

.NET 开发经典名著

# ASP.NET 4.5 入门经典

## (第 7 版)

[美] Imar Spaanjaars 著

刘楠 陈晓宇 译

清华大学出版社

北 京



Imar Spaanjaars  
Beginning ASP.NET 4.5 in C# and VB  
EISBN: 978-1-118-31180-6  
Copyright © 2013 by John Wiley & Sons, Inc., Indianapolis, Indiana  
All Rights Reserved. This translation published under license.

本书中文简体字版由 Wiley Publishing, Inc. 授权清华大学出版社出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

北京市版权局著作权合同登记号 图字: 01-2013-4535

Copies of this book sold without a Wiley sticker on the cover are unauthorized and illegal.

本书封面贴有 Wiley 公司防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

#### 图书在版编目(CIP)数据

ASP.NET 4.5 入门经典(第 7 版) / (美)史潘加斯(Spaanjaars, I.)著; 刘楠, 陈晓宇 译. —北京: 清华大学出版社, 2013.9

(.NET 开发经典名著)

书名原文: Beginning ASP.NET 4.5 in C# and VB

ISBN 978-7-302-33414-9

I. A… II. ①史… ②刘… ③陈… III. ①网页制作工具—程序设计 IV. ①TP393.092

中国版本图书馆 CIP 数据核字(2013)第 180808 号

责任编辑: 王 军 刘伟琴

装帧设计: 孔祥峰

责任校对: 成凤进

责任印制:

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 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](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈: 010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 刷 者:

装 订 者:

经 销: 全国新华书店

开 本: 185mm×260mm 印 张: 44.5 字 数: 1194 千字

版 次: 2013 年 9 月第 1 版 印 次: 2013 年 9 月第 1 次印刷

印 数: 1~4000

定 价: 98.00 元

---

产品编号:

# 作者简介

Imar Spaanjaars 毕业于荷兰 Leisure Management School, 主修休闲管理专业, 但不久就转入互联网领域。

在超过 12 年的职业生涯中, Imar 曾就职于互联网领域的多家网络公司。最近, 他创办了自己的公司 De Vier Koeden(<http://devierkoeden.com>), 为客户提供使用 Microsoft 的 ASP.NET 4.5 等技术开发互联网和内部网应用程序方面的咨询和开发等服务。

Imar 编写了多本关于 ASP.NET 和 Macromedia Dreamweaver 的书籍, 包括《ASP.NET 4 入门经典——涵盖 C#和 VB.NET(第 6 版)》(由清华大学出版社引进并出版)。他是 Wrox 社区论坛([p2p.wrox.com](http://p2p.wrox.com))的主要贡献者之一, 在那里他与其他程序员分享自己的知识。

自 2008 年以来, 鉴于 Imar 对 ASP.NET 社区做出的突出贡献, Microsoft 公司每年都授予他“最有价值专家(MVP)”称号。2012 年早期, Imar 加入了 ASPInsiders, 这是专业人士为 ASP.NET 未来版本的新功能提供反馈和指导的国际小组。

Imar 和他的女朋友 Fleur 及儿子 Niek 居住在荷兰乌德勒支。可以通过他的个人网站 <http://imar.spaanjaars.com> 或者电子邮件 [imar@spaanjaars.com](mailto:imar@spaanjaars.com) 联系他。

# 致 谢

尽管版本只是从 ASP.NET 4 变化为 ASP.NET 4.5，这似乎表明 ASP.NET 4.5 或 Visual Studio 2012 没有太多的新内容，但读者会惊讶于这些产品中有太多有变化的内容。我花费了好几个月的时间把这本书从 .NET 4 版本升级到新的 .NET 4.5 版本。我每天都能发现新特性和新功能。其中一些变化很小，但积累起来，就对提高效率的帮助很大，另外一些变化就比较大，会影响构建或部署 Web 站点的方式。只要 ASP.NET 和 Visual Studio 中的新功能对读者有益，我就会把它们吸收进来，尽管有些读者的 ASP.NET 经验很少或几乎没有。

本书还根据读者的反馈进行了许多修改，与本书以前的版本一样，我仔细查看了已提交的所有勘误表和数百个论坛帖子，找出了读者觉得有困难的部分，并提出了改进它们的方法。如果您拥有本书的以前版本，并在 Wrox 论坛上提出了问题，那么我要感谢您宝贵的反馈信息：您为本书的完善贡献了自己的力量。

除了读者之外，我还要感谢帮助我编写本书的其他人。

首先，非常感谢 Brian Herrmann 和 Kim Cofer 的编辑工作，与他们一起工作真是很高兴！我还要感谢 Damien Foggon，作为技术编辑，他提出了许多有益的建议，并帮助我为本书定型。另外要感谢 Wrox 的许多人对本书的支持和贡献。

另一个要特别感谢的是我的朋友 Anne Ward，她来自 Blue Violet，这是一家总部在英国的网络和图形设计公司。Anne 完成了本书中的大多数新设计，我非常感谢她的参与。本书中的音乐会图片由 Nigel D. Nudds 提供，他允许我使用他相册中的照片。

最后我要感谢我深爱的女朋友 Fleur 在完成这个项目的过程中对我的支持。有了她的帮助，编写本书时，虽然我们的儿子 Niek 刚刚出生，我也没有觉得太困难。

# 序

每天，像 HTML5 和 CSS3 这样新出现的标准的采用率都在提高。以前只能在胖客户应用程序中实现的功能，目前在 Web 上已成为现实。浏览器每天都变得越来越快、越来越好，常见任务也逐渐成为可重用的库，几乎所有大型的 Web 框架也开放了源代码，我们的 WWW 成为一个快乐的地方。移动设备和各种移动应用程序开发技术的渗透，使开发人员进一步把开放的、可访问的 Web 看作一个表演平台。

在这个过程中，像 jQuery 和 jQuery mobile 这样的客户端库、像 ASP.NET 这样的服务器端技术使通常很难完成的繁琐任务不再遥不可及。其中，像 Visual Web Developer 2012 这样的免费工具使 Web 开发比以前有趣得多。目前，成为 Web 开发人员的确是很有趣的，而本书的出版，使每个人都能成为 Web 开发人员。

本书的作者 Imar Spaanjaars 自 2008 年以来一直是 Microsoft 的“最有价值专家(MVP)”，那时我们还邀请他加入了 ASP.NET Insiders 小组，在这个小组中，我们交流对各种功能和尚未公开发布的预发布产品的看法。团队在开发过程中，Imar 一直在提供反馈，将来他也将继续这么做。

本书的开头比较浅显，渐渐深入地探讨概念，并介绍了 ASP.NET 4.5 和 Visual Studio 2012 中的最新功能。无论读者是刚开始 Web 开发，还是更新到 ASP.NET 4.5，本书都值得添加到工具箱中。

我很高兴认识 Imar，感谢他对我们社区的贡献。他的远见卓识对于 ASP.NET 和 Visual Studio 产品团队而言是无价之宝。我希望他的这些远见卓识也对你有帮助。

Microsoft 公司 Windows Azure 小组的程序管理领队 Vishal R. Joshi  
<http://vishalrjoshi.com>

# 前言

为了构建有效且有吸引力的数据库驱动的 Web 站点，需要两个条件：一个是运行 Web 页面的稳固而快速的框架；另一个是创建和编写 Web 页面的丰富而广泛的环境。通过 ASP.NET 4.5 和 Visual Studio 2012 可以满足这两个条件。它们结合在一起形成了一个创建动态的、交互式的 Web 站点的平台。

ASP.NET 4.5 建立在它广受欢迎的前身 ASP.NET 2.0、3.5 和 4.0 基础之上。除了维持对使用老版本构建的 Web 站点的向后兼容性外，ASP.NET 4.5 和 Visual Studio 2012 还增加了大量新的、激动人心的功能，并对框架和开发工具进行了一些较小的但很有用的改进。

自 Visual Studio 2003 以来，对于每一个 Visual Studio 新版本的发布，我都惊讶于 Microsoft 在产品中添加的新功能的数量和对产品所做的改进。Visual Studio 2012 也不例外。如果你熟悉旧版本，就会注意到该版本的一个主要变化是其新式的设计。Visual Studio 的 UI 更新为 Windows 8 设计外观，且更好地与 Microsoft 的其他产品保持一致。

在 ASP.NET Framework 和 Visual Studio 中有许多大大小小的变化。其中一些变化是改进了的 CSS 和 JavaScript 编辑器(分别参见第 3 章和第 10 章)，强类型化的数据绑定控件(参见第 14 章)，包含 NuGet(参见第 11 章)，引入了 Bundling 和 Minification(参见第 19 章)。

我最喜欢的新功能是 Page Inspector，它有助于同时调试客户端和服务端代码，参见第 18 章。

熟悉 ASP.NET 早期版本的读者会很高兴地发现，新版本中的许多小功能将大大简化开发工作。我们将在本书的适当位置介绍并讨论这些新功能。有关 ASP.NET 新功能的完整列表，请查看 ASP.NET 官方网站(<http://www.asp.net/vnext/overview/whitepapers/whats-new>)上的白皮书。

如果阅读本书时，该链接不再有效，可以在 [www.asp.net](http://www.asp.net) 上搜索 What's new in ASP.NET 4.5。

关于 Visual Studio Express 2012 for Web 最好的消息可能是它的价格：现在仍然可以免费使用。因此，Visual Studio 和 ASP.NET 可能是如今最引人注目的 Web 开发技术。

## 0.1 本书读者对象

本书适用于想了解如何在 Microsoft 平台上构建丰富的交互式 Web 站点的任何人。利用从本书学到的知识，可以为构建各种类型的 Web 站点(从简单的只是业余爱好的 Web 站点，到为商业目的创建的站点)打下基础。

Web 编程新手也能使用本书，因为本书并没有事先假定读者具有 Web 开发背景(虽然了解基本的 HTML 和 Web 概念确实很有用)。本书从头开始介绍 Web 开发，说明了如何获得与

安装 Visual Studio。后面的章节建立在前面章节的基础上，循序渐进地介绍新技术。

你是否更喜欢 Visual Basic 而不是 C#；或者反过来，更喜欢 C#而不喜欢 Visual Basic？或者认为这两种语言都不错？或者还没有决定要学哪种语言，或想两种都学？无论是哪种情况，你都会喜欢本书，因为本书的所有代码示例都是用这两种语言表示的。

即使已经熟悉了 ASP.NET 以前的版本，还是可以从本书中获益。虽然 ASP.NET 4.5 沿用了以前版本中的很多概念，但是在本书中还是可以发现大量新内容，包括强类型化的数据控件，更智能的代码编辑器，新的调试功能等。

## 0.2 本书主要内容

本书将介绍如何构建一个名为 Planet Wrox 的功能丰富的、数据驱动的交互式 Web 站点。虽然这句话很长，但是你会发现使用 Visual Studio 2012 来开发这样的 Web 站点并不像看起来那样困难。本书将介绍构建 Web 站点的整个过程，从第 1 章安装 Visual Studio，一直到第 19 章将 Web 站点部署到真正的服务器上。本书分为 19 章，每一章重点介绍一个特定主题。

- **第 1 章“ASP.NET 4.5 入门”**：该章介绍如何获得并安装 Visual Studio 2012 的免费版本 Visual Studio Express 2012 for Web 来建立 ASP.NET 网站。该章还将介绍定义 Web 页面的最新标准 HTML5，最后概述了 Visual Studio 提供的自定义选项。
- **第 2 章“构建 ASP.NET Web 站点”**：该章介绍如何创建一个新的 Web 站点，以及如何向它增加新元素，如页面。除了学习如何创建结构良好的站点外，还会介绍如何用 Visual Studio 中的大量工具来创建 HTML 与 ASP.NET 页面。
- **第 3 章“设计 Web 页面”**：Visual Studio 附带了大量的工具用于创建设计良好的、有吸引力的 Web 页面。该章将说明如何充分利用这些工具。此外，还会介绍 CSS 这种用来定义 Web 页面格式的语言。
- **第 4 章“使用 ASP.NET 服务器控件”**：ASP.NET 服务器控件是 ASP.NET 中最重要的概念之一，它们允许使用少量代码创建复杂而功能丰富的 Web 站点。该章介绍了大量可用的服务器控件，解释了它们的用途，并说明了它们的用法。
- **第 5 章“ASP.NET Web 页面编程”**：虽然内置 CSS 工具与 ASP.NET 服务器控件非常有助于创建 Web 页面，但是使用编程语言能够增强页面。该章花了大量篇幅介绍 Web 页面编程。值得一提的是，该章(以及本书余下章节)的所有示例都使用 Visual Basic 和 C#两种语言表述，因此可以选择一种最喜欢的语言。
- **第 6 章“创建外观一致的 Web 站点”**：一致性比较容易使 Web 站点具有吸引力且给人比较专业的印象。ASP.NET 通过使用母版页来帮助创建外观一致的页面。母版页可以用来定义页面的全局外观。外观(skin)和主题有助于集中控件和 Web 站点中其他可视化元素的外观。该章还会介绍如何创建基页来帮助集中站点中所有页面都需要的编程代码。
- **第 7 章“导航”**：为了帮助访问者在站点中找到浏览路径，ASP.NET 配置了一些导航控件。这些控件用来构建站点的导航结构。可以将它们连接到站点的集中站点地图(定



义 Web 站点中的页面)。该章还会介绍如何通过编写程序将用户从一个页面发送到另一个页面。

- **第 8 章“用户控件”**：用户控件是可用在多个 Web 页面中的可重用页面片段。因此它们对于一些重复内容(如菜单、横幅等)很有用。该章将介绍如何创建与使用用户控件，并用一些程序化的智能来增强它们。
- **第 9 章“验证用户输入的有效性”**：站点中的大部分交互性是通过用户的输入定义的。该章介绍如何使用 ASP.NET 服务器控件接受、验证和处理用户输入。此外，还将介绍如何从 ASP.NET Web 站点中发送电子邮件，以及如何从文本文件中读信息。
- **第 10 章“ASP.NET AJAX”**：Microsoft ASP.NET AJAX 允许创建漂亮、无闪烁的 Web 页面，消除了传统桌面应用程序与 Web 应用程序之间的差距。本章将介绍如何用内置的 Ajax 功能增强 Web 页面的表现，从而获得与 Web 站点更平滑的交互。
- **第 11 章“jQuery”**：jQuery 是一个流行的、开源的且跨浏览器的 JavaScript 库，专用于简化与客户端浏览器的 Web 页面交互。该章介绍了 jQuery 的基础知识以及如何给 Web 页面添加丰富的可视化效果和动画。
- **第 12 章“初识数据库”**：了解如何使用数据库对于构建 Web 站点是至关重要的，因为大多数现代 Web 站点都要求使用数据库。该章将介绍 SQL(访问和更改数据库中数据的查询语言)的基础知识。此外，还将介绍 Visual Studio 中帮助创建和管理 SQL Server 数据库的数据库工具。
- **第 13 章“显示和更新数据”**：该章建立在从第 12 章学到的知识的基础上，说明了如何使用 ASP.NET 数据绑定控件与数据源控件创建一个丰富的界面，使用户能与这些控件的目标数据库中的数据交互。
- **第 14 章“LINQ 和 ADO.NET Entity Framework”**：LINQ 是 Microsoft 的解决方案，用来访问对象、数据库、XML 等。ADO.NET Entity Framework(EF)是 Microsoft 数据库访问的新技术。该章将介绍 LINQ 的概念，如何使用内置在 Visual Studio 中的可视化 EF 设计器，如何编写 LINQ to Entity 查询来让数据进出 SQL Server 数据库，以及如何使用强类型化的数据控件来简化代码的编写，且错误更少。
- **第 15 章“处理数据——高级主题”**：前面的章节大多集中于处理数据的技术基础，而该章从前端角度来看同样的主题。该章将介绍如何使用控件样式来改变数据的可视化外观。该章还将介绍与数据绑定控件的交互，以及如何通过保持经常访问的数据的本地副本来加速 Web 站点。
- **第 16 章“ASP.NET 4.5 Web 站点中的安全性”**：虽然安全性在本书中提出得相当晚，但是安全性是最首要且重要的主题。该章将介绍如何使用与安全性相关的内置 ASP.NET 功能。介绍若干促进安全性的应用程序服务。还将介绍如何让用户在 Web 站点上注册一个账户，如何区分匿名与登录用户，以及如何管理系统中的用户。
- **第 17 章“个性化 Web 站点”**：该章建立在第 16 章介绍的安全性功能基础之上，介绍了如何用针对个人用户的内容创建个性化 Web 页面。该章将介绍如何配置与使用 ASP.NET Profile 为已知和匿名访问者存储个性化数据。
- **第 18 章“异常处理、调试和跟踪”**：为了理解、改进和修复为 ASP.NET Web 页面编写的代码，需要有优秀的调试工具。Visual Studio 提供了出色的调试支持，可以在运行

