rss>
```

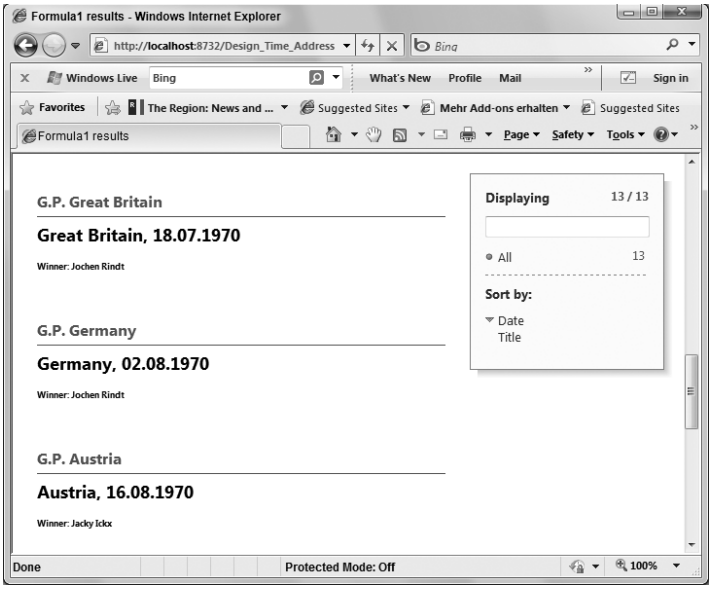

Рис. 47.4. *Результат запуска примера*

Канал в формате Atom возвращается в случае применения запроса `?format=atom`. Результат показан ниже. Здесь видно, что в таком случае корневым является элемент `feed`; свойство `Description` превращается в элемент `subtitle`, а значения свойства `Author` отображаются совершенно не так, как в случае показанного выше канала RSS. Формат Atom позволяет содержимому быть незакодированным. Здесь легко обнаруживаются элементы XHTML.

```
<feed xml:lang="en-us" xmlns="http://www.w3.org/2005/Atom">
 <title type="text">Formula1 results</title>
 <subtitle type="text">Sample Formula 1</subtitle>
 <id>uuid:c19284e7-aa40-4bc2-9be8-f1960b0f747e;id=1</id>
 <updated>2009-07-25T11:44:01+02:00</updated>
 <category term="Formula1"/>
  <author>
    <name>Christian Nagel</name>
    <uri>http://www.christiannagel.com</uri>
    <email>web@christiannagel.com</email>
  </author>
  <generator>Pro C# 4.0 Sample Feed Generator</generator>
```

```
  <entry>
    <id>uuid:c19284e7-aa40-4bc2-9be8-f1960b0f747e;id=2</id>
    <title type="text">G.P. Australia</title>
    <updated>2007-12-04T23:46:43Z</updated>
    <content type="xhtml">
      <p><h3>Australia, 18.03.2007</h3><b>Winner: Kimi Raikkonen</b></p>
    </content>
  </entry>
  <entry>
    <id>uuid:c19284e7-aa40-4bc2-9be8-f1960b0f747e;id=3</id>
    <title type="text" > G.P. South Africa</title>
    <updated>2009-07-25T09:44:05Z</updated>
    <content type="xhtml">
     <p><h3>South Africa, 07.03.1970</h3><b>Winner: Jack Brabham</b></p>
    </content>
  </entry>
  <!-- ... -->
</feed>
```

Резюме

В настоящей главе было показано, как применять классы из пространства имен `System.ServiceModel.Syndication` для создания приложений, читающих и предоставляющих синдицируемый канал. API-интерфейс Syndication поддерживает RSS 2.0 и Atom 1.0. По мере развития этих стандартов будут становиться доступными все новые и новые форматы. Как здесь было показано, классы `SyndicationXXX` не зависят от генерируемого формата. То, какие свойства должны использоваться, и в какой формат они должны преобразовываться, определено реализацией абстрактного класса `SyndicationFeedFormatter`.

Эта глава завершает часть, посвященную коммуникациям. В этой части рассматривались коммуникационные технологии, позволяющие напрямую использовать сокеты, предлагаемые уровни абстракции, а также технологии Windows Communication Foundation и Message Queuing.

В приложении будет описан полезный пакет Windows API Code Pack, который представляет собой .NET-расширение для работы с ОС Windows 7 и Windows Server 2008 R2, а также предложены рекомендации по разработке приложений, ориентированных на эти платформы.