PROGRAMMER TO PROGRAMMER™

Professional ASP.NET 3.5 SP1 In C# and VB

ASP.NET 3.5 SP1 高级编程(第6版)

涵盖 Service Pack 1

Bill Evjen
(美) Scott Hanselman 著
Devin Rader
姜奇平 译

wrox™
WILEY
清华大学出版社

ASP.NET 3.5 SP1 高级编程

(第 6 版)

Bill Evjen
(美) Scott Hanselman 著
Devin Rader

姜奇平 译

清华大学出版社

北 京

Bill Evjen，Scott Hanselman，Devin Rader
Professional ASP.NET 3.5 SP1：In C# and VB
EISBN：978-0-470-47826-4

图书在版编目(CIP)数据

ASP.NET 3.5 SP1 高级编程(第 6 版)/ (美) 伊文詹(Evjen,B.)，(美) 汉森姆(Hanselman，S.)，(美) 瑞德(Rader，D.)著；姜奇平 译 —北京：清华大学出版社，2010.1
书名原文：Professional ASP.NET 3.5 SP1：In C# and VB
ISBN 978-7-302-21548-6
Ⅰ. ①A…　Ⅱ. ①伊…　②汉…　③瑞…　④姜…　Ⅲ. 主页制作－程序设计　Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字(2009)第 219072 号

责任编辑：王　军　李　阳
装帧设计：孔祥丰
责任校对：成凤进
责任印制：李红英
出版发行：清华大学出版社　　地　　址：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　邮　　编：100084
社　总　机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　印　张：96.75　插　页：2　字　数：2596 千字
版　　次：2010 年 1 月第 1 版　　印　　次：2010 年 1 月第1 次印刷
印　　数：1～3000
定　　价：158.00 元

作 者 简 介

Bill Evjen 是.NET 技术的积极支持者，也是基于社区主动学习.NET 的支持者。自从.NET 在 2000 年推出以来，他就积极地介入其中。同年，Bill 建立了 St. Louis .NET User Group (www.stlnet.org)，这是实际意义上的第一个.NET 用户组。Bill 还是 International .NET Association (www.ineta.org) 的奠基人和创始人，现在其成员已超过 500 000 人。

Bill 住在密苏里州的路易斯街，是位热情赞赏.NET 和 XML Web 服务的作者和鼓吹者。他独立编写和与他人合著的图书超过 15 本，包括 *Professional C# 2008*、*Professional VB 2008*、*ASP.NET Professional Secrets*、*XML Web Services for ASP.NET* 和 *Web Services Enhancements: Understanding the WSE for Enterprise Applications* 等(全部由 Wiley 出版)。除了写作之外，Bill 还在各种会议上发表演讲，包括 DevConnections、VSLive 和 TechEd。另外，Bill 还与 Microsoft 关系密切，是 Microsoft 区域主管和 MVP。Bill 是 Platform Architecture for Lipper(www.lipperweb.com)的全球领袖，Platform Architecture for Lipper 是 Thomson Reuters(一家国际新闻和金融服务公司)的一个边栏。Bill 毕业于华盛顿州 Bellingham 的华盛顿大学，获得了俄语学位。他在休闲时，会到芬兰的 Toivakka 度假。Bill 的联系方式是 bill.evjen.public@gmail.com。

Scott Hanselman 是 Microsoft 开发部的一位资深项目经理，其主要工作是大力宣传 Microsoft 在软件开发方面的成就。在此之前，Scott 在 eFinance 工作了 6 年多，还担任一家 Microsoft 合作伙伴的主要顾问近 7 年。他还涉及 MVP 和 RD 程序领域，只要有人倾听，他就会谈及计算机以及其他主题。他的博客网址是 http://www.hanselman.com，播客网址(podcast)是 http://www.hanselminutes.com，Scott 还经常在 http://www.asp.net、http://www.windowsclient.net 和 http://www.silverlight.net 上发表文章。

Devin Rader 是 Infragistics Web Client 团队的一位产品经理，负责领导 Infragistics ASP.NET 和 Silverlight 产品的开发。Devin 是.NET 技术的积极支持者，.NET 开发团队的一员，St. Louis .NET User Group 的创始人之一，New Jersey .NET User Group 的一位很积极的成员，International .NET Association 的前任成员。他经常在用户组上发表文章，还与他人合著了 Wrox 图书 *Silverlight 1.0*，他是其他几本 Wrox 图书的技术编辑，经常为 ASP.NET Pro 杂志撰写文章，在 MSDN Online 发表过多篇有关.NET 技术方面的文章。Devin 的联系方式是 www.geekswithblogs.com/devin。