时诊断应用程序的状态，帮助在用户发现问题之前解决问题。还介绍了 Visual Studio 2012 中的 Page Inspector。

- **第 19 章“部署 Web 站点”**：到本书末尾，应当有一个准备公布于众的 Web 站点。但是具体如何做呢？要发布 Web 站点，需要知道和了解哪些事情呢？该章给出了这些问题的答案，并说明为了运行最终的 Web 站点，如何配置不同的生产系统。另外还介绍了如何实现 Bundling 和 Minification 功能来提升网站的性能。

## 0.3 本书组织结构

本书通过运行示例和详细说明逐步地介绍概念。使用 Wrox 惯有的“试一试”练习与“工作原理”部分，可带领你一步步完成任务，并在任务进行过程中详细说明重要的地方。每个“试一试”后面都有一个详细的“工作原理”部分，用来解释在该练习中执行的步骤。

在每章的末尾都有一些练习题，可以帮助测试从该章学到的知识。各个问题的答案在本书最后的附录 A 中可以找到。如果不知道问题的所有答案也不要担心，后面的章节不会假定你已经完成了前面章节中练习部分的任务。

由于这是一本面向初学者的书，所以对一些主题的介绍不是很详细。本书的每一章中都会提到一些专门介绍该章所讨论主题的其他书籍。在适当的时候还包含了对这些书的引用，所以当想要深入研究某一个特定主题时，可以很容易决定下一步该怎么办。

## 0.4 使用本书所需条件

本书假定你有一个符合下列要求的系统：

- 能够运行 Visual Studio。要了解具体系统要求，请参见该软件附带的 readme 文件。
- 运行 Windows 7 或 Windows 8(两者都要求至少是 Home Premium 版)，或者是 Windows Server 2008 R2 或 2012 版本。

第 1 章介绍了如何获得并安装 Visual Studio 2012，然后安装 Microsoft .NET Framework 4.5 和 SQL Server Express LocalDB 版本，你所需要的只是一个优秀的操作系统以及阅读本书的动力！

## 0.5 源代码

读者在学习本书中的示例时，既可以手动输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从站点 <http://www.wrox.com/>或 [www.tupwk.com.cn/downpage](http://www.tupwk.com.cn/downpage) 上下载。登录到站点 <http://www.wrox.com/>，使用 Search 工具或使用书名列表就可以找到本书。接着单击本书细目页面上的 Download Code 链接，就可以获得所有源代码。



**提示：**由于许多图书的书名都很类似，所以按 ISBN 进行搜索是最简单的，本书英文版的 ISBN 是 978-1-118-31180-6。

可以将本书的所有源代码作为一个文件下载(可根据语言来选择版本,C#或 Visual Basic),然后用喜欢的解压缩工具对其进行解压缩即可。提取源代码时,请确保维持作为代码下载一部分的原始文件夹结构。不同的解压缩工具对这个功能有着不同的名称,不过尽可能寻找一个像 User Folder Names 或 Maintain Directory Structure 这样的功能。从下载的代码中提取了文件之后,最后应有一个名为 Source 的文件夹以及一个名为 Resources 的文件夹。然后在 C 盘的根位置创建一个新文件夹,命名为 BegASPNET,并将 Source 和 Resources 文件夹移到这个新文件夹中,最后得到类似 C:\BegASPNET\Source 和 C:\BegASPNET\Resources 这样的文件夹。Source 文件夹中包含本书 19 章中每一章的源代码文件,以及 Planet Wrox Web 站点的最终版本。Resources 文件夹包含本书的一些练习中所需要的文件。如果一切正常,最后应看到图 0-1 所示的结构。

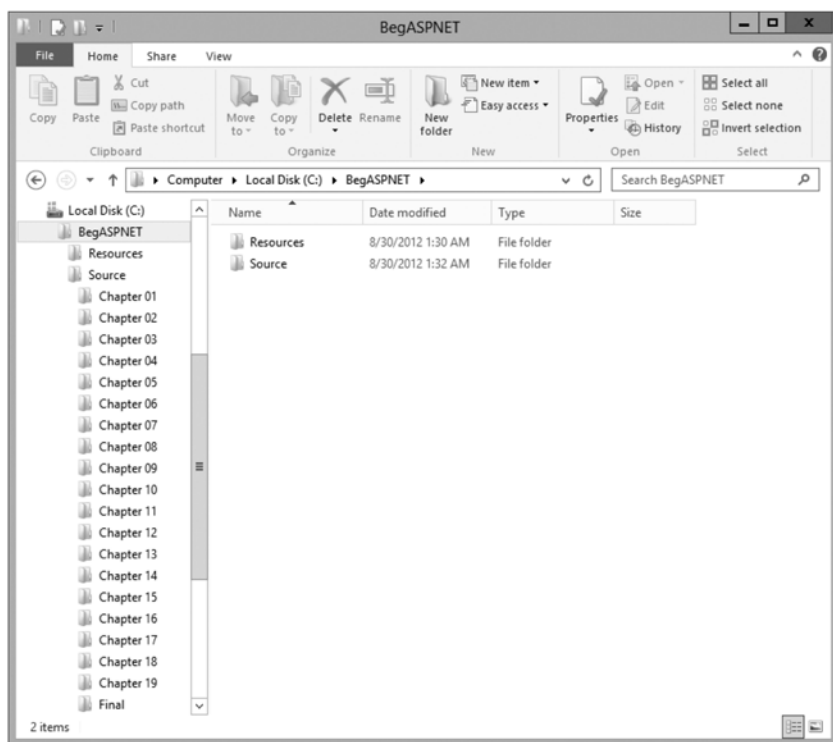


图 0-1

在以后的章节中将在 C:\BegASPNET 文件夹中创建名为 Site 和 Release 的文件夹,从而文件夹结构将为图 0-2 所示的样子。

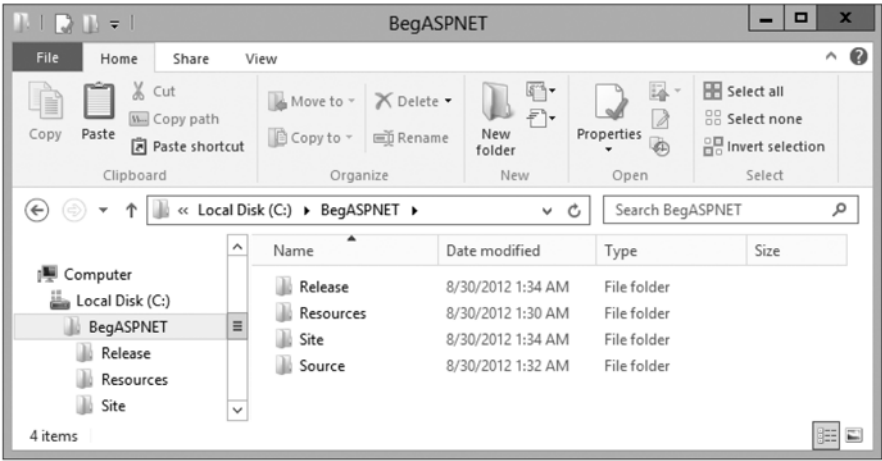


图 0-2

Site 文件夹包含本书将要构建的站点，而 Release 文件夹将包含本书末尾的站点的最终版本。每当做本书的一些练习受阻时，都可以打开 Source 文件夹查看一切最后应是什么样。

如果要为特定章节运行站点来看看它是如何工作的，一定要在 Visual Studio 中打开那一章的文件夹作为一个 Web 站点。因此应直接打开诸如 C:\BegASPNET\Source\ Chapter 12 这样的文件夹，而不是打开它的父文件夹 C:\BegASPNET\Source。

如果想要使用两种编程语言完成操作，则创建第二个文件夹 C:\BegASPNETVB 或 C:\BegASPNETCS 来存放另一个版本的文件。这样一来，这两个站点就可以共存而不产生冲突。如果专门为 C# 语言创建一个文件夹，请不要包含 # 符号。因为对一个 Web 站点来说，路径名中的 # 是一个无效字符。

坚持采用这个结构可以确保顺利执行本书的“试一试”练习。错误地混合或嵌套这些文件夹会使练习的完成变得困难，还可能导致发生预料之外的情况和错误。每当遇到本书中没有解释的问题或错误时，请确保站点结构仍然与这里提出的结构紧密相关。

## 0.6 勘误表

尽管我们已经尽了最大的努力来保证文章或代码中不出现错误，但是错误总是难免的，如果你在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者避免走入误区，当然，这还有助于提供更高质量的信息。

要在网站上找到本书英文版的勘误表，可以登录 [www.wrox.com/remtitle.cgi?isbn=1118311809](http://www.wrox.com/remtitle.cgi?isbn=1118311809)，或者访问 <http://www.wrox.com>，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Errata 链接。在这个页面上可以查看到 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 [www.wrox.com/misc-pages/booklist.shtml](http://www.wrox.com/misc-pages/booklist.shtml)。

如果你在勘误表上没有找到错误，那么可以到 [www.wrox.com/contact/techsupport.shtml](http://www.wrox.com/contact/techsupport.shtml)

上，完成上面的表格，并把找到的错误发送给我们。我们将会核查这些信息，如果无误的话，会把它放置到本书的勘误表中，并在本书的后续版本中更正这些问题。

## 0.7 p2p.wrox.com

要与作者和同行讨论，请加入 [p2p.wrox.com](http://p2p.wrox.com) 上的 P2P 论坛。这个论坛是一个基于 Web 的系统，便于你张贴与 Wrox 图书相关的消息和相关技术，与其他读者和技术用户交流心得。该论坛提供了订阅功能，当论坛上有新的消息时，它可以给你传送感兴趣的论题。Wrox 作者、编辑和其他业界专家和读者都会到这个论坛上来探讨问题。

在 <http://p2p.wrox.com> 上，有许多不同的论坛，它们不仅有助于阅读本书，还有助于开发自己的应用程序。要加入论坛，可以遵循下面的步骤：

- (1) 进入 [p2p.wrox.com](http://p2p.wrox.com)，单击 Register 链接。
- (2) 阅读使用协议，并单击 Agree 按钮。
- (3) 填写加入该论坛所需要的信息和自己希望提供的其他信息，并单击 Submit 按钮。
- (4) 你会收到一封电子邮件，其中的信息描述了如何验证账户和完成加入过程。

不加入 P2P 也可以阅读论坛上的消息，但要张贴自己的消息，就必须加入该论坛。

加入论坛后，就可以张贴新消息，回复其他用户张贴的消息。可以随时在 Web 上阅读消息。如果要想让该网站给自己发送特定论坛中的消息，可以单击论坛列表中该论坛名旁边的 **Subscribe to this Forum** 图标。

关于使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作情况以及 P2P 和 Wrox 图书的许多常见问题的解答。要阅读 FAQ，可以在任意 P2P 页面上单击 FAQ 链接。

# 目 录

第 1 章 ASP.NET 4.5 入门 .....	1		
1.1 Microsoft Visual Studio			
Express for Web .....	2		
1.1.1 获取 Visual Studio .....	2		
1.1.2 安装 Visual Studio			
Express(VSEW) .....	3		
1.2 创建第一个 ASP.NET 4.5			
Web 站点 .....	5		
1.3 ASP.NET 4.5 简介 .....	8		
1.3.1 HTML .....	9		
1.3.2 初识 ASP.NET 标记 .....	14		
1.4 IDE .....	14		
1.4.1 主开发区 .....	14		
1.4.2 信息窗口 .....	19		
1.5 定制 IDE .....	20		
1.5.1 重新排列窗口 .....	20		
1.5.2 修改 Toolbox .....	21		
1.5.3 定制文档窗口 .....	23		
1.5.4 定制工具栏 .....	23		
1.5.5 定制键盘快捷键 .....	24		
1.5.6 重置修改 .....	24		
1.6 示例应用程序 .....	25		
1.7 关于 Visual Studio 的实用提示 .....	27		
1.8 本章小结 .....	27		
1.9 练习 .....	27		
第 2 章 构建 ASP.NET Web 站点 .....	29		
2.1 使用 Visual Studio 2012			
创建 Web 站点 .....	29		
2.1.1 不同的项目类型 .....	30		
2.1.2 选择正确的 Web 站点模板 .....	31		
2.1.3 创建与打开新的 Web 站点 .....	32		
2.2 操作 Web 站点中的文件 .....	35		
2.2.1 ASP.NET 4.5 Web 站点的			
文件类型 .....	35		
2.2.2 添加现有文件 .....	38		
2.2.3 组织站点 .....	39		
2.2.4 特殊文件类型 .....	40		
2.3 使用 Web 窗体 .....	40		
2.3.1 Web 窗体的不同视图 .....	41		
2.3.2 在 Code Behind 和带内联			
代码的页面之间选择 .....	42		
2.3.3 向页面添加标记 .....	46		
2.3.4 连接页面 .....	52		
2.4 使用 Web 窗体的实用提示 .....	53		
2.5 本章小结 .....	54		
2.6 练习 .....	54		
第 3 章 设计 Web 页面 .....	55		
3.1 需要 CSS 的原因 .....	55		
3.1.1 HTML 格式化的问题 .....	56		
3.1.2 CSS 如何解决格式化问题 .....	56		
3.2 CSS 简介 .....	57		
3.2.1 CSS 语言 .....	61		
3.2.2 样式表 .....	61		
3.2.3 向页面中添加 CSS .....	73		
3.3 在 Visual Studio 中使用 CSS .....	75		
3.3.1 使用 CSS 编辑器 .....	75		
3.3.2 创建内嵌和内联样式表 .....	79		
3.3.3 应用样式 .....	84		