致　谢

我以前就说过，但现在还要说：编写一本书不仅需要尽一个人的最大努力，而且它还需要一个团队的紧密合作，才能使技术类图书成功面世——本书也不例外。首先我要感谢 Wrox 出版社的 Jim Minatel 给我这样一个机会编写 ASP.NET 图书的第 1 版，因此才有了现在的这版书。能为全世界最杰出的出版社就自己喜欢的主题编写图书，没有比这更幸福、更自豪的事情了。

除了 Jim 之外，我还与 Adaobi Obi Tulton 合作编写了本书的第 1 版，在这个 SP1 版本中，Lori Cerreto 是开发编辑，Nancy Rapoport 是技术编辑，没有他们的辛勤劳动，本书不可能面世。

我与 Scott Hanselman、Devin Rader 合作编写了本书的最初版本，非常感谢他们二位的帮助和提出的建议(本书的新附录是由 Devin 执笔的)。

最后，感谢我的家人。图书编写工作是很痛苦的，虽然我喜欢这个工作，但它占用了我与家人在一起的许多时间。感谢家人对我的容忍，支持并帮助我把本书编写出来。

——Bill Evjen

前　言

ASP.NET 3.5 是一种建立 Web 解决方案的令人惊异的技术。早在 ASP.NET 1.0 版本于 2000 年发布时，许多人就认为它在 Web 应用程序开发方面迈出了具有革命性的一步。而后来的 ASP.NET 2.0 更激动人心、更富有革命性，ASP.NET 3.5 Service Pack 1(SP1)则继续朝着这个方向前进，为在 Web 上建立应用程序提供了目前最佳的框架。ASP.NET 3.5 SP1 建立在已发布的 ASP.NET 1.0 基础之上，但它主要关注的是开发人员的效率。

本书介绍 ASP.NET 的所有内容，除了论述新主题外，还列举了一些有关这些新技术的例子。

0.1　简史

在各个公司考虑为 Internet 开发应用程序之前，应用程序的开发主要集中在桌面应用程序上。这些胖客户端应用程序适用于所有场合：家用计算、游戏、办公等。这种应用程序模型的流行可谓势不可挡。

在这个过程中，Microsoft 开发胖客户端应用程序使用的是其主要产品 Visual Basic(VB)。

Visual Basic 不仅是一种编程语言，它还与便于开发胖客户应用程序的 IDE 有密切的关系。在 Visual Basic 模型中，开发人员可以把控件拖放到窗体上，设置这些控件的属性，给它们提供代码来处理控件的事件。例如，终端用户单击 Visual Basic 窗体上的一个按钮时，窗体的隐藏代码就会处理该事件。

在 20 世纪 90 年代中期，Internet 开始崭露头角。Microsoft 未能将 Visual Basic 模型转向基于 Internet 应用程序的开发。Internet 的确有强大的功能，此时胖客户端应用程序模型面临的问题也开始显露出来。基于 Internet 的应用程序创建了每个人都能访问的一个应用程序实例。拥有应用程序的一个实例，意味着在对应用程序进行升级或打补丁时，对这个实例的修改会立即展现给通过浏览器访问该应用程序的每个用户。

为了进入 Web 应用程序行业，Microsoft 开发了 Active Server Pages (ASP)。ASP 是开发 Web 页面的一种快捷方式。ASP 页面由一个页面组成，其中包含了标记和语言的混合。ASP 的强大之处在于，在将页面发送给终端用户的 Web 浏览器之前，可以在页面上包含在 Web 服务器上执行的 VBScript 或 JScript 代码指令。这是创建动态 Web 页面的一种简单方式，动态 Web 页面是根据开发人员指定的参数进行定制的。

ASP 在尖括号和百分号<% %>之间使用脚本来控制服务器端的行为。开发人员可以先从一组静态的 HTML 开始建立 ASP 页面。页面需要的动态元素用脚本语言(如 VBScript 或 JScript)来定义。当用户使用浏览器从服务器上请求页面时，asp.dll (这是一个 ISAPI 应用程序，它在脚本语言和 Web 服务器之间架起了一座桥梁)就提取页面，根据脚本中指定的编程逻辑定义页面中的动态部分。定义页面中的所有动态部分后，所得到的结果就是一个 HTML 页面，该页面被输出到请求客户机的浏览器上。

在开发 Web 应用程序模型的过程中，静态 HTML 中混合了越来越多的语言，以帮助处理输出页面的操作方式和外观。随着时间的推移，ASP 页面上将出现非常多的语言、脚本和纯文本，开发人员开始把使用这些特性的页面称为 spaghetti code(意大利细面条式代码)。例如，页面上可能使用了 HTML、VBScript、JavaScript、层叠样式表、T-SQL 等。在这种情况下，页面是很难管理的。