3.3.4 管理样式 .....	85	5.3.1 运算符 .....	132
3.4 关于使用 CSS 的实用提示 .....	88	5.3.2 做决策 .....	140
3.5 本章小结 .....	89	5.3.3 循环 .....	146
3.6 练习 .....	89	5.4 组织代码 .....	150
<b>第 4 章 使用 ASP.NET 服务器控件 .....</b>	<b>91</b>	5.4.1 方法: 函数与子例程 .....	151
4.1 服务器控件简介 .....	91	5.4.2 App_Code 文件夹 .....	153
4.2 ASP.NET 服务器控件详解 .....	95	5.4.3 使用名称空间组织代码 .....	157
4.2.1 在页面中定义控件 .....	95	5.4.4 写注释 .....	159
4.2.2 所有控件的共同属性 .....	96	5.5 面向对象编程基础知识 .....	161
4.3 控件的类型 .....	98	5.5.1 重要的面向对象术语 .....	161
4.3.1 标准控件 .....	98	5.5.2 事件 .....	172
4.3.2 HTML 控件 .....	111	5.6 关于编程的实用提示 .....	173
4.3.3 数据控件 .....	111	5.7 本章小结 .....	174
4.3.4 有效性验证控件 .....	112	5.8 练习 .....	174
4.3.5 导航控件 .....	112	<b>第 6 章 创建外观一致的 Web 站点 .....</b>	<b>177</b>
4.3.6 登录控件 .....	112	6.1 用母版页创建一致的	
4.3.7 Ajax 扩展 .....	112	页面布局 .....	177
4.3.8 WebParts .....	112	6.1.1 创建母版页 .....	179
4.3.9 动态数据 .....	112	6.1.2 创建内容页 .....	181
4.4 ASP.NET 状态引擎 .....	113	6.2 使用集中的基页 .....	186
4.4.1 状态的定义及其重要性 .....	113	6.2.1 ASP.NET 页面生命周期 .....	187
4.4.2 状态引擎的工作原理 .....	113	6.2.2 实现基页 .....	189
4.4.3 并非所有控件都依赖		6.2.3 创建可重用的页面模板 .....	193
View State .....	117	6.3 主题 .....	196
4.4.4 关于 View State 和性能的		6.3.1 不同类型的主题 .....	197
一个注意点 .....	118	6.3.2 在 Theme 和 StyleSheetTheme	
4.5 使用控件的实用提示 .....	119	之间进行选择 .....	197
4.6 本章小结 .....	119	6.3.3 应用主题 .....	197
4.7 练习 .....	120	6.3.4 扩展主题 .....	201
<b>第 5 章 ASP.NET Web 页面编程 .....</b>	<b>121</b>	6.3.5 动态切换主题 .....	203
5.1 编程简介 .....	122	6.4 外观 .....	211
5.2 数据类型与变量 .....	122	6.4.1 创建 skin 文件 .....	212
5.2.1 转换数据类型 .....	125	6.4.2 已命名外观 .....	213
5.2.2 使用数组和集合 .....	127	6.4.3 对特定控件禁用主题 .....	214
5.3 语句 .....	131	6.5 创建一致页面的实用提示 .....	214
		6.6 本章小结 .....	215

6.7 练习	215	9.1.1 验证 Web 窗体中用户 输入的有效性	269
第 7 章 导航	217	9.1.2 理解请求有效性验证	285
7.1 在站点中移动的不同方式	217	9.2 在服务器上处理数据	286
7.1.1 理解绝对 URL 与相对 URL	218	9.2.1 从 Web 站点中发送电子邮件	286
7.1.2 默认文档	220	9.2.2 从文本文件中读取数据	292
7.2 使用导航控件	220	9.3 关于验证数据有效性的 实用提示	297
7.2.1 导航控件的体系结构	221	9.4 本章小结	297
7.2.2 分析 Web.sitemap 文件	221	9.5 练习	298
7.2.3 使用 Menu 控件	223	第 10 章 ASP.NET AJAX	299
7.2.4 使用 TreeView 控件	231	10.1 Ajax 简介	300
7.2.5 使用 SiteMapPath 控件	234	10.2 在项目中使用 ASP.NET AJAX	301
7.3 以编程的方式重定向	236	10.2.1 创建无闪烁页面	301
7.3.1 通过编程将客户重定向到 不同页面	236	10.2.2 给用户反馈	306
7.3.2 服务器端重定向	239	10.2.3 Timer 控件	310
7.4 关于导航的实用提示	240	10.3 在 Ajax Web 站点中使用 Web 服务和页面方法	311
7.5 本章小结	241	10.3.1 Web 服务的定义	311
7.6 练习	241	10.3.2 创建 Web 服务	316
第 8 章 用户控件	243	10.3.3 页面方法简介	324
8.1 用户控件简介	243	10.4 有关 Ajax 的实用提示	326
8.1.1 创建用户控件	244	10.5 本章小结	327
8.1.2 向内容页或母版页中 添加用户控件	247	10.6 练习	327
8.1.3 用户控件的站点范围注册	250	第 11 章 jQuery	329
8.1.4 关于用户控件的警告	251	11.1 jQuery 简介	330
8.2 向用户控件添加逻辑	253	11.1.1 NuGet 简介	330
8.2.1 为属性创建自己的数据类型	253	11.1.2 选择引用 jQuery 的位置	334
8.2.2 实现 View State 属性	258	11.1.3 包含 jQuery 库的不同方式	334
8.2.3 关于 View State 要考虑的事项	263	11.2 jQuery 语法	337
8.3 关于用户控件的实用提示	264	11.2.1 jQuery Core	338
8.4 本章小结	264	11.2.2 使用 jQuery 进行选择	338
8.5 练习	264	11.3 使用 jQuery 修改 DOM	345
第 9 章 验证用户输入的有效性	267	11.3.1 CSS 方法	345
9.1 收集用户数据	268		



11.3.2 处理事件 .....	346	13.2.3 在 Web.config 文件中 存储连接字符串 .....	395
11.3.3 jQuery 的各种功能 .....	347	13.2.4 筛选数据 .....	396
11.3.4 使用 jQuery 时常犯的错误 .....	348	13.3 自定义数据控件的外观 .....	401
11.4 使用 jQuery 的效果 .....	349	13.4 更新和插入数据 .....	407
11.5 jQuery 和有效性验证 .....	354	13.5 显示和更新数据的实用提示 .....	420
11.6 关于 jQuery 的实用提示 .....	357	13.6 本章小结 .....	420
11.7 本章小结 .....	358	13.7 练习 .....	420
11.8 练习 .....	358		
<b>第 12 章 初识数据库 .....</b>	<b>359</b>	<b>第 14 章 LINQ 和 ADO.NET Entity Framework .....</b>	<b>423</b>
12.1 数据库的概念 .....	360	14.1 LINQ 简介 .....	423
12.2 不同类型的关系数据库 .....	361	14.1.1 LINQ to Objects .....	424
12.3 运用 SQL 处理数据库数据 .....	362	14.1.2 LINQ to XML .....	424
12.4 使用 SQL 检索和操纵数据 .....	364	14.1.3 LINQ to ADO.NET .....	425
12.4.1 读取数据 .....	364	14.2 ADO.NET Entity Framework 简介 .....	425
12.4.2 创建数据 .....	372	14.3 将数据模型映射到对象模型 .....	426
12.4.3 更新数据 .....	372	14.4 查询语法 .....	432
12.4.4 删除数据 .....	373	14.4.1 标准查询操作符 .....	432
12.5 创建自己的表 .....	375	14.4.2 用匿名类型定义数据 .....	436
12.5.1 SQL Server 中的数据类型 .....	375	14.5 结合使用服务器控件和 LINQ 查询 .....	441
12.5.2 了解主键和标识列 .....	377	14.5.1 在 Entity Framework 中 使用数据控件 .....	441
12.5.3 创建表之间的关系 .....	379	14.5.2 有关性能的一些注意点 .....	469
12.6 有关数据库的实用提示 .....	383	14.6 有关 LINQ 和 ADO.NET Entity Framework 的实用提示 .....	469
12.7 本章小结 .....	383	14.7 本章小结 .....	469
12.8 练习 .....	384	14.8 练习 .....	470
<b>第 13 章 显示和更新数据 .....</b>	<b>385</b>	<b>第 15 章 处理数据——高级主题 .....</b>	<b>471</b>
13.1 数据控件 .....	385	15.1 使用样式格式化控件 .....	471
13.1.1 数据绑定控件 .....	386	15.1.1 关于样式 .....	472
13.1.2 数据源控件 .....	387	15.1.2 合并样式、主题和外观 .....	475
13.1.3 其他数据控件 .....	387	15.2 处理事件 .....	479
13.2 联合使用数据源和数据 绑定控件 .....	388	15.2.1 回顾 ASP.NET 页面和 控件生命周期 .....	479
13.2.1 使用 GridView 显示和 编辑数据 .....	388		
13.2.2 使用 DetailsView 插入数据 .....	393		

15.2.2 ASP.NET 页面生命周期 和数据控件中的事件 .....	485	17.2 Profile 的其他使用方法 .....	573
15.2.3 处理数据源控件中发生 的错误 .....	489	17.2.1 匿名标识 .....	573
15.3 手动编写数据访问代码 .....	493	17.2.2 清除旧的匿名配置文件 .....	573
15.4 缓存 .....	502	17.2.3 查看其他用户的配置文件 .....	575
15.4.1 缓存数据的常见问题 .....	503	17.3 关于个性化的实用提示 .....	577
15.4.2 在 ASP.NET Web 应用程序 中缓存数据的不同方法 .....	504	17.4 本章小结 .....	578
15.5 有关数据的实用提示 .....	512	17.5 练习 .....	578
15.6 本章小结 .....	513	第 18 章 异常处理、调试和跟踪 .....	581
15.7 练习 .....	513	18.1 异常处理 .....	581
第 16 章 ASP.NET 4.5 Web 站点 中的安全性 .....	515	18.1.1 不同类型的错误 .....	582
16.1 安全性 .....	515	18.1.2 捕获和处理异常 .....	583
16.1.1 身份：你是谁 .....	516	18.1.3 全局错误处理和 自定义错误页面 .....	591
16.1.2 身份验证：如何证明 你是谁 .....	516	18.2 调试基础知识 .....	598
16.1.3 授权：允许你做什么 .....	516	18.3 调试的工具支持 .....	600
16.1.4 ASP.NET 应用程序服务 .....	516	18.3.1 在调试代码中移动 .....	600
16.2 登录控件 .....	518	18.3.2 调试窗口 .....	601
16.2.1 登录控件 .....	522	18.4 调试客户端脚本 .....	607
16.2.2 配置 Web 应用程序 .....	533	18.5 用 Page Inspector 调试 .....	609
16.3 Role Manager .....	535	18.5.1 Page Inspector 简介 .....	609
16.3.1 配置 Role Manager .....	535	18.5.2 使用 Page Inspector .....	610
16.3.2 使用 WSAT 管理用户 .....	536	18.6 跟踪 ASP.NET Web 页面 .....	614
16.3.3 配置 Web 应用程序以 使用角色 .....	539	18.6.1 使用标准的跟踪功能 .....	615
16.3.4 以编程方式检查角色 .....	543	18.6.2 添加自己的信息到 Trace 中 .....	617
16.4 有关安全性的实用提示 .....	547	18.6.3 跟踪和性能 .....	619
16.5 本章小结 .....	547	18.6.4 安全警告 .....	619
16.6 练习 .....	547	18.7 有关调试的实用提示 .....	620
第 17 章 个性化 Web 站点 .....	549	18.8 本章小结 .....	621
17.1 Profile .....	550	18.9 练习 .....	621
17.1.1 配置 Profile .....	550	第 19 章 部署 Web 站点 .....	623
17.1.2 使用 Profile .....	556	19.1 准备部署 Web 站点 .....	623
		19.1.1 避免硬编码的设置 .....	624
		19.1.2 Web.config 文件 .....	624
		19.1.3 表达式语法 .....	624
		19.1.4 WebConfigurationManager 类 .....	626

19.2	Bundling 和 Minification 简介	631	19.5.1	把数据导出到文件中	649
19.3	复制 Web 站点	633	19.5.2	重建数据库	651
19.3.1	创建 Web 站点的简单副本	634	19.6	部署清单	652
19.3.2	发布 Web 站点	637	19.7	补充资源	653
19.4	在 IIS 下运行站点	638	19.8	本章小结	654
19.4.1	安装和配置 Web 服务器	638	19.9	练习	654
19.4.2	安装和配置 ASP.NET	640	附录 A	练习答案	655
19.4.3	了解 IIS 中的安全性	643	附录 B	配置 SQL Server 2012	679
19.4.4	Planet Wrox 的 NTFS 设置	644			
19.4.5	检修 Web 服务器错误	647			
19.5	将数据移动到远程服务器中	649			

# 第 1 章

## ASP.NET 4.5 入门

### 本章要点

---

- 如何获取和安装 Visual Studio Express 2012 for Web 和 Visual Studio 2012
- 如何使用 Visual Studio Express 创建第一个 Web 站点
- 服务器如何处理 ASP.NET 页面并将其传送给浏览器
- 如何使用和定制开发环境

自从 .NET Framework 1.0 在 2002 年初首次发布以来, Microsoft 花了大量精力和时间来开发 ASP.NET, 它是 .NET Framework 的一部分, 可以用来构建富 Web 应用程序。首次发布意味着从过去的 Microsoft 技术向构建 ASP (Active Server Page, 活动服务器页面, 现在人们常称之为传统 ASP) Web 站点的飞跃。与传统 ASP 相比, ASP.NET 1.0 及相关的 Visual Studio .NET 的引入给开发人员带来了如下好处:

- 页面显示与代码清楚地分开。使用传统 ASP 时, 编程逻辑常常散布在整个页面的 HTML 中, 使得后面对页面的修改比较困难。
- 开发模型更接近于桌面应用程序的编程方式。这样很多 Visual Basic 桌面程序员可以轻松地转换到 Web 应用程序。
- 它有一个功能丰富的开发工具(称为 Visual Studio .NET), 开发人员可以通过它可视化地创建和编写 Web 应用程序代码。
- 有几种面向对象的编程语言可供选择, 其中 Visual Basic .NET 和 C#(读作 C-Sharp)是目前最流行的两种语言。
- 它可以访问整个 .NET Framework, 这意味着 Web 开发人员首次拥有了一种统一且容易的方式, 来使用数据库、文件、e-mail、网络工具等许多高级功能。

尽管 ASP.NET 远优于旧模型, 但使用它也意味着构建应用程序时的复杂性以及所需知识量的增加, 所以它对于许多新的程序员来说, 更难上手。

在 2002 年首次发布以后, Microsoft 在 2003 年发布了 .NET Framework 的另一个版本(称为 .NET 1.1)和开发 IDE(称为 Visual Studio .NET 2003)。尽管在架构和开发工具方面都有了不少新的改进, 但

很多人还是习惯把这些看做是初始版本的一个服务包。

2005 年 11 月, Microsoft 发布了 Visual Studio 2005 和 ASP.NET 2.0。让全球许多开发人员感到惊喜的是, Microsoft 又大大改进和扩展了产品, 增加了许多功能和工具来帮助降低 ASP.NET 1.0 所带来的复杂性。新的向导和智能控件减少了构建应用程序所需的代码, 降低了新开发人员的学习难度, 并且提高了开发效率。

2007 年 11 月, Microsoft 发布了 Visual Studio 2008 和 ASP.NET 3.5 框架, 之后又在 2010 年 3 月发布了 Visual Studio 2010 和 ASP.NET 4, 这两个版本都增加了许多新功能, 包括 LINQ(参见第 14 章), AJAX 框架整合(参见第 10 章), ADO.NET Entity Framework(参见第 14 章), 和 jQuery(参见第 11 章)。

目前的版本是 Visual Studio 2012 和 ASP.NET 4.5, 它是在已成功发行的 Visual Studio 2010 和 ASP.NET 4 基础之上构建的, 它保留了很多令人喜爱的功能, 并增加了一些其他领域的新功能和工具。

在本书接下来的 19 章中, 将会介绍如何使用 Visual Studio Express 2012 for Web(它是 Microsoft 为 ASP.NET Web 应用程序设计的免费开发工具, 也是 Visual Studio 2012 完整套装的一部分)构建功能完全的 ASP.NET Web 站点。本书将引导你了解构建功能完全的、数据库驱动的 Web 站点的过程, 从第 2 章的基本 Web 站点开始, 直到第 19 章将它部署到生产环境中。

本书的示例站点和所有示例都是用 Visual Studio Express 2012 for Web (VSEW)构建的, 因此需要把它安装到开发机器上。下一节将介绍如何获得和安装 VSEW。安装并运行 VSEW 后, 就学习如何创建第一个 Web 站点, 接着介绍 VSEW 的许多功能。

## 1.1 Microsoft Visual Studio Express for Web

尽管从理论上讲, 只用 Notepad 或其他文本编辑器就可以编写 ASP.NET Web 应用程序, 但你可能还是希望安装 Microsoft Visual Studio(VS)。VS 包含了大量有助于快速创建复杂 ASP.NET Web 应用程序的工具。