ASP 进一步演化并推出了新版本。ASP 2.0 和 3.0 开始流行，因为这些技术非常便于 Web 页面的创建。它们的日益流行是因为它们出现在 20 世纪 90 年代后期，那时诞生了.com。在这个阶段，人们开发了许多 Web 页面和门户，ASP 是一种业界领先的技术，个人和公司都使用该技术建立 Web 页面。甚至到现在，仍可以在 Internet 上找到许多.asp 页面——包括 Microsoft 的一些 Web 页面。

但在 1998 年后期，Active Server Pages 发布其最后一个版本时，Microsoft 雇员 Marc Anders 和 Scott Guthrie 有了另一个想法。他们称之为 XSP(这个缩写词没有什么特别的含义)——这是以面向对象的方式创建 Web 应用程序的方法，而不是使用 ASP 3.0 的过程式方法来创建。他们把这个想法告诉 Microsoft 中许多不同的团体，并得到了广泛的认可。2000 年夏， Microsoft 的 Professional Developers Conference 发布了其测试版 ASP+。与会者都非常渴望使用它。该技术发布时(与.NET Framework 1.0 的最终版本一起发布)，被重新命名为 ASP.NET——加上.NET 标记是因为在那时 Microsoft 的大多数新产品都加上了这个标记。

在引入.NET 之前，传统 ASP 提供的模型和 Visual Basic 中开发的模型大相径庭，很少有 VB 开发人员能开发 Web 应用程序，Web 开发人员也不能开发 VB 的胖客户应用程序。这是一条很大的鸿沟，ASP.NET 则为此搭建了一座桥梁。ASP.NET 把 Visual Basic 样式的事件模型引入到 Web 应用程序的开发中，为无状态的 HTTP 提供了迫切需要的状态管理技术。其模型非常类似于早期的 Visual Basic 模型，因为开发人员可以把控件拖放到设计界面或窗体上，处理控件的属性，处理控件的代码，甚至基于控件的生存周期指定一些事件。ASP.NET 综合了这两个模型的优点，如本书后面所述。

读者一定很希望使用 ASP.NET 3.5 SP1 这个最新版本，看看这种新技术能给自己带来什么。下面就讨论 ASP.NET 的目标，看看它有什么新内容。

0.2 ASP.NET 的目标

ASP.NET 3.5 是该产品的另一个重要版本，建立在.NET Framework 2.0 的核心功能之上，并带有额外的类和功能。Framework 的这个版本在 Microsoft 内部的代码名称是 Orcas，读者可能听说过其他人把 ASP.NET 的这个版本称为 ASP.NET Orcas。ASP.NET 3.5 继续使 ASP.NET 开发人员成为 Web 领域中最高效的开发人员。本书也重点介绍 ASP.NET 3.5 SP1 版本中的 ASP.NET 3.5 和.NET Framework 3.5 的新功能。

Microsoft 小组刚开始开发 ASP.NET 2.0 时，就设定了要实现的目标。这些目标集中于开发人员的效率、管理、性能和可伸缩性。

0.2.1 开发人员的效率

ASP.NET 3.5 的主要目标是效率。ASP.NET 1.x 的发布就已经达到了很高的效率，但效率还能进一步提高吗？

ASP.NET 开发小组的一个目标是去除 ASP.NET 中原来必需的大量繁琐的编码，使常见的 ASP.NET 任务更容易完成。开发人员的高效率将在本书中体现出来。在介绍这些功能之前，首先看看以前的 ASP.NET 1.0 技术，以便与 ASP.NET 3.5 进行比较。程序清单 0-1 使用 ASP.NET 1.0 在 Web 页面上建立了一个表，并可以对所提供的数据进行简单的分页。

程序清单 0-1 在支持分页功能的 DataGrid 服务器控件上显示数据(仅用于 VB)

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<script runat="server">

    Private Sub Page_Load(ByVal sender As System.Object, _
      ByVal e As System.EventArgs)
        If Not Page.IsPostBack Then
            BindData()
        End If
    End Sub

    Private Sub BindData()
        Dim conn As SqlConnection = New SqlConnection("server='localhost';
          trusted_connection=true; Database='Northwind'")
        Dim cmd As SqlCommand = New SqlCommand("Select * From Customers", conn)
        conn.Open()

        Dim da As SqlDataAdapter = New SqlDataAdapter(cmd)
        Dim ds As New DataSet

        da.Fill(ds, "Customers")

        DataGrid1.DataSource = ds
        DataGrid1.DataBind()
    End Sub

    Private Sub DataGrid1_PageIndexChanged(ByVal source As Object, _
      ByVal e As System.Web.UI.WebControls.DataGridPageChangedEventArgs)
        DataGrid1.CurrentPageIndex = e.NewPageIndex
        BindData()
    End Sub

</script>
<html>
<head>
</head>
<body>
    <form runat="server">
        <asp:DataGrid id="DataGrid1" runat="server" AllowPaging="True"
```

```
        OnPageIndexChanged="DataGrid1_PageIndexChanged"></asp:DataGrid>
    </form>
</body>
</html>
```

尽管这里使用了相当多的代码，但与使用传统的 Active Server Pages 3.0 相比，所需的代码量削减了很多。这里不详细探讨这些旧代码的细节，只说明为了给表中显示的数据添加额外的常见功能(如分页功能)，开发人员必须创建定制的代码。

在 ASP.NET 3.5 中，这是开发人员效率提高最显著的一个方面。ASP.NET 3.5 提供了一个新的控件，称为 GridView 服务器控件，这个控件非常类似于以前的 DataGrid 服务器控件，但 GridView 服务器控件(除了提供许多其他新功能之外)内置了分页、排序和编辑数据的功能，而用户不需要做任何工作。程序清单 0-2 就是使用 GridView 服务器控件的一个例子，该例子为 Northwind 数据库的 Customers 表中的数据建立了一个包含分页功能的表。

程序清单 0-2　用新的 GridView 服务器控件查看分页的数据集

```
<%@ Page Language="VB" %>

<script runat="server">

</script>

<html xmlns=http://www.w3.org/1999/xhtml>
<head runat="server">
    <title>GridView Demo</title>
</head>
<body>
    <form runat="server">
        <asp:GridView ID="GridView1" Runat="server". AllowPaging="True"
         DataSourceId="Sqldatasource1" />
        <asp:SqlDataSource ID="SqlDataSource1" Runat="server"
         SelectCommand="Select * From Customers"
         ProviderName="System.Data.OleDb"
         ConnectionString="Provider=SQLOLEDB;Server=localhost;uid=sa;
         pwd=password;database=Northwind" />
    </form>
</body>
</html>
```

使用几个新服务器控件就可以应用分页功能。要启用这个功能，只需使用 GridView 服务器控件的 AllowPaging 属性即可：

```
<asp:GridView ID="GridView1" Runat="server" AllowPaging="True"
 DataSourceId="SqlDataSource1" />
```

在文档的代码部分，所发生的另一个有趣事件如下：

```
<script runat="server">

</script>
```

实际上，运行该文件并不需要这两行代码。这里包含它们是为了说明一点：不需要编写任何服务器端代码，就可以使页面工作！只需包含一些服务器控件：获取数据的控件和显示数据的控件。然后，将这些控件关联到一起。

0.2.2　性能和可伸缩性

Microsoft 小组为 ASP.NET 设定的一个目标是提供世界上最快的 Web 应用程序服务器。本书也介绍 ASP.NET 3.5 中许多性能上的改进。

最激动人心的性能改进之一是为了利用 Microsoft 的 SQL Server 而新增的高速缓存功能。ASP.NET 3.5 包含一个名为“禁用 SQL 高速缓存”的特性。在 ASP.NET 2.0 之前，可以高速缓存来自 SQL Server 的结果，并根据一定的时间间隔来更新高速缓存，例如，每 15 秒更新一次。这样，如果结果集在 15 秒的时间内发生了变化，终端用户看到的可能就是过期的数据。

在一些情况下，更新结果集的这个时间间隔是无法接受的。在理想情况下，如果从中提取结果集的数据源(这里是 SQL Server)发生了变化，那么存储在高速缓存中的结果集就应删除。在 ASP.NET 3.5 中，禁用 SQL 高速缓存功能就可以达到这个目的。也就是说，来自 SQL Server 的结果集发生变化时，输出的高速缓存也会变化，终端用户看到的总是最新的结果集，显示的数据永远都没有过期。

ASP.NET 3.5 提供了 64 位支持，于是，现在可以在 64 位的 Intel 或 AMD 处理器上运行 ASP.NET 应用程序。

ASP.NET 3.5 完全兼容 ASP.NET 1.0、1.1 和 2.0，所以可以在.NET Framework 3.5 上打开以前的 ASP.NET 应用程序，重新编译，并在 64 位处理器上运行它们。

0.3　ASP.NET 3.5 和 3.5 SP1 的其他新特性

前面学习了 ASP.NET 小组为 ASP.NET 设定的主要目标。为了达到这些目标，ASP.NET 小组在 ASP.NET 中内置了许多新特性。下面就介绍其中的几个。

0.3.1　新的开发基础结构

ASP.NET 3.5 中的一个重要改进是新的基础结构，可以在应用程序中使用它。ASP.NET 小组选择 Web 应用程序中一些最常见的编程操作，直接内置到 ASP.NET 中，这样可以省去大量的时间和编码量。

1. 成员和角色管理

在 ASP.NET 2.0 以前的版本中，如果开发一个要求用户登录应用程序，以获得许可的访问权限的门户，那么总是需要自己创建它。这样创建出来的应用程序在某些方面就存在一些限制，只有选定的人才可以访问它。

在 ASP.NET 3.5 中内置了这个功能。现在可以使用程序清单 0-3 来验证用户。

程序清单 0-3　通过代码验证用户

VB