Visual Studio 有两个版本: 一个是独立而免费的版本, 称为 Microsoft Visual Studio Express 2012 for Web; 还有一个版本是作为较大的开发套件 Visual Studio 2012 的一部分, 它有不同的版本可用, 且各个版本的价格各不相同。使用 Visual Studio 的商用版本, 可以完全集成 Web 组件。只需要启动 Visual Studio 2012, 再创建一个 Web 站点项目或 Web 应用程序项目, 就可以启用 Visual Studio 的 Web 组件。

虽然 Visual Studio 的 Express 版本是免费的, 但是它包含了创建复杂且功能丰富的 Web 应用程序所需的所有功能和工具。本书中的所有示例都可以用免费的 Express 版本构建出来, 因此不需要为了学习本书而花大本钱去购买 Visual Studio 2012 的商业版本。

这里使用 Visual Studio(VS)表示 Visual Studio 的商用和免费版本。在专门提到免费版本时, 使用术语 Express 版本或 Visual Studio Express 2012(VSEW)。

Visual Studio 很容易获得。从 Microsoft 站点下载即可, 具体方法参见下一小节。

### 1.1.1 获取 Visual Studio

可以从 Microsoft 站点 [www.microsoft.com/express/](http://www.microsoft.com/express/) 上下载 Visual Studio Express 2012 for Web 的免

费版本。在 Express 的主页上,依次单击 Download 链接,直到打开提供了下载 Express 产品的页面,其中包括 VSEW。在这个页面上可以以 Web 安装方式下载 VSEW,这里只下载安装程序,文件的其余部分在安装过程中下载。一定要在这个页面上选择 Visual Studio Express 2012 for Web,而不要选择其他免费的 Express 产品或 Visual Studio 的以前版本。

不要被以 Web 安装方式下载的大约几 MB 的文件大小所迷惑。以这种方式下载的文件只是从 Internet 上下载必需文件的安装程序。全部的下载量依赖于当前的系统,大约在 180MB 到 270MB 之间。

如果想试试同样包含 Web 组件的 Visual Studio 2012 完整版本,则可以从 Microsoft 的站点 <http://msdn.microsoft.com/vstudio> 上下载有免费试用期的版本。如果需要刻录到 DVD 上,则可以选择下载一个 ISO 映像程序,或者选择下载 Web 安装程序。

最后,可以从 [www.microsoft.com/web/platform](http://www.microsoft.com/web/platform) 和 [www.asp.net/downloads/](http://www.asp.net/downloads/) 站点上下载 Microsoft Web Platform Installer(WPI)应用程序。除了 VSEW 以外,这个工具还便于访问其他许多与 Web 开发相关的工具和程序。通过使用 WPI 这个优秀的工具,可以同时获得大量与 Web 开发相关的程序和工具。我经常使用这个工具来快速建立一个开发环境。

### 1.1.2 安装 Visual Studio Express(VSEW)

VSEW 的安装很简单,只是过程有点长。根据所选的安装方法、计算机配置和 Internet 连接速度,安装 VSEW 可能需要 20 分钟到一个小时,甚至更长时间。

#### 试一试

#### 安装 Visual Studio Express 2012 for Web

本“试一试”练习用来指导如何在计算机上安装 VSEW。它假定用户选择前面介绍的 Web Platform Installer 选项,但从 DVD 安装 Express 版本的过程几乎也是这样。安装 Visual Studio 2012 的完整版本所需要执行的步骤与之相似,只是看到的屏幕略有不同。

不管安装 VSEW 的哪个版本,都要安装 Microsoft SQL Express LocalDB Edition 11.0——本书的很多示例都会用到这个组件。安装 VSEW 时,它应自动安装,但万一它没有安装,可以在本练习的最后找到确保正确安装它的指令。

(1) 首先浏览到 [www.microsoft.com/express/](http://www.microsoft.com/express/),按照指令下载 VSEW 2012。确保安装 Visual Studio Express 2012 for Web,而不要选择其他免费的 Express 产品或 Visual Studio 的以前版本。如果这个链接改变了或不再提供对 VSEW 下载的直接访问,则在 web 上搜索 install Visual Studio Express 2012 for Web,就会进入可以下载安装程序的下载页面。

(2) 根据启动安装程序的过程,此时有几个选项。如果启动 VSEW 下载程序,就会看到如图 1-1 所示的屏幕。

单击 Install,开始安装相关的组件。

如果启动了 Web Platform Installer,就在工具列表中找到 Visual Studio Express 2012 for Web,并选择它。最后,启动安装过程。

(3) 在这两种情况下,会显示一个屏幕,其中列出了要安装的组件。另外,还需要接受软件许可条款,之后,就应看到如图 1-2 所示的屏幕。



图 1-1

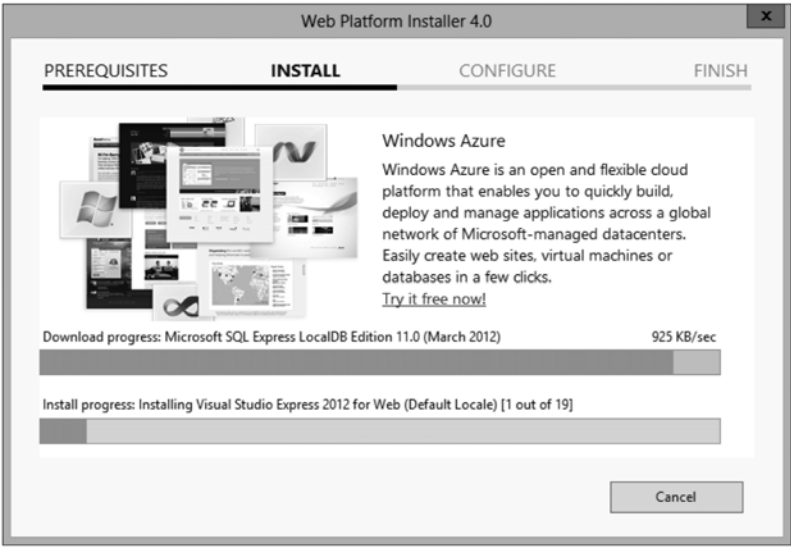


图 1-2

(4) 过一会儿，就会看到一个消息，说明 VSEW 已成功安装。在安装过程中和安装过程后，安装程序可能需要重启计算机。安装程序执行完毕后，VSEW 就可以使用了。为了检查 SQL Express LocalDB 是否正确安装，可以从 Windows 的“开始”菜单或“开始”屏幕上启动 Web Platform Installer。接着，在 Products 项下，找到 Microsoft SQL Express LocalDB Edition 11.0，如果它还没有安装，就安装它。

### 工作原理

这个简单的安装过程介绍了使用 Web Platform Installer 安装 VSEW 的过程。在安装过程中，WPI 还安装了 Microsoft SQL Express LocalDB Edition 11.0(Microsoft SQL Server 2012 数据库引擎的免费版本)的副本。从第 12 章开始，本书将多次讨论和使用 SQL Server 2012。附录 B 说明了如何对 SQL Server 2012 的各个版本进行安全设置。

现在已经安装了 VSEW，接下来应启动它并开始使用。下一节将介绍如何使用 VSEW 创建第一个站点。你将了解到如何创建站点、向 Web 页面中添加内容，以及如何在浏览器中浏览该页面。

## 1.2 创建第一个 ASP.NET 4.5 Web 站点

你可能已经迫不及待地想要开始创建第一个 ASP.NET Web 站点，所以，我们现在就不多进行 VS 中网站的理论概述了。下面的“试一试”练习将直接进入实战运用，指出如何构建第一个 Web 项目。然后，在“工作原理”部分及其后的小节中，会做一些解释，使你对在浏览器中浏览 ASP.NET 页面时后台的工作有一个很好的了解。

### 试一试

### 创建第一个 ASP.NET Web 站点

(1) 如果还没有启动 VS，则从 Windows 的“开始”菜单或“开始”屏幕中启动 VS。按照屏幕上的指令在线注册 VSEW，获得一个键。在 VSEW 起始屏幕中输入这个键，单击 Next 继续。注意，如果使用的是 Visual Studio 的商业版本，则只需要在“开始”菜单中启动 Visual Studio 2012。所有与 Web 相关的组件都从主 VS 程序中访问。如果是第一次启动 VS，则在开始使用 VS 工作之前，会有一些延迟，因为它要先做一些必要的配置。以后再启动时就会快得多。

(2) 如果使用的是 Visual Studio 的商业版本，则在首次启动 Visual Studio 时还会出现一个对话框，要求在不同的设置集中作出选择。在这个对话框中所作的选择将会影响窗口、工具箱、菜单和快捷键的布局。选择 Web Development Settings，因为这些设置是专门为 ASP.NET 开发人员设计的。通常，可以通过重置设置来选择不同的配置文件，本章后面将会解释。

(3) 完全配置好 VS 后，就会出现主屏幕，如图 1-3 所示。

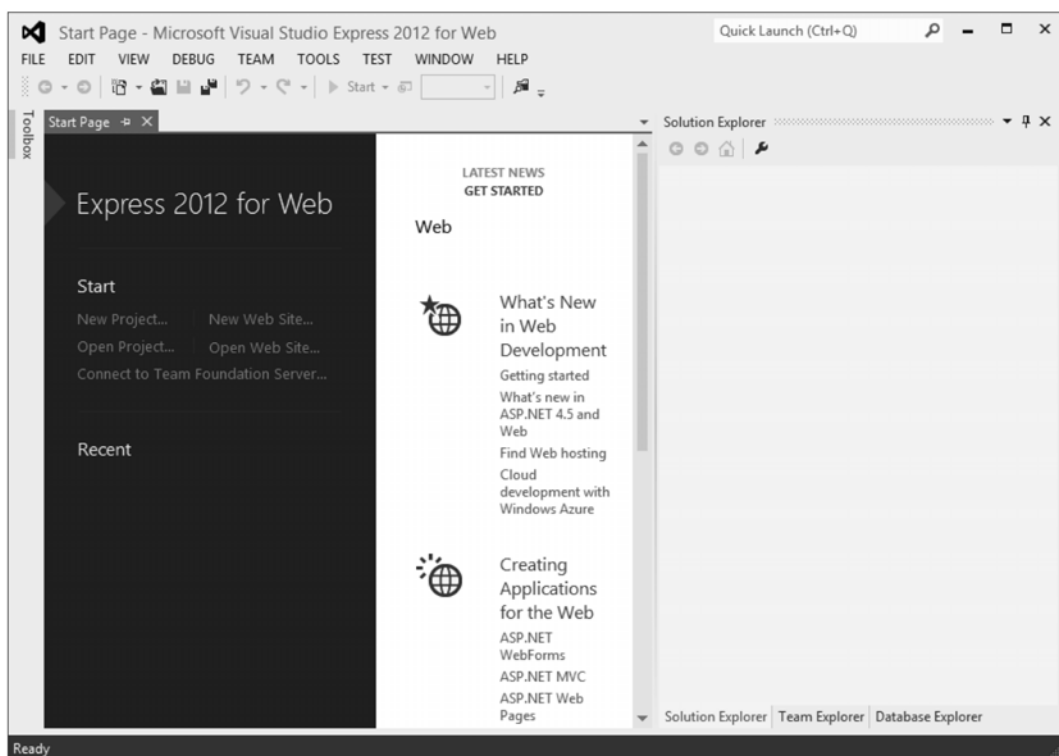


图 1-3



下一节将对所有的窗口、工具栏、面板和菜单进行描述。因此这里将重点放在创建新 Web 站点上。单击左上角的 File 菜单，并选择 New Web Site。如果使用的是 Visual Studio 的商业版本，那么根据首次启动 Visual Studio 时选择的设置，可能会先打开子菜单 New。

注意，不要选择 New Project 菜单项，因为它是用来创建另一种类型的 .NET 应用程序的。这时会出现 New Web Site 对话框，如图 1-4 所示。

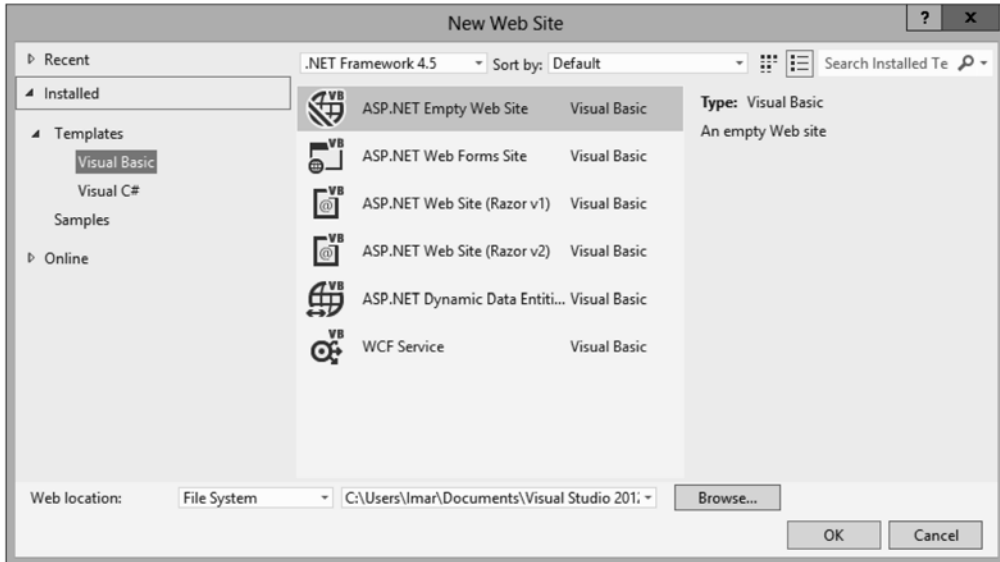


图 1-4

(4) 在左边的 Installed Templates 部分，选择一种供站点使用的编程语言。本书显示的所有示例都使用了 Visual Basic 和 Visual C#，因此可以根据自己的喜好选择一种语言。

(5) 在中间的模板列表框中，确认选择了 ASP.NET Web Forms Site。还要确认在左下方的 Web Location 下拉列表中选中 File System。如果愿意，可以改变 Web 站点在磁盘上的存储位置，单击 Browse 按钮，在计算机的硬盘驱动器上选择一个新位置即可。就目前而言，用默认位置即可——Documents 文件夹下的一个文件夹，因此可以让这个位置保持原样。

(6) 单击 OK 按钮。这时 VS 就创建了一个新的 Web 站点，其中包括许多如图 1-5 所示的文件和文件夹，它们可以用来开始创建 Web 站点。

(7) 双击文件 Default.aspx，删除<asp:Content>代码块中的所有代码(其 ID 设置为 BodyContent，起始标记是<h2>，大约在第 19 行，结束标记是</p>)，用下面突出显示的文本和代码进行替换：

```
<asp:Content runat="server" ID="BodyContent"
ContentPlaceHolderID="MainContent">
    <h2>Hello World</h2>
    <p>Welcome to Beginning ASP.NET 4.5 on <%: DateTime.Now.ToString() %></p>
```

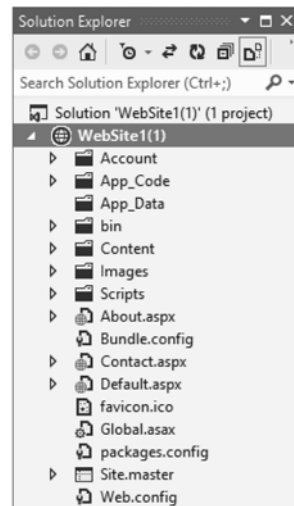


图 1-5

```
</asp:Content>
```

在本书中，你将会看到很多这种格式的代码。当书中要求输入这种粗体格式的代码时，只需要输入突出显示的代码即可。其余的代码应该已经存在于文件中了。

不用关心欢迎消息中使用尖括号(<>)的代码和百分号，它们称为标记，稍后自然就会知道它是如何工作的。虽然到目前为止，你可能还不熟悉这样的代码，但是不难猜出它是用来做什么的：输出今天的日期和时间。

(8) 按 Ctrl+F5 组合键在默认的 Web 浏览器中打开页面，如图 1-6 所示。

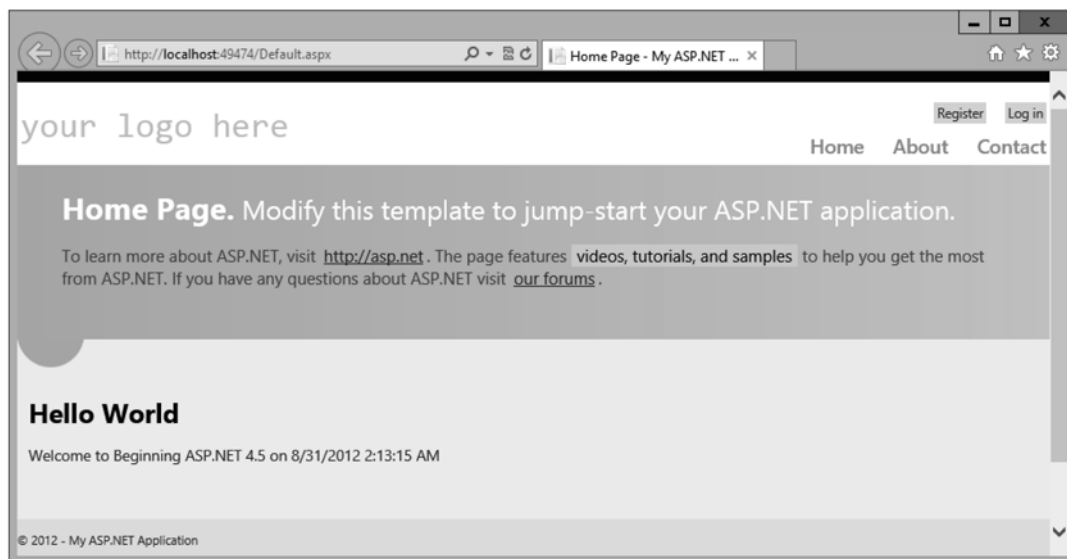


图 1-6

如果有一些 Visual Studio 经验，就可能习惯按下 F5。如果使用这个选项，站点就会在调试模式下打开，并显示一个对话框，询问是否要启用调试功能(可以安全地这么做)。用 F5 进行调试将在第 18 章讨论，在此之前，最好使用 Ctrl+F5 组合键。

如果在 Internet Explorer 中看到了关于 Intranet 设置的信息警告条，则单击 Turn on Intranet Settings。

如果在页面中没有看到日期和时间，或者收到了错误消息，则回顾一下欢迎消息中的代码。它以前尖括号(<)开头，后面跟着一个百分号和一个冒号，以一个百分号和另一个后尖括号(>)结束。还要确保输入和此处完全相同的代码，包括大小写也要一致。如果使用的语言是 C#，则这一点特别重要，因为该语言是区分大小写的。

(9) 注意，Windows 的任务栏中会出现一个表示 IIS Express 的小图标，如图 1-7 所示。



图 1-7

如果看到另一个图标，就在 VS 中右击站点，选择 Use IIS Express。如果没有在任务栏中看到该图标，则可以单击 Windows 任务栏中其他图标旁边的箭头，并单击 Customize 选项。然后把 IIS Express System Tray 设置为 Show Icon and Notifications。这个图标属于内置的 Web 服务器 IIS Express。该 Web 服务器由 VS 自动启动，以响应对页面的请求。本书后面将会介绍 Web 服务器如何处理页面。

这就是用 Visual Studio 创建第一个 ASP.NET Web 站点的过程。

### 工作原理

虽然在上面的“试一试”练习中创建的这个 Web 站点相当简单，但是让 Default.aspx 页面显示在浏览器中的过程却没有那么简单。ASP.NET 页面(根据它的扩展名，也称为 ASPX 页面或 Web 窗体)本身并不能做太多的事。在浏览器能够显示它之前，需要一个 Web 服务器对它进行处理。这就是 VS 自动启动 IIS Express 来处理页面请求的原因。接下来，它会启动默认的 Web 浏览器并定向到“试一试”练习中的 Web 服务器的地址：`http://localhost:49474/Default.aspx`，不过每次启动 Web 服务器时，这个地址中的实际端口号可能都不同，因为该数字是 VS 随机选择的。

重要的是要意识到，在 VS 中修改的 ASPX 文件并不是在浏览器中最终显示的文件。

在 VS 中创建一个页面时，就向它添加了标记(markup)。ASPX 页面中的标记由以下内容组成：HTML、ASP.NET 服务器控件的代码(在本章和第 4 章中将对它进行更多的介绍)、用 Visual Basic .NET 或 C#编写的代码等。

在浏览器中请求一个 ASPX 页面时，Web 服务器就会处理这个页面，执行它在文件中找到的所有服务器端代码，并有效地将 ASP.NET 标记转换为纯 HTML，然后发送给显示这个页面的浏览器。在上面的“试一试”练习中，最终的 HTML 会引起浏览器显示当前日期和时间。HTML(HyperText Markup Language, 超文本标记语言)是浏览器用来显示 Web 页面的语言。本章后面将会介绍 HTML，以及如何使用 HTML。

要查看最终的 HTML 与原始的 ASPX 页面有什么区别，请在浏览器中打开该页面的源代码。在大多数浏览器中，可以通过右击页面并选择 View Source 或 View Page Source 命令来打开源代码窗口。这样还会打开一个默认的文本编辑器，用于显示该页面的 HTML 代码。

如果在完成上面的“试一试”练习的操作后已经关闭了浏览器，则也可以在 VS 中按 Ctrl+F5 组合键，再次打开该页面，并选择 View Source 命令。

在源代码中向下滚动，找到包含欢迎消息的代码行，注意标记之间不再是代码，而是实际的日期和时间。

```
<h2>Hello World</h2>
<p>Welcome to Beginning ASP.NET 4.5 on 8/31/2012 2:13:15 AM</p>
```

当 Web 服务器处理页面时，它就从服务器中查询当前日期和时间，并把它们插入到要发送给浏览器的 HTML 中。根据 Windows 安装中的语言设置，会看到为适应 Windows 区域设置而进行格式化的不同日期和时间。

---

下一节将更详细地介绍 ASP.NET 的工作原理。

## 1.3 ASP.NET 4.5 简介

在 Web 浏览器中输入 `www.wrox.com` 这样的 URL 并按下 Enter 键时，浏览器就会向那个地址的服务器发送一个请求。这个过程是通过 HTTP(HyperText Transfer Protocol, 超文本传输协议)完成的。HTTP 是 Web 浏览器与 Web 服务器之间进行通信的协议。当发送地址时，就是向服务器发送了一个请求。当服务器处于活动状态且请求有效时，服务器就会接受并处理请求，然后将响应发送回客户

端浏览器。请求与响应之间的关系如图 1-8 所示。

由于使用了 IIS Express, 因此服务器和客户端是同一台机器。但是, 在现实情况下, 会将 Web 站点放到外部 Web 服务器上, 该服务器可以被许多不同的客户端访问。

对于简单的静态文件, 比如 HTML 文件或图像, Web 服务器只是简单地从它的本地硬盘驱动器中读取文件, 并发送给浏览器。然而, 对于动态文件, 比如 ASPX 页面, 这样做显然是不够的。假如 Web 服务器将 ASPX 文件直接作为文本文件发送给浏览器, 那么在浏览器中就看不到当前的日期与时间, 而只能看到实际的代码(<%: DateTime.Now.ToString() %>)。因此, Web 服务器不是直接发送这个文件, 而是将这个请求传递给能够处理该页面的另一个软件模块。这是通过所谓的 Application Mapping 或 Handler Mapping 概念来完成的, 其中文件的扩展名(在本例中是.aspx)表明了能够处理它的应用程序。就.aspx 页面而言, 请求最终由 ASP.NET 运行库(它是 Microsoft .NET Framework 的一部分, 专门为处理 Web 请求而设计)处理。

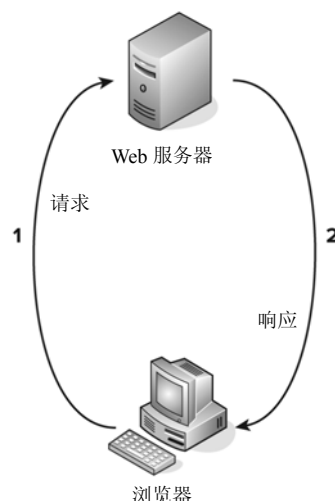


图 1-8

在页面的处理过程中, 有 3 个重要方面会影响页面最终出现在浏览器中的方式:

- **静态文本:** 任何静态文本(如 HTML、CSS 或者可以放在页面中的 JavaScript 代码)都是直接发送给浏览器的。本章和接下来的几章将介绍有关 HTML、CSS 和 JavaScript(一种在客户端使用的编程语言)的更多知识, 其中第 3 章将详细讨论 CSS。
- **ASP.NET 服务器控件:** 这些控件位于 ASPX 页面中, 在处理它们时, 它们会显示插在页面中的 HTML。在本章讨论了 HTML 之后, 将介绍有关服务器控件的更多知识, 第 4 章会集中讨论 ASP.NET 服务器控件。
- **编写代码:** 正如在“试一试”练习中显示的那样, 可以把代码直接嵌在页面中, 如 Visual Basic .NET 或 C#代码。此外, 可以将代码放在单独的代码文件(这个代码文件的官方名称是 Code Beside, 但大多数开发人员把它称为 Code Behind 文件, 本书使用这个术语)中。该代码可以由运行库自动执行, 或者基于用户的动作执行。无论采用哪种方式, 代码的执行都会大大影响页面的显示方式, 包括访问数据库、执行计算、隐藏或显示特定控件等。在第 5 章中将会介绍有关 Code Behind 文件的更多内容, 还会详细地讨论 ASP.NET Web 页面程序的编写。

处理了页面并收集了页面的所有 HTML 后, 就将其发送回浏览器。然后浏览器会读取该 HTML 并进行分析, 最终显示出该页面。

HTML 对于显示 Web 页面很关键, 下一节将概述 HTML。

### 1.3.1 HTML

HTML 是实际创建 Web 页面的语言, 如今现有的每个 Web 浏览器都能理解这种语言。自从 20 世纪 90 年代初以来, 它就成为了 World Wide Web 的驱动力量、Internet 处理 Web 页面的部分。HTML 文档是含有标记、文本和影响文本的附加数据的简单文本文件。HTML 的最新版本是 HTML5。尽管 HTML5 的规范仍在开发, 但大量现代浏览器都支持这个规范的重要部分, 这些浏览器的每个新

版本都在提升这种支持。尽管并不是所有的浏览器都完全支持 HTML5，但它实际上代表了基于 HTML 的规范的未来，因此本书使用它，并用于 Planet Wrox 演示网站。不必过于担心有限的浏览器支持。所有主流浏览器都支持本书使用的所有 HTML5 功能，或者通过脚本库 Modernizr 很容易模拟该支持，如本章后面所述。

1. HTML 元素和标记

HTML 用尖括号间的文本指示内容在浏览器中如何显示。这种带有尖括号的文本称为标记(tag)；含有文本或其他内容的一对标记称为元素。在上面的“试一试”练习中，在浏览器中打开页面的源代码窗口，所看到的 HTML 如下所示：

```
<h2>Hello World</h2>
<p>Welcome to Beginning ASP.NET 4.5 on 8/31/2012 2:13:15 AM</p>
```

该示例的第一行含有一个带起始标记<h2>和结束标记</h2>的<h2>元素。此元素用来表示二级标题(如果在浏览器最终的源代码中向上滚动，就会看到<h1>元素)。注意，元素的结束标记和起始标记相似，只是前面多了个斜杠(/)：</h2>。这些起始标记和结束标记之间的所有文本都被看做是标题部分，因此呈现为标题。在大多数浏览器中，这意味着标题会呈现为较大的字体。与<h2>标记类似，还有创建一直到 6 级标题的标记，比如<h1>、<h3>等。

在标题元素下面，可以看到一个<p>元素，它用来表示段落。<p>标记对中的所有文本都被看做是段落部分。默认情况下，浏览器呈现段落时会在下方留出一些空白，不过也可以忽略这个行为。

HTML 中有许多可用的标记，不可能在这里一一介绍。表 1-1 列出了一些最重要的标记，并说明了它们的用法。要了解关于 HTML 元素的完整列表，参见维护 HTML 标准的组织的站点：[www.w3.org/TR/html5/index.html](http://www.w3.org/TR/html5/index.html)。

表 1-1

标 记	说 明	示 例
<html>	用来表示整个页面的开始和结束	<html> ...All other content goes here </html>
<head>	用来表示包含页面数据的页面的特殊部分，包括其标题以及对外部资源的引用。这个元素的内容不直接输出到屏幕上，但会影响页面的外观，如后面所述。这个元素放在<html>元素中	<head> ... Content goes here </head>
<title>	用来定义页面的标题。这个标题将会显示在浏览器的标题栏中。这个元素放在< head>元素中	<title> ... Welcome to Planet Wrox 4.5 </title>
<body>	用来表示页面体的开始和结束，其内容就是在浏览器中显示的内容，这个元素放在< html>元素中	<body> ... Page body goes here </body>
<header>	用来表示页面的标题。这个元素和这个表中的所有剩余元素都放在<body>元素中	<header> ... </header>

(续表)

标 记	说 明	示 例
<section>	用来表示页面中的各个部分。每个页面中可以有多个部分	<pre>&lt;section&gt; ...Content goes here &lt;/section&gt;</pre>
<aside>	用来表示不属于核心页面的内容, 这些内容显示为旁白	<pre>&lt;aside&gt; ...&lt;img src="Banner.png" .../&gt; &lt;/aside&gt;</pre>
<article>	用来表示页面中的主要内容	<pre>&lt;article&gt; ...Main page content goes here &lt;/article&gt;</pre>
<footer>	用来表示页面的脚注部分	<pre>&lt;footer&gt; ...Content for footer goes here &lt;/footer&gt;</pre>
<a>	用来将一个 Web 页面链接到另一个页面, 或者创建页面中的链接	<pre>&lt;a href="http://www.wrox.com"&gt;   Visit the Wrox site &lt;/a&gt;</pre>
<img>	用来向页面中嵌入图像	<pre>&lt;img src="Logo.gif" /&gt;</pre>
<strong> <em>	用来将文本格式化为粗体或斜体	<pre>This is &lt;strong&gt;bold text&lt;/strong&gt; while &lt;em&gt;this text is in italic&lt;/em&gt;</pre>
<form> <input> <textarea> <select>	用于描述允许用户向服务器提交信息的输入格式	<pre>&lt;input type="text" value="Some Text" /&gt;</pre>
<table> <tr> <th> <td>	这些标记用来创建含有表格的布局。<table>标记定义了整个表, <tr>用于表示标题单元格, 而<tr>和<td>标记分别用来定义行和单元格	<pre>&lt;table&gt; &lt;tr&gt;   &lt;td&gt;This is a Cell in Column 1&lt;/td&gt;   &lt;td&gt;This is a Cell in Column 2&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>
<ul> <ol> <li>	这 3 个标记用来创建带有编号或项目符号的列表。<ul>和<ol>标记定义了列表的外观(可能是无序的, 带一个简单的项目符号; 也可能是有序的, 带有编号), 而<li>标记用来表示列表中的项	<pre>&lt;ul&gt;   &lt;li&gt;First item with a bullet&lt;/li&gt;   &lt;li&gt;Second item with a bullet&lt;/li&gt; &lt;/ul&gt;  &lt;ol&gt;   &lt;li&gt;First item with a number&lt;/li&gt;   &lt;li&gt;Second item with a number&lt;/li&gt; &lt;/ol&gt;</pre>

(续表)