```
If (Membership.ValidateUser (Username.Text, Password.Text)) Then
   ' Allow access code here
End If
```

C#

```
if (Membership.ValidateUser (Username.Text, Password.Text)) {
   // Allow access code here
}
```

使用 ASP.NET 3.5 中的一系列 API、控件和提供程序可以控制应用程序的用户成员和角色管理。使用这些 API，可以轻松地管理用户及其复杂的角色——创建、删除和编辑它们。这些功能只需使用 API 或内置的 Web 工具 Web Site Administration Tool 即可实现。

对于用户及其角色的存储，ASP.NET 3.5 使用.mdf 文件(该文件类型用于 SQL Server Express Edition)来存储所有的用户和角色。但这个数据存储是没有限制的，可以扩展 ASP.NET 提供的功能，使用任何数据存储建立自己的提供程序。例如，可以轻松地在 LDAP 或 Oracle 数据库中建立自己的用户存储。

2. 个性化

门户(portal)为其成员提供的一个高级特性是个性化产品，终端用户可以定制站点的外观和功能。个性化应用程序和存储个性化设置的功能现在已完全内置到 ASP.NET 框架中。

个性化常常围绕一个用户和这个用户承担的角色来进行，所以个性化体系结构可以与成员和角色基础结构紧密联系起来。存储所创建的个性化设置有几种方式。在 Microsoft Access 和 SQL Server 中存储这些设置的功能已内置到 ASP.NET 3.5 中。与成员和角色 API 的功能一样，可以使用灵活的提供程序模型，通过改变内置提供程序使用可用数据存储的方式，或者建立自己的定制数据提供程序，来实现全新的数据存储。个性化 API 也支持一系列数据存储，因此可以使用多个数据存储。

使用这些新 API 很容易地为定制过程创建一个站点，所以这个特性对于用户建立的任意应用程序来说都是有价值的。

3. ASP.NET 门户框架

在使用 ASP.NET 1.0 时，开发人员可以访问 ASP.NET 小组的站点(http://www.asp.net)，下载一些 Web 演示应用程序，如 IBuySpy。这些演示程序叫作 Developer Solution Kits，可以用作目前 Internet 上许多 Web 站点的基础。其中一些甚至扩展为开放源代码的框架，如 DotNetNuke。

IBuySpy 的优点是，可以把它提供的代码作为基础，建立 Web 存储或门户。只需以基本代码作为起点，扩展它即可。例如，可以改变代码显示部分的外观和操作方式，或者在其模块化体系结构中引入高级功能。Developer Solution Kits 相当流行，因为它们使这些类型的操作非常容易实现。

像 IBuySpy 这样的框架非常流行，因此，ASP.NET 3.5 提供了内置功能，以使用 Web Parts 方便地建立门户。可以使用 Portal Framework 建立的项目类型非常多。使用 Web Parts 建立项目

的优点在于终端用户可以根据自己的喜好完全定制门户。

4. 站点导航

ASP.NET 小组成员认识到，终端用户希望方便地在整个应用程序中导航。以逻辑方式进行导航的机制有时很难编码。该小组在 ASP.NET 中使用一系列基于导航的服务器控件解决了这个问题。

首先，在一个 XML 文件中为应用程序建立一个站点地图，特定的控件可以在该站点地图中工作。程序清单 0-4 显示了一个示例站点地图文件。

程序清单 0-4　站点地图文件示例

```
<?xml version="1.0" encoding="utf-8" ?>

<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode title="Home" description="Home Page" url="default.aspx">
    <siteMapNode title="News" description="The Latest News" url="News.aspx">
      <siteMapNode title="U.S." description="U.S. News"
       url="News.aspx?cat=us" />
      <siteMapNode title="World" description="World News"
       url="News.aspx?cat=world" />
      <siteMapNode title="Technology" description="Technology News"
       url="News.aspx?cat=tech" />
      <siteMapNode title="Sports" description="Sports News"
       url="News.aspx?cat=sport" />
    </siteMapNode>
    <siteMapNode title="Finance" description="The Latest Financial Information"
     url="Finance.aspx">
      <siteMapNode title="Quotes" description="Get the Latest Quotes"
       url="Quotes.aspx" />
      <siteMapNode title="Markets" description="The Latest Market Information"
       url="Markets.aspx">
        <siteMapNode title="U.S. Market Report"
         description="Looking at the U.S. Market" url="MarketsUS.aspx" />
        <siteMapNode title="NYSE"
         description="The New York Stock Exchange" url="NYSE.aspx" />
      </siteMapNode>
      <siteMapNode title="Funds" description="Mutual Funds"
       url="Funds.aspx" />
    </siteMapNode>
    <siteMapNode title="Weather" description="The Latest Weather"
     url="Weather.aspx" />
  </siteMapNode>
</siteMap>
```

有了站点地图之后，就可以把这个文件用作几个站点导航服务器控件的数据源，如 TreeView 和 SiteMapPath 服务器控件。TreeView 服务器控件允许在应用程序中放置可扩展的站点导航系统。图 0-1 显示了 TreeView 服务器控件的许多外观之一。

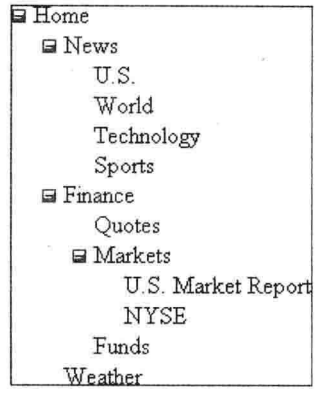

图 0-1

SiteMapPath 控件可以在应用程序中放置一些导航项，这样终端用户就可以查看应用程序当前采用的路径，并能轻松地导航到树型结构的上一级。图 0-2 显示了 SiteMapPath 服务器控件的一个例子。

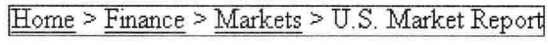

图 0-2

这些站点导航功能为编程访问站点布局提供了极大的便利，甚至也考虑到了类似于终端用户的角色如何确定显示站点的哪一部分这类事情。

5. ADO.NET 实体框架

大多数开发人员都需要使用某种底层的数据库，无论使用的是 Microsoft SQL Server 数据库还是 Oracle 数据库，应用程序通常都要提取某类内容，以执行操作。使用底层数据库的难点在于数据库与面向对象的代码在处理对象时使用的方式大相径庭。

在数据库领域中，数据结构放在表中，项中的集合(例如，附带有 Orders 的 Customers 对象)简单地表示为用 Join 语句连接起来的两个表。而在面向对象的代码中，在表示这些对象时，Orders 项是 Customers 对象的一个属性。把这两个领域关联起来，并映射出这些区别总是比较麻烦的。

ASP.NET 3.5 SP1 引入了 ADO.NET 实体框架，它有点类似于 LINQ to SQL。ADO.NET 实体框架的目标是允许创建一个实体数据模型(Entity Data Model，EDM)，以便于映射面向对象的数据与这些数据在数据库中的表达方式。

ADO.NET 实体框架的一个优点是它可以使用许多不同类型的数据库，所以不像 LINQ to SQL 那样只使用一个数据库。另一个优点是 ADO.NET 实体框架是 ASP.NET 3.5 SP1 引入的其他一些新技术的基础，如 ADO.NET 数据服务。

6. ASP.NET 动态数据

ASP.NET 3.5 SP1 的另一个新特性是 ASP.NET 动态数据，利用这个功能可以方便地在几分钟内，通过数据库创建报表和数据输入应用程序。