标 记	说 明	示 例
<span>	这个标记用来包装和影响文档的其他部分。它作为内联文本出现，所以不会向页面添加换行符	<p>This is some normal text while <span style="color: red;">this text appears in red </span></p>
<div>	与<span>标记一样，<div>标记用来作为其他元素的容器。然而，<div>标记是块级元素，默认情况下它会使<div>元素的前后出现显式的换行符	<div> This is some text on 1 line </div> <div> This text is put directly under the previous text on a new line. </div>
<audio> <video> <source>	用于在 Web 页面中嵌入音频和视频文件。<source>元素用于定义多种类型的音频和视频资源	<video src="Somevideo.mpg" />

2. HTML 特性

表 1-1 中除了有 HTML 元素外，还显示了 HTML 特性。这些特性包含了一些改变特定元素行为方式的额外信息。例如，使用<img>标记显示一个图像，src 特性定义图像的源代码。类似地，<span>标记含有一个将文本改为红色的 style 特性。style 特性的值(color: red; )是层叠样式表(Cascading Style Sheet, CSS)的一部分，这个概念将在第 3 章详细介绍。与 HTML 元素一样，在 W3C 的 Web 站点上有一个很长的可用特性列表：[www.w3.org/TR/html5/index.html#attributes-1](http://www.w3.org/TR/html5/index.html#attributes-1)。

不需要记住所有这些元素和特性。在大多数情况下，VS 会自动地生成它们。而在其他情况下，当需要手工输入它们时，VS 会提供 IntelliSense 工具，帮助找到正确的标记或特性。第 2 章将介绍 IntelliSense 工具。

3. HTML 注释

为了在 HTML 中注释一些内容，可以把它们放在注释标记中，如下所示：

<!-- This is a comment -->

浏览器不会处理添加了注释符号的代码(因此是不可见的)，但这样的代码仍会发送给浏览器(因此最终用户可以看到它们)。因为这些代码仍会发送给浏览器，这会增加页面的尺寸，所以应尽量少地使用注释。后面的章节将说明如何在服务器上注释代码，使这些代码不发送到客户端。

4. HTML5 的规则

HTML5 的规则相当简单，而且大多数时候 VS 都会帮助你作出正确决定，或者提供错误列表和关于如何修复的建议。在执行规则时，HTML5 实际上比 HTML 以前的版本(称为 XHTML，XHTML

是用 XML 规则重新表示的 HTML 4.01)更简单。

### 1) 闭合元素

HTML 中的大多数元素都必须闭合。所以开始一个<div>标记后,就必须在页面后面的某个地方使用</div>来闭合它。也会有一些例外(例如,后面紧跟其他一些元素的<p>元素),但本书总是会闭合标记。对于没有结束标记的元素也是如此,比如<img>或<br>(用来输入一个换行符)。在 HTML5 中,这些标记被写为自结束标记,其中结束标记中的斜杠直接嵌在标记自身中,比如在或<br />中。

### 2) 特性的用法

每当在标记中写特性时,都可以用双引号、单引号将它的值括起来,或者完全不使用引号。例如,在写<img>标记和它的 src 特性时,应该这样写:

```

```

也可以使用单引号将特性值括起来,如下所示:

```
<img src='Logo.gif' />
```

这两个选项都有效,只是在值的两端要使用相同类型的引号。对于不包含空格的值,也可以不使用引号:

```
<input value=yes>
```

有时还有必要嵌套单引号和双引号。当有些特殊 ASP.NET 语法要求使用双引号时,应当用单引号括起特性的值:

```
<asp:Label ID="TitleLabel" runat="server" Text='<%# Eval("Title") %>' />
```

在本书的其他章节中将不时地出现这种语法。

为了保持一致,本书在最终出现在客户端的所有 HTML 中可能用到引号的地方都使用双引号,因为这是被广泛接受的标准。

### 3) 正确地嵌套元素

在写嵌套元素时,确保首先闭合最后打开的内部元素,然后闭合外部元素。考虑这个示例,即同时用粗体和斜体格式化一段文本:

```
<strong><em>This is some formatted text</em></strong>
```

注意<em>标记是在<strong>标记之前闭合的。交换结束标记的顺序会导致无效的 XHTML:

```
<strong><em>This is some formatted text</strong></em>
```

### 4) 向页面中添加一个 DOCTYPE 声明

DOCTYPE 给出了期望的关于 HTML 类型的浏览器信息。默认情况下,VS 向页面中添加 HTML5 的 DOCTYPE。

```
<!DOCTYPE html>
```

DOCTYPE 大大地影响了浏览器(如 Internet Explorer)呈现页面的方式。所以,如果在页面上看到了古怪的行为,应检查页面是否有正确的 DOCTYPE。



在 W3C 站点 [www.w3.org/TR/html-markup/syntax.html](http://www.w3.org/TR/html-markup/syntax.html) 上可以看到完整的 HTML5 语法规则。

除了 HTML 之外, ASP.NET Web 页面也可能包含其他标记。大多数页面上都有一个或多个 ASP.NET 服务器控件, 给出了一些附加的功能。下一节将简要介绍这些 ASP.NET 服务器控件, 第 4 章再深入介绍它们。

### 1.3.2 初识 ASP.NET 标记

在某种程度上, ASP.NET 服务器控件的标记与 HTML 的标记很相似。它也像 HTML 一样用尖括号和结束标记表示标记、元素和特性。然而, 一个主要区别是, ASP.NET 标记以 `asp:` 前缀开头。例如, ASP.NET 中的按钮如下所示:

```
<asp:Button ID="Button1" runat="server" Text="Click Me" />
```

注意此标记是用斜线字符(`/`)自闭合的, 所以不必另外输入结束标记。但如果需要输入结束标记, 可以使用单独的结束标记。

处理服务器控件时, 它会返回 HTML。所以, 在浏览器上显示时, 上述按钮的代码如下所示:

```
<input type="submit" name="Button1" value="Click Me" id="Button1" />
```

把服务器控件转换为 HTML 表达方式的过程类似于前面显示当前日期的代码。服务器控件由 ASP.NET 处理程序在服务器中进行处理。结果产生的 HTML 被发送到浏览器, 并在浏览器中显示, 详见第 4 章。

我们已经介绍了 ASP.NET 页面和它生成的 HTML 的基础知识, 接下来介绍 VS。学会使用这个应用程序及其众多的工具和窗口, 是构建有趣、漂亮且实用的 Web 站点的重要一步。

## 1.4 IDE

Visual Studio 是目前为止构建 ASP.NET Web 页面使用最为广泛、功能最为丰富的集成开发环境 (IDE)。缩略词 IDE 是指构建复杂 Web 应用程序所需的所有独立工具都集成在一个环境中。VS 不需要在文本编辑器中编写代码、在命令行中编译代码、在单独的应用程序中编写 HTML 和 CSS 以及在另一个应用程序中管理数据库; 它允许在同一个环境中执行所有这些任务及更多其他的任务。由于不必一直进行工具转换, 因此这样一来效率得到了提高, 并且因为许多内置工具的工作方式都是一样的, 所以人们能够更容易地学习 VS 的新功能。

### 1.4.1 主开发区

为了熟悉 VS 的界面中含有的众多工具, 可以参看图 1-9。它显示了用 VS 创建第一个 Web 站点后所看到的屏幕, 不过现在它突出显示了部分最重要的屏幕元素。如果已经熟悉了 Visual Studio 之前的版本, 就可以跳过该部分, 并在本章后面的“试一试”练习中学习有关知识。

如果安装的是 Visual Studio 以前的版本, 屏幕可能会与图 1-9 有所不同, 因为 Visual Studio 2012 能够导入老版本的设置。

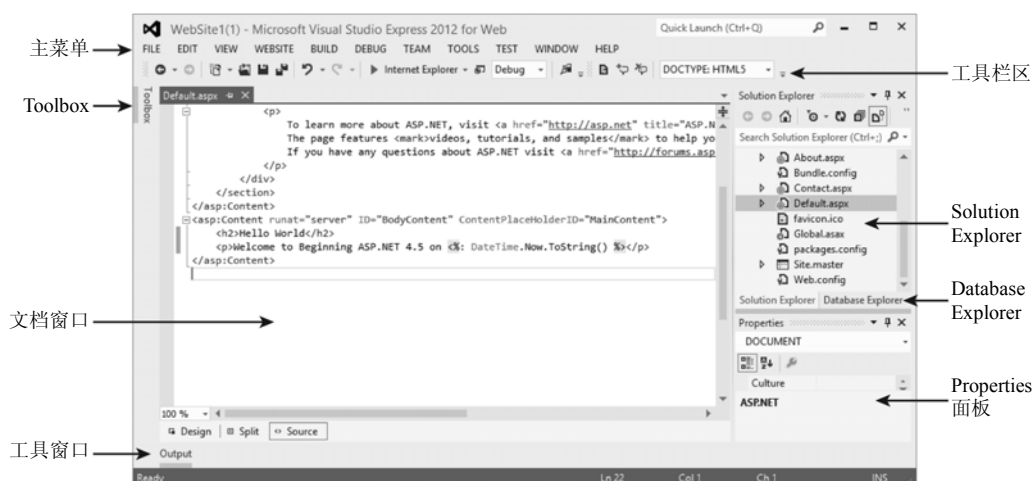


图 1-9

### 1. 选择开发配置文件

VSEW 面向 ASP.NET 开发新手以及经验丰富的 Web 开发人员，因此它提供了 3 种不同的开发配置文件供开发人员选择：基本设置(Basic Settings)、只有代码(Code Only)以及专家设置(Expert Settings)。在基本设置模式下，许多不常使用的菜单项会被隐藏或放在自己的子菜单中。只有代码模式对于纯编码会话来说很有用处，而 VSEW 的这些设计特色(例如设计视图或 Toolbox)往往是你不感兴趣的。而专家设置模式允许访问 VSEW 的全部功能。可以使用 Tools | Settings 菜单来转换不同的设置。在本书中，假设使用专家设置模式。本书开头所介绍的全部功能可能不会都用到，但是到最后肯定会用到大多数的功能。由于菜单项可以根据选择的模式改变其位置，因此需要使用专家设置模式，以便更简单地使用特定的菜单项或功能。在 Visual Studio 的商用版本中没有这个选项——专家设置模式默认为打开。

### 2. 主菜单

在这个应用程序的上方，Windows 标题栏的下面，可以看到主菜单。此菜单栏中包含的菜单项你一定很熟悉，因为在很多其他的 Windows 应用程序中都可以找到这些菜单项，如 File、Edit、Help 菜单，以及一些 VS 中特有的菜单，如 Website 和 Debug 菜单。该菜单可以根据执行的具体任务动态地改变，因此在使用应用程序的过程中你会发现某些菜单项有时出现、有时消失。可以通过使用 Help | Set Help Preference 菜单进行在线配置和离线帮助。离线帮助需要首先安装该应用程序，而在线帮助则需要连接到 Internet。

### 3. 工具栏区

在菜单的下方，可以看到工具栏区，在该区域显示了不同的工具栏，从而可以快速地访问 VS 中的大部分常见功能。在图 1-9 中，只启用了两个工具栏，但是在面向特定任务的场景中，VS 会随之打开很多可以使用的其他工具栏。有些工具栏会在执行需要特定工具栏出现的任务时自动地出现，但是也可以根据自己的喜好启用或禁用工具栏。要启用或禁用工具栏，只需要右击现有工具栏或菜单栏，从出现的菜单中选择工具栏即可。

#### 4. Toolbox

在主屏幕的左边, 可以看到折叠在 VS 边缘的 Toolbox 选项卡。如果把鼠标指针悬停在该选项卡上, Toolbox 就会展开, 这样就能看到它包含的内容。如果单击 Toolbox(或者有小钉图标的其他面板)右上角的小钉图标, 它就会锁定在 IDE 上, 保持打开状态。

与菜单栏和工具栏一样, Toolbox 会自动更新, 以显示与正在执行的任务相关的内容。在编辑标准 ASPX 页面时, Toolbox 会显示可用于页面的许多控件。可以简单地从 Toolbox 中拖动一个控件, 然后把它放到希望在页面中出现的位置上。这些控件将在第 4 章详细讨论。注意每个 Toolbox 类别还包含一个 Pointer 图标, 但它不是一个控件。在 Visual Studio 的其他设计器(例如 Win 窗体)中, 这个图标用于退出控件绘制模式, 但它在 ASP.NET 中没有什么用。Toolbox 包含多个类别, 其中的工具可以根据意愿展开和折叠, 以便找到正确的工具。也可以重新排列列表中工具的顺序, 从 Toolbox 中添加和删除工具, 甚至可以向其中添加自己的工具。关于定制 IDE 将在本章后面讨论。

如果在屏幕上看不到 Toolbox, 则可以按 Ctrl+Alt+X 组合键打开它, 或者从 View 菜单中选择 Toolbox 命令来打开(假设已经选择了 Tools | Settings 菜单中的专家设置选项)。

#### 5. Solution Explorer

在屏幕右边, 可以看到 Solution Explorer。Solution Explorer 是一个重要窗口, 因为它提供了组成 Web 站点的文件概览。Solution Explorer 没有把所有文件都放在一个大文件夹中, 而是将文件存储在单独的文件夹中, 创建了一个有逻辑且有组织的站点结构。可以用 Solution Explorer 向站点中添加新文件, 使用拖放功能或者剪切粘贴功能来移动现有文件, 从项目中重命名文件以及删除文件等。Solution Explorer 的大部分功能都隐藏在它的右击菜单中, 该菜单根据在浏览器窗口中右击的项目而改变。

在 Solution Explorer 的上方有一个小工具栏, 可以用来快速访问与 Web 站点相关的一些功能: 包括刷新 Solution Explorer 窗口, 嵌套相关文件的选项, 以及用来复制和配置 Web 站点的两个按钮。大多数这些功能在本书后面都会讨论到。

可以通过从主菜单中选择 View | Solution Explorer 或者按 Ctrl+Alt+L 组合键来访问 Solution Explorer。

#### 6. Database Explorer

这个窗口隐藏在图 1-9 所示的 Solution Explorer 后面, 通过它可以操作数据库。如果使用的是 Visual Studio 的商业版本, 如 Visual Studio 2012 Professional, 那么这个窗口就称为 Server Explorer, 它可能位于屏幕左侧。

要访问 Database Explorer, 可以选择 View | Database Explorer, 或者按下 Ctrl+Alt+S。Database Explorer 将在关于数据库的那几章(从第 12 章开始)详细讨论。

#### 7. Properties 面板

用 Properties 面板可以查看和编辑 Visual Studio 中的许多项目的属性, 包括 Solution Explorer 中的文件、Web 页面上的控件、页面本身的属性及其他更多内容。这个窗口会不断地更新, 以反映选中的项。按 F4 键可以快速打开 Properties 面板。这个快捷键还可以用来强制 Properties 面板显示选

中项的详细信息。

## 8. 文档窗口

应用程序中间的文档窗口是主要区域。大部分动作都在这里发生。可以用文档窗口来操作很多不同的文档格式,包括 ASPX 和 HTML 文件、CSS 和 JavaScript 文件、VB 和 C#的代码文件、XML 和文本文件,甚至图像文件。此外,用这个窗口还可以管理数据库、创建站点的副本,并在内置的微型浏览器中浏览页面等。

默认情况下,文档窗口是一个带选项卡的窗口,这意味着它能驻留多个文档,各个文档通过选项卡用窗口上方显示的文件名进行区分。各选项卡的右击菜单中包含使用该文件的一些有用的快捷键,包括保存与关闭文件,以及在 Windows Explorer 中打开该文件的父文件夹。

要在文档之间进行切换,可以按 Ctrl+Tab 组合键,或者单击要查看的文档的标签,或者单击文档窗口右上角的下拉箭头,该文档窗口邻近 Solution Explorer,如图 1-9 所示。单击下拉箭头会显示出一个打开文档的列表,因此可以轻而易举地从中选择要打开的文档。

切换文档的另一种方式是按下 Ctrl+Tab 组合键,然后按住 Ctrl 键。在弹出的窗口中,可以在右手边的那一栏中选择要使用的文档。然后可以在打开文档的列表中向上或向下移动光标。这样选择正确的文件就变得相当容易。

在同一个对话框中,可以看到一个包含所有活动工具窗口的列表。单击这个列表中的一个窗口,可以将它显示在屏幕上;如有必要,还能将它移到其他窗口的前面。

为了快速预览文档,而无须打开它,进行编辑,可以在 Solution Explorer 中单击要查看的文件,该文件会在其选项卡中处于预览模式,该选项卡停靠在右边一行,而不是左边放置打开文件的行上。

在图 1-9 所示的文档窗口下方可以看到 3 个按钮,分别是 Design、Split 和 Source。在操作含有标记的文件(如 ASPX 和 HTML 页面)时,这些按钮会自动出现。它允许打开页面的 Design View(让你对页面在浏览器中的样子有个概念)、它的 Markup View(HTML 和其他标记),或者同时打开这两者。其工作原理将在第 2 章详细解释。现在重要的是要知道,可以通过单击相应的按钮在 Markup 视图、Split 视图和 Design 视图之间切换。Markup 视图也常称为 Source 视图或 Code 视图窗口。然而,为了避免与用于编辑 Code Behind 文件的代码编辑器混淆,本书只采用 Markup 视图这个术语。

## 9. Start Page

每次启动 VS 时,Start Page 都要加载到文档窗口中。有了 Start Page,就可以快速地创建新的 Web 站点或者打开现有站点和其他项目。Start Page 还提供了一些有关 Web 开发的新闻和信息的链接。为了再次打开 Start Page,可以选择 View | Start Page。

为了对如何使用所有这些窗口有一个感性的认识,下面的“试一试”练习将会介绍如何构建一个包含一些 ASP.NET 服务器控件的简单 Web 页面。

### 试一试

### 创建第一个 ASP.NET Web 页面

这个“试一试”练习将创建一个只有一个页面的新 Web 站点,该页面包含若干个 ASP.NET 服务器控件。该练习将介绍如何使用文档窗口和 Solution Explorer,以及如何使用 Toolbox 和 Properties 面板向页面中添加 ASP.NET 服务器控件并修改它们的外观。

- (1) 启动 VSEW 或 Visual Studio 2012。
- (2) 如果使用的是 Express 版本, 则选择 Tools | Settings 并打开专家设置中的开发人员配置, 以便访问 VSEW 的全部功能集。
- (3) 在 File 菜单中选择 New Web Site。根据不同的配置, 可能需要选择 File | New | Web Site。这样会打开 New Web Site 对话框。
- (4) 在该对话框中, 确保选中的是 ASP.NET Empty Web Site, 而不是先前练习中使用的 ASP.NET Web Forms Site。一定要从 Web Location 的下拉列表中选择 File System。然后单击 OK 创建新站点。
- (5) 接着, 右击 Solution Explorer 中新的 Web 站点。确保单击了网站 WebSite2, 而不是父 Solution 元素。图 1-5 中突出显示了该元素。从显示的上下文菜单中选择 Add | Add New Item 命令。
- (6) 在出现的新窗口中, 单击 Web 窗体并输入 ControlsDemo 作为名称。当单击 Add 按钮时, 它会自动添加扩展名 ASPX。可以允许对话框中的其他设置保留其默认值。该页面应该能在 Markup 视图中打开, 显示默认的 HTML, 如<html>、<head>、<title>和<body>元素, 这些元素是在创建新页面时由 VS 自动添加的。
- (7) 单击文档窗口下方的 Design 按钮, 将页面切换到 Design 视图。
- (8) 如果 Toolbox 还没有打开, 则按 Ctrl+Alt+X 组合键打开它, 或者将鼠标悬停在 Toolbox 选项卡上来显示它, 然后单击锁定图标使 Toolbox 一直可见。从 Toolbox 的 Standard 类别中将一个 TextBox 和一个 Button 拖放到页面 Design 视图中的阴影区域内。最后应当看到一个如图 1-10 所示的页面。
- (9) 右击 Design 视图中的 Button 按钮并选择 Properties 命令。在 Properties 面板中, 定位到 Appearance 类别下面的 Text 属性(如图 1-11 所示), 并将它从 Button 改为 Submit Information。一旦按下了 Tab 键或者在 Properties 面板之外的某处单击了鼠标, 页面的 Design 视图就会更新, 并在按钮上显示新文本。

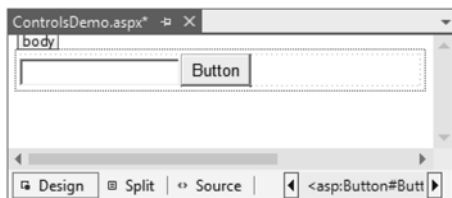


图 1-10

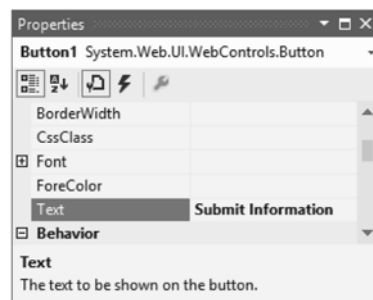


图 1-11

- (10) 按 Ctrl+F5 组合键在默认浏览器中打开该页面。注意, 没有必要显式地保存对页面的修改(不过用快捷键 Ctrl+S 经常保存会有好处的)。按 Ctrl+F5 组合键运行页面后, VS 就会自动保存对打开文档的所有修改。
- (11) 在文本框中输入一些文本, 然后单击这个按钮。注意, 当重新加载页面后, 文本仍然会显示在文本框中。如果没有显示, 则很可能是因为还没有为这个按钮编写任何代码。

### 工作原理

当从 Toolbox 中拖动 Button 和 TextBox 到页面的 Design 视图中时, VS 会自动在 Markup 视图中添加相应的代码。类似地, 当在 Properties 面板中修改按钮的 Text 属性时, VS 会自动更新 Markup

视图中控件的标记。如果不使用 Properties 面板,也可以在 Markup 视图中 Text 属性的引号之间直接输入文本。

修改了 Text 属性之后,页面在 Markup 视图中应包含如下代码:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Submit Information" />
```

当按 Ctrl+F5 组合键查看浏览器中的页面时,Web 服务器会收到请求,页面则由 ASP.NET 运行库处理,为页面产生的最终的 HTML 将被发送到浏览器。

当输入一些文本并单击按钮时,就会重复同样的过程:Web 服务器接收请求,处理页面,将结果发送回浏览器。当单击该按钮时,就引发了一个回发(postback),此时页面中的所有信息(如在文本框中输入的文本)都会被发送回服务器。ASP.NET 通过再次呈现页面来对回发作出响应。然而,这次它用发送到页面的值预先填充了控件,如 TextBox。

使用浏览器的 View Source 命令查看页面最终的 HTML 代码(如果已经关闭了,则按 Ctrl+F5 组合键从 VS 中重新运行页面)。代码应如下所示:

```
<input name="TextBox1" type="text" value="Hello World" id="TextBox1" />
<input type="submit" name="Button1" value="Submit Information" id="Button1" />
```

从前面的示例中可以看出,最终的 HTML 与原始 ASPX 标记有相当大的区别。

回发是 ASP.NET 中的一个重要概念,在包括第 4 章在内的其他章节中将更详细地介绍它们。

在 VSEW 中驻留的窗口和工具面板远远不止目前为止所提到的这些。下一节将简要介绍一些构建 ASP.NET Web 页面时最常用的窗口。如果使用的是专家设置模式,则这里提到的所有窗口都可以从 VS 或 VSEW 的主 View 菜单中访问。

## 1.4.2 信息窗口

除了在启动 VS 时见到的默认窗口外,VS 中还有很多可用窗口。在本书的其余部分,将运用到这些窗口,现在要先介绍几个重要的窗口。你可以访问接下来讨论的主 View 菜单中的所有窗口。

### 1. Error List

Error List 提供了一个列表,列出了当前因为某种原因在站点中被中断的内容,包括 ASPX 或 HTML 文件中的错误标记,以及 VB 或 C# 文件中的编程错误。这个窗口甚至可以显示 XML 和 CSS 文件中的错误。这个错误列表显示了 3 类消息——Errors、Warnings 和 Messages,它们分别表示不同的问题严重程度。图 1-12 显示了 CSS 和 XHTML 有问题的页面的错误列表。

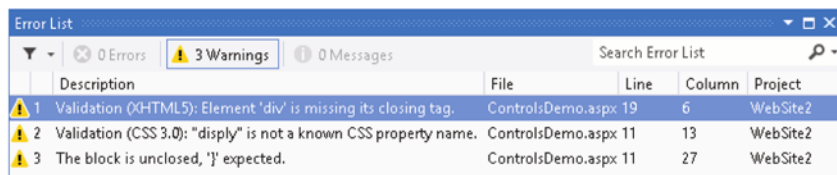


图 1-12

## 2. Output 窗口

当使用 Build 菜单构建站点时，Output 窗口会提示是否构建成功。如果构建失败，如存在编程错误，那么 Output 窗口就会指出构建失败的原因。在 Visual Studio 的商业版本中，Output 窗口还用来输出其他信息，包括外部插件程序的状态。构建或编译 Web 站点将在本书的第 19 章中讨论，该章主要介绍 Web 站点的部署。

## 3. Find Results 窗口

当开始管理站点的内容时，VS 的 Find 和 Replace 功能是非常有用的工具。在工作中经常需要替换当前文档甚至整个站点中的某些文本。Find in Files(Ctrl+Shift+F)和 Replace in Files(Ctrl+ Shift+H)都会在 Find Results 窗口中输出它们的结果，如图 1-13 所示。

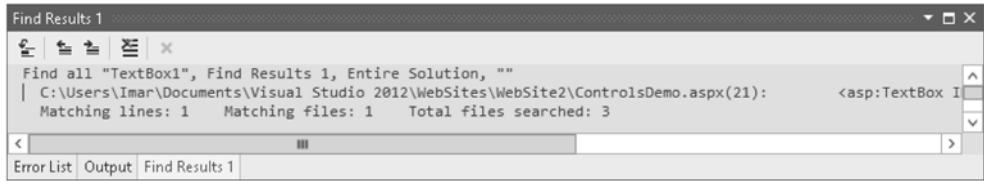


图 1-13

因为同时打开多个信息窗口可能会占用宝贵的屏幕空间，所以最好停靠(dock)它们。这样，一次就只能看到一个窗口，且仍然能够快速访问其他窗口。下一节将解释如何定制 IDE，包括如何停靠窗口。

# 1.5 定制 IDE

虽然 Visual Studio 的标准配置和它的工具窗口相当有用，但是也可以根据自己的偏好定制 IDE。有时可能需要把窗口重新安排到容易找到的位置，有时可能希望打开其他经常用到的窗口。Visual Studio 是可以完全自定义的，而且 IDE 的任何细节都可以进行调整。下一节将介绍如何完成大多数常见的自定义任务。

## 1.5.1 重新排列窗口

可以在主 IDE 中通过拖放重新排列窗口。只需简单地按住窗口的标题栏或者它下面的选项卡，并沿着新位置的方向拖动即可。开始拖动窗口时，会看到 Visual Studio 提供的窗口将停在何处的可视化线索(如图 1-14 所示)。

如果按住指示器边上的 4 个方块指示器之一拖动窗口，窗口将停靠在现有窗口旁边。当放下它时，窗口就会出现在它的新位置上。如果将窗口放在大指示器中间的方块上，窗口就会停靠在那个窗口的位置，并与之共享相同的屏幕空间。每个窗口都有自己的选项卡，这一点从图 1-14 下方的窗口上可以看出来。

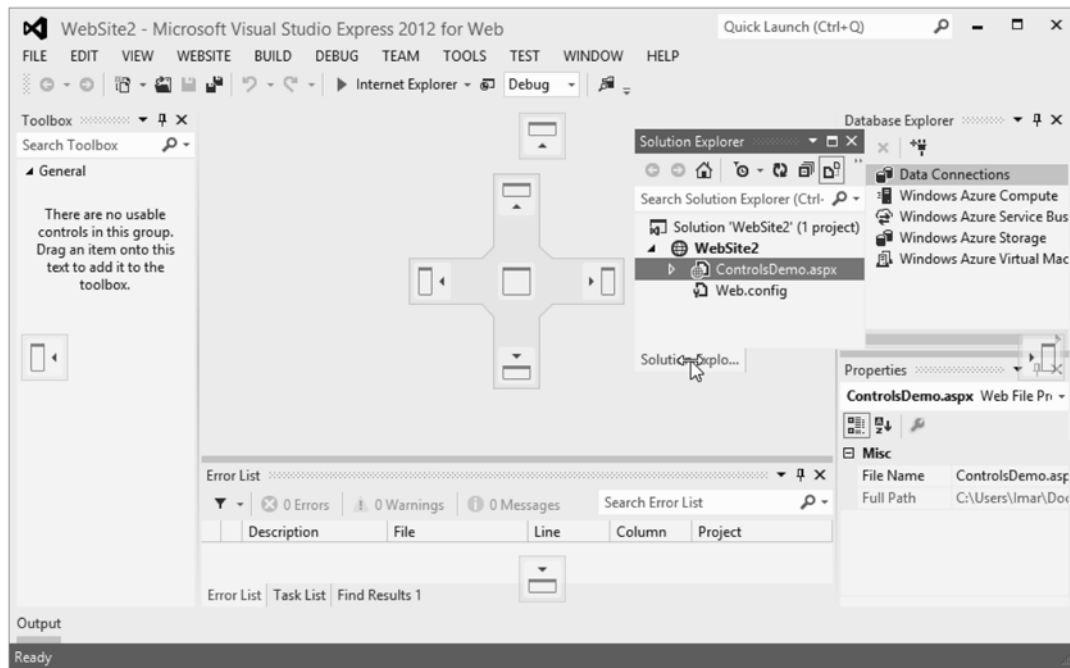


图 1-14

除了在 IDE 中将窗口与其他窗口停靠在一起外，还可以使之成为浮动窗口。要把停靠窗口改为浮动窗口，可以将窗口从它的当前位置拖走，并放在 IDE 中的任何位置，不需要单击屏幕中的某个可视化线索。也可以从主菜单中选择 **Window | Float** 命令，或者右击窗口的标题栏，选择 **Float**，使其浮动。

要将浮动面板重新恢复到它原来的停靠位置，可以右击其标题栏，并选择 **Dock**，或者从主菜单中选择 **Window | Dock** 命令即可。请确保不是选择 **Dock as Tabbed Document** 作为工具窗口(如工具箱或 **Solution Explorer**)，否则将以主文档窗口结束。由于两个窗口将共享相同的空间，因此很难使这些工具窗口与打开的文件一起使用。

### 1.5.2 修改 Toolbox

**Toolbox** 也是可以修改的。默认情况下，**Toolbox** 按照字母顺序对控件进行分类，但是也可以使用拖放工具对它们重新进行排列。为此，需要打开 **Toolbox**(按 **Ctrl+Alt+X** 组合键)，把一个控件(如 **Standard** 类别下的 **TextBox**)拖到一个不同的位置上。也可以通过右击它们并从上下文菜单中选择 **Delete** 命令，把它们从 **Toolbox** 中删除。不要担心控件会永久失去，因为可以从同样的菜单中选择 **Reset Toolbox** 命令来重置 **Toolbox**。

还可以向 **Toolbox** 中添加自己的控件。这个功能最常用于代码段。只需要简单地突出显示文档窗口中的一些文本或代码，并把它拖到 **Toolbox** 中。然后可以右击该项目并选择 **Rename Item** 命令给它起一个更有意义的名称，以方便识别。

为了避免 **Toolbox** 与自己的代码段混淆，可以为它们创建一个单独的类别。为此，可以从 **Toolbox** 的右击菜单中选择 **Add Tab** 命令。输入一个名称，然后按下 **Enter** 键，**Toolbox** 选项卡就可以使用了。



在下面的“试一试”练习中，可以修改 Visual Studio IDE，将它定制为自己喜欢的样子。

## 试一试

## 定制 IDE

这个练习是为了实践打开和重新排列 VS IDE 中的窗口。不用害怕使 IDE 陷入混乱。本章稍后将介绍如何将 IDE 重置为第一次打开时的样子。

(1) 如果关闭了在上一个“试一试”练习中建立的 Web 站点，就再打开它，或者使用 File 菜单新建一个站点。

(2) 从 View 菜单中，选择 Error List 命令，打开 Error List 窗口。如果没有看到 Error List 选项，则首先选择 Tools | Settings | Expert Settings。注意默认情况下，它停靠在文档窗口的下方。

(3) 还是从 View 菜单中，选择 Task List 命令。默认情况下，它会停靠在与 Error List 相同的空间中，两个窗口的选项卡彼此相邻。

(4) 单击 Task List 的选项卡，在按住鼠标按钮的同时，沿着文档窗口的方向将 Task List 拖离它的位置。释放后，它就会显示为 IDE 中的浮动窗口。要恢复该窗口，则将它拖回到 Error List 的中心方块上。要改变选项卡在选项卡组中出现的位置，把选项卡拖到其他选项卡上，并在所需的位置处释放。

(5) 如果愿意，可以对 IDE 中默认可见的其他窗口或从 View 菜单下找到的窗口重复上面的步骤。花一些时间来熟悉所有不同的窗口，并练习如何把它们排列在屏幕上。由于在本书的其余部分将会频繁地用到这些窗口，所以最好先熟悉它们的位置。

(6) 接下来，双击打开 Solution Explorer 中的 ControlsDemo.aspx 页面(或者如果创建了一个新的 Web 站点，则首先添加一个新的 ASPX 页面)。当该页面打开时，Toolbox 会自动可视。如果不可视，则按 Ctrl+Alt+X 组合键打开它。

(7) 右击 Toolbox 并选择 Add Tab 命令。输入 HTML Fragments 作为它的新名称，并按下 Enter 键。这样就向 Toolbox 中添加了一个新类别，它的行为与所有其他类别一样。

(8) 当文档窗口在 Markup 视图中显示了 ASPX 页面后，则在起始<div>标记后直接输入<h1>。注意 Visual Studio 会自动地插入结束标记</h1>。最后在 Markup 视图中的代码如下所示：