使用 ASP.NET 动态数据是很简单的，只需指向在应用程序中创建好的实体数据模型(EDM)，允许动态数据引擎自动创建 Web 页面，以提供数据库中的创建、编辑、更新和删除等全部功能。

ASP.NET 动态数据要求有一个实体数据模型才能工作。但不一定非要使用 ADO.NET 实体框架，也可以使用已创建好的任意 LINQ to SQL 模型。

ASP.NET 动态数据的体系结构的一个优点是，它在动态生成站点上的页面时使用的是模板。像使用这个系统的开发人员一样，用户也可以使用系统“as－is”，甚至可以提取其中的一部分，把它的功能合并到已有的 ASP.NET 应用程序中。

7. ADO.NET 数据服务

ASP.NET 3.5 SP1 版本引入的另一个新特性是 ADO.NET 数据服务。ADO.NET 数据服务的正式名称为 Project Astoria，它允许用户在数据库上创建 Restful 服务接口。

使用 ADO.NET 数据服务，可以把请求的 URL 用作命令驱动的 URI，并和 HTTP 动词一起使用，告诉服务器如何处理底层数据。使用这种技术可以创建、读取、更新或删除底层数据库的数据，但接口的实现人员也可以限制终端用户可用的功能和访问权限。

0.3.2 ASP.NET 编译系统

ASP.NET 1.0 中的编译是一个很困难的过程。在 ASP.NET 1.0 中，要使用 ASP.NET 和 Visual Studio 建立应用程序的隐藏代码文件，部署它，然后在请求每个页面时，逐个观察.aspx 文件是否已编译。如果在 ASP.NET 1.0 中对隐藏代码文件进行了修改，这些修改的内容将不能反映到应用程序中，除非重新建立了整个应用程序。也就是说，在重新编译整个应用程序之前，必须再次逐个地请求每个页面。

ASP.NET 1.0 处理类和编译的方式在 ASP.NET 3.5 中都得到了改进。新编译系统的机制首先从页面在 ASP.NET 3.5 中的构建方式入手。在 ASP.NET 1.0 中，可以使用隐藏代码模型构建页面，也可以把所有的服务器代码放在.aspx 页面的<script>标记之间。大多数页面都使用隐藏代码模型来构建，因为在 Visual Studio .NET 2002 或 2003 中，这是默认方式。在这些 IDE 中使用内嵌编码方式创建页面是比较困难的。如果用这种方式创建页面，就不能使用 IntelliSense，而在使用.NET Framework 提供的绝大多数类集合时，使用 IntelliSense 可以使创建工作更容易完成。

ASP.NET 3.5 提供了一个与 1.0/1.1 不同的隐藏代码模型，因为.NET Framework 3.5 提供了使用部分类(也称为部分类型)的功能。在编译时，各个文件将组合为一个产品，这将生成非常简洁的隐藏代码页面。类中 Web Form Designer Generated 部分的代码与自己创建的隐藏代码类是分开的。而在 ASP.NET 1.0 中与此相反，.aspx 页面需要从它自己的隐藏代码文件中派生，以表示一个逻辑页面。

ASP.NET 3.5 应用程序可以包含一个\App_Code 目录，以放置类的源代码。放在该目录下的所有类都是动态编译的，并在应用程序中反映出来。在进行修改时，不需要像 ASP.NET 1.0 那样使用独立的建立过程，这是一个“只要保存了就可以使用”的部署模型，类似于传统 ASP.NET 3.0 中的部署模型。Visual Studio 2008 也为\App_Code 目录中的对象提供了 IntelliSense，因此可以使用隐藏代码模型，也可以在线编码。

ASP.NET 3.5 还提供了一些工具，可以预先编译 ASP.NET 应用程序，包括.aspx 页面和隐藏代码，这样当第一次检索页面时，应用程序中的所有页面都不会出现延迟现象。如果页面中有错误，即使不调用每个页面，也可以找出这些错误。ASP.NET 2.0(3.0 和 3.5)应用程序的预编译非常简单，只需使用 aspnet_compiler.exe，利用某些可用的标记即可。在预编译整个应用程序时，如果在应用程序中发现错误，我们就会收到错误通知。还可以预先编译应用程序，再给部署服务器传送已创建好的程序集，从而防止代码在部署后被窃取、修改和损坏。本书后面会列举这方面的示例。

0.3.3 ASP.NET 应用程序的健康监控

ASP.NET 内置的健康监控功能是相当强大的，它非常便于管理已部署的 ASP.NET 应用程序。顾名思义，健康监控功能可以监控已部署的 ASP.NET 应用程序的健康状况和性能。

使用健康监控系统可以执行健康监控事件(称为 Web 事件)的记录操作，如失败的登录、应用程序的启动和停止，或未处理的异常等。事件记录操作可以在多个地方执行，因此可以把事件记录到事件日志或数据库中。除了执行这个基于磁盘的记录操作之外，还可以使用该系统通过电子邮件发送健康监控信息。

除了处理应用程序中的特定事件之外，健康监控系统还可以记录正在运行的应用程序在某一刻的健康状况。与 ASP.NET 3.5 内置的大多数系统一样，也可以扩展健康监控系统，创建自己的事件，以记录应用程序信息。

在系统的.config 文件中，默认为激活健康监控功能。健康监控的默认设置会把所有的错误和失败的审查记录到事件日志中。例如，每在应用程序中抛出一个错误，就会在 Application 日志中记录一个错误通知。

只需对应用程序的 web.config 文件进行一些小的修改，就可以改变默认的事件记录操作。例如，假设要把这些错误事件信息存储在应用程序的一个 SQL Express 文件中，就可以在 web.config 文件中添加一个<healthMonitoring>节点，如程序清单 0-5 所示。

程序清单 0-5　在 web.config 文件中定义健康监控功能

```
<healthMonitoring enabled="true">
 <providers>
     <clear />
     <add name="SqlWebEventProvider"
      connectionStringName ="LocalSqlServer"
      maxEventDetailsLength="1073741823" buffer="false"
      bufferMode="Notification"
      type="System.Web.Management.SqlWebEventProvider,
        System.Web, Version=2.0.0.0, Culture = neutral,
        PublicKeyToken=b03f5f7f11d50a3a"
</providers>
<rules>
   <clear>
   <add name ="All Errors Default" eventName="All Errors"
   Providers ="SqlWebEventProvider"
```

```
        profile="Default" minInstances="1" maxLimit="Infinite"
        minInterval="00:01:00" custom="" />
        <add name ="Failure Audits Default" eventName="Failure Audits"
        Providers ="SqlWebEventProvider" profile="Default"
        minInstances="1" maxLimit="Infinite"
        minInterval="00:01:00" custom="" />
    </rules>
    </healthMonitoring>
```

进行了这个修改后，事件就被记录到项目中自动创建的 ASPNETDB.MDF 文件中。

打开这个 SQL Express 文件，可以找到 aspnet_WebEvent_Events 表，其中包含所有这些信息。

第 35 章将介绍 ASP.NET 3.5 提供的许多健康监控功能。

0.3.4　读写配置设置

使用 WebConfigurationManager 类，可以读写服务器或应用程序配置文件。也就是说，可以读写应用程序所用的 machine.config 或 web.config 文件中的设置。

读写配置文件的功能不但可以处理应用程序所在的本地机器，还可以在远程服务器或应用程序中执行这些操作。

当然，在配置文件上执行这些读取或修改操作时，还可以采用 GUI 方式。内置的 GUI 工具使用 WebConfigurationManager 类提供了这个功能(如 Windows XP 中的 ASP.NET MMC 插件或 Windows Vista 中的新 IIS 界面)，这个类还可以用于定制管理工具。

程序清单 0-6 演示了如何从应用程序的 web.config 文件中读取连接字符串。

程序清单 0-6　从应用程序的 web.config 文件中读取连接字符串

VB

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
  Try
    Dim connectionString As String = _
        ConfigurationManager.ConnectionString("Northwind").
              ConnectionString.ToString()
      Label1.Text = connectionString
    Catch ex As Exception
      Label1.Text = "No connection string found."
    End Try
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
  try
  {
    string connectionString =
        ConfigurationManager.ConnectionString["Northwind"].
            ConnectionString.ToString();
      Label1.Text = connectionString;
    }
```

```
        catch(exception) {
          Label1.Text = "No connection string found. ";
        }
    }
```

这段代码使用一个 Label 控件把 web.config 文件中的 Northwind 连接字符串写到屏幕上。可以看出，从配置文件中提取信息是非常简单的。

0.3.5 本地化

ASP.NET 比以前更易于使应用程序本地化。除了使用 Visual Studio 之外，还可以通过创建资源文件(.resx)根据请求者的文化设置动态地修改已创建的页面。

ASP.NET 3.5 还通过两个应用程序文件夹 App_GlobalResources 和 App_LocalResources 提供可在整个应用程序的范围内使用的资源，或提供在应用程序的某些页面上使用的资源。

在.resx 文件中定义的项可以直接在 ASP.NET 服务器控件中访问，或使用如下表达式来编程访问：

```
<%= Resources.Resource.Question %>
```

这个系统很直观，也很容易实现。有关这个主题的内容详见第 33 章。

0.3.6 页面框架的新增内容

可以根据可视化继承来构建 ASP.NET 页面。这在 Windows Forms 中已经实现，但使用 ASP.NET 很难实现。使用主题很容易给应用程序中的各个页面应用统一的外观和操作方式。过去使用 ADO.NET 时碰到的许多困难现在都通过一系列新的数据源控件克服了，这些数据源控件可以从一个大的数据存储集合中访问和检索数据。

1. Master 页面

ASP.NET 引入了 Master 页面，现在可以在 ASP.NET 应用程序中使用可视化继承了。因为许多 ASP.NET 应用程序中的页面都有类似的结构，所以创建一个页面模板，把这个模板应用于整个应用程序是很合理的。

在 ASP.NET 中，通过创建一个带后缀.master 的页面可以实现这一点，如图 0-3 所示。

Master 页面包含标题、脚标以及所有页面都有的其他元素。每个继承和使用这个模板的页面都要包含这些元素。除了这些核心元素之外，还可以在 Master 页面中包含用于子页面(或内容页面)的服务器控件<asp:ContentPlaceHolder>，以改变 Master 页面模板的特定区域。编辑子页面时的情形如图 0-4 所示。

终端用户调用某个子页面时，就是在查看用该子页面及其继承的 Master 页面编译的单个页面。这也意味着页面的服务器和客户端代码在新的页面上处于激活状态。

Master 页面的优点是，可以在一个地方进行影响整个站点的修改，而不需要修改应用程序中的每个页面。