```
<form id="form1" runat="server">
  <div>
    <h1></h1>
  </div>
```

(9) 突出显示起始和结束<h1>标记，然后将 Markup 视图窗口中的选项拖到在步骤(7)中创建的新 Toolbox 选项卡上。该选项会显示为文本：<h1></h1>。

(10) 右击刚刚创建的 Toolbox 项目，选择 Rename Item 命令，并输入 Heading 1 作为其名称。

(11) 再次把光标放到文档窗口中，按下 Ctrl+D 后直接按 Ctrl+K，以便格式化文档窗口中的文档。另一种方法是从主菜单中选择 Edit | Format Document 命令来格式化文档。这是根据在 Text Editor 选项对话框中设定的规则对文档进行格式化的。还可以对一些其他的文档类型进行格式化，包括 C# 和 VB.NET 代码，以及 CSS 和 XML 文件。

从现在起，每当在 Markup 视图中的文档需要一个标题时，只需把光标放在文档窗口中希望标题出现的地方，然后双击 Toolbox 中适当的标题即可。



注意：这个练习用作给 Toolbox 添加代码的示例。对于<h1>元素，可以在代码编辑器中直接输入代码。另外，也可以使用代码片段。在编辑器中输入 h1，再按下 Tab

### 工作原理

该“试一试”练习中的大部分步骤都是容易理解的。首先打开几个构建 Web 站点时经常需要的窗口。然后使用 IDE 的拖放功能将窗口的布局重新排列为个人偏爱的样子。

然后将一个 HTML 片段添加到 Toolbox 的自定义选项卡中。当把任何标记拖到 Toolbox 中时，Visual Studio 都会为它创建一个含有选定标记的 Toolbox 项。每当在页面中需要该标记的一个副本时，只要双击该项，或把它从 Toolbox 中拖到 Markup 视图窗口中即可。这样对于频繁用到的 HTML 片段可以节省很多时间。该技术一般用于较大的代码块，而对于<h1>这样的元素，Visual Studio 有一个更好的工具 Code Snippets，稍后会在本书中介绍该工具。

最后，使用 Visual Studio 的文档格式化选项改变文档中代码的布局。这样有助于保持代码的有序性且使其更易于阅读。要完全改变代码的格式化方式，可以通过 Tools | Options 命令打开 Options 对话框，然后扩展路径 Text Editor | HTML | Formatting，并单击 Tag Specific Options。

除了窗口布局和 Toolbox 外，Visual Studio 允许在 IDE 中进行更多的定制。下一节将讲解如何定制其他 3 个重要的 IDE 功能：文档窗口、工具栏和键盘快捷键。

### 1.5.3 定制文档窗口

Visual Studio 使文本在文档窗口中的显示方式有极大的灵活性。可以修改一些属性，如字体大小、字体颜色，甚至是文本的背景色。要访问 Font and Colors 设置，可以选择 Tools | Options 命令，然后选择 Environment | Fonts and Colors 命令。

在文档窗口中，笔者喜欢定制 tab 的大小，它控制缩进代码时插入的空格数目。要修改 tab 的大小，选择 Tools | Options 命令，然后在 Text Editor 下方选择 All Languages | Tabs 选项。通常将 Tab Size 和 Indent Size 都设置为 2，让 Tab 面板中的其他设置保持不动。笔者还喜欢定制在 HTML 元素前后换行符的数目。Options 窗口为此提供了完全的控制权：选择 Text Editor | HTML | Formatting 命令，然后单击 Tag Specific Options，在左边的列表中可以选择一个标记，再通过右边的设置控制标记的格式化方式。Preview 框便于查看各种设置如何改变格式。

除了将 Tab Size 设置为 2 以及在一些 HTML 元素周围设置的换行符的数目之外，本书所有的屏幕截图显示的都是 VSEW 的默认配置。

### 1.5.4 定制工具栏

工具栏可以通过 3 种方式定制：显示或隐藏内置工具栏、在现有工具栏上添加或删除按钮、使用常用的按钮创建自己的工具栏。

## 1. 启用与禁用工具栏

要禁用和启用已有的工具栏，可以右击任何现有工具栏或者菜单栏，然后从列表中选择合适的项。一旦显示了工具栏，就能用左边的拖动手柄把它拖到工具栏区的新位置。

## 2. 编辑现有工具栏

如果感觉现有工具栏缺少了重要的按钮，或者工具栏中包含几乎用不到的按钮，则可以定制工具栏上的按钮。为此，可以右击任何工具栏或菜单栏，并选择 **Customize** 命令，然后切换到 **Commands** 选项卡，并从 **Toolbar** 下拉菜单中选择希望修改的工具栏。通过右边的 **Command** 按钮，可以添加新的命令并删除已有的命令，或者改变命令的顺序。

## 3. 创建自己的工具栏

如果想将常用的功能组合在一起，那么最好创建自己的工具栏。要创建一个新工具栏，用上一节介绍的方法打开定制窗口。单击 **New** 按钮并输入工具栏的名称。然后切换到 **Commands** 选项卡，用与修改现有工具栏一样的方式修改自己的工具栏。

### 1.5.5 定制键盘快捷键

很多开发人员喜欢修改的另一个设置是键盘快捷键。使用键盘快捷键是节省时间的好方法，因为它们允许用一个简单的键盘命令来执行任务，而不需要拿起鼠标选择菜单的正确项目。要修改键盘快捷键，选择 **Tools | Options**，展开 **Environment**，然后单击 **Keyboard** 按钮。定位到命令列表上要修改快捷键的命令。由于这个列表中含有许多项目，因此可以通过输入命令的几个字母来过滤列表。例如，在 **Show Commands Containing** 字段中输入 **print** 就会给出所有与打印相关的命令。

下一步，在 **Press Shortcut Key** 字段中，输入一个新的快捷键并单击 **Assign**。Visual Studio 允许为一个命令输入两个快捷键。例如，可以将命令 **Close All Documents** 绑定到命令 **Ctrl+K**、**Ctrl+O** 上。要执行这个命令，需要快速而连续地按下组合键。虽然两个快捷键看起来不太必要，但是它大大增加了可用快捷键的数目。

### 1.5.6 重置修改

如果觉得尝试了大量的自定义选项而弄乱了 Visual Studio，也不要担心。有多种办法可以把 Visual Studio 恢复到它以前的状态。

#### 1. 重置 Window 布局

命令 **Reset Window Layout** 可以从 **Window** 菜单中访问，它用来将所有窗口重置为第一次启动 Visual Studio 时的位置。如果误放了太多窗口，最后得到的 IDE 比较混乱，那么这个命令就比较有用。

#### 2. 重置 Toolbox

如果误删了 **Toolbox** 中的项目，或者删除了整个选项卡，可以通过右击 **Toolbox** 并选择 **Reset Toolbox** 命令将 **Toolbox** 重置为它的原始状态。在使用这个命令前需要慎重考虑，因为它也会删除所有的自定义代码段。

### 3. 重置所有设置

如果你跟着前面的“试一试”练习做了，而且又进行了一些自定义设置，那么 IDE 现在很可能是这两种状态之一：要么看起来就是所希望的样子，要么看起来完全混乱了。对于后面这种情况，你需要先学习一下如何轻松地消除混乱状态。

要将所有的 Visual Studio 设置完全恢复到它们刚安装后的状态，请根据所使用的 VS 版本，选择 Tools | Settings | Import and Export Settings 命令，或者选择 Tools | Import and Export Settings 命令。接下来，选择 Reset All Settings 选项并单击 Next 按钮。如果愿意，可以为现有设置创建一个备份；否则，就选择 No，仅重置所有设置即可。此时，会打开另一个界面，允许在众多的设置集合中进行选择。要选择专家设置或 Web 开发，因为这些选项提供了对本书中涉及的所有功能的访问。最后，单击 Finish 按钮。这个动作会导致所有设置重置为它们的默认值，包括窗口布局、工具箱及工具箱定制、快捷键以及可能在 Visual Studio Options 对话框中修改了的所有内容。因此，只有当确实想要 Visual Studio 的一个全新配置时，才使用这个命令。

了解了关于 ASP.NET 页面和 Visual Studio 的一些基本知识后，就可以进行一些实际操作了。在第2章中，将更详细地介绍如何创建 ASP.NET Web 站点和 Web 页面。你将学会如何以一种合乎逻辑的、结构化的方式组织站点，如何将许多不同类型的文件添加到站点中，如何使用它们，以及如何链接到站点中的页面。

但是，在学习第2章之前，还需了解一个更重要的话题：伴随本书的示例应用程序。

## 1.6 示例应用程序

构建 Web 站点是本书的宗旨，因此很有必要先了解一下本书的完整而实用的示例站点，我们将用它来展示 ASP.NET 的很多功能。

本书将构建的这个站点叫做 Planet Wrox，这是一个为对音乐感兴趣的人创建的在线社区。这个站点向它的访问者提供了下列功能：

- 评论管理员在站点上发布的 CD 和音乐会。
- Gig Pics 部分，它是一个在线相册，用户可以共享在音乐会上拍的照片。
- 在站点提供的不同图形化主题之间切换，这样就可以在不更改站点内容的情况下改变站点的外观。
- 音乐首选项，它影响用户在站点上看到的信息。
- 访问注册用户的奖励内容。

站点允许管理员(即作为站点拥有者的你)做下列事情：

- 添加和维护评论。
- 管理系统中不同的音乐类型。
- 管理站点访问者创建的相册。

图 1-15 显示了 Planet Wrox 的主页。

图 1-16 显示了 Planet Wrox 的另一个页面，只是应用的主题不同。该页面允许用户输入他们的个人信息，并根据他们喜爱的音乐类型指定首选项。

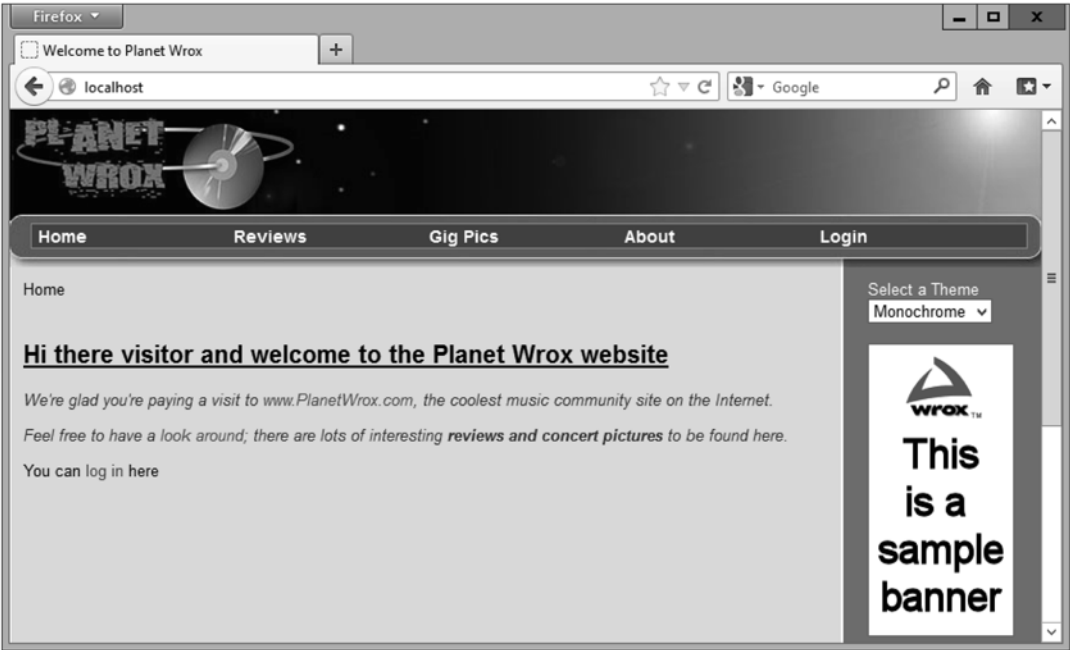


图 1-15



图 1-16

在 [www.PlanetWrox.com](http://www.PlanetWrox.com) 上可以找到一个站点在线运行示例。在该站点上可以从用户的角度使用站点。

也可以从 Wrox 网站 [www.wrox.com/remtitle.cgi?isbn=1118311809](http://www.wrox.com/remtitle.cgi?isbn=1118311809) 上下载该示例应用程序以及本书所有其他示例的源代码。

学习完本书，就能在其他 Web 站点中构建该示例站点的所有功能(甚至更多)。这听起来似乎相

当复杂，但不用太担心。笔者会逐步提供指导，从应用程序开始一直介绍到最后一个功能。只要保持做这件事的兴趣，相信你一定能做好它的。

## 1.7 关于 Visual Studio 的实用提示

本书大部分章节的末尾都有一小节有用的提示。这些提示要么没有出现在正文中的任何地方，要么鼓励你进一步探索或进行某些测试。有时它们可能看起来似乎并不相干，或者乍一看难以理解，但是当顺着本书的指导做下去，回过头来看前面章节的提示时，就会发现它们实际上是有意义的。如果在第一次看到某些内容时不能完全理解，也不要担心。进行一些思考，然后过几天再回顾这一主题。让这个想法稍微深入一点，很可能其意义自然就明白了。这不仅适用于实用提示部分，而且适用于整本书。

- 在开始继续学习第 2 章之前，先多了解一下 Visual Studio。向站点中添加几个页面，从 Toolbox 中拖放一些控件到页面上，并在浏览器中浏览它们。这样，当开始学习第 2 章时就会对其中的工具和许多可用的控件有一个更好的了解。
- 熟悉调整 Visual Studio IDE 的许多选项。在构建 Web 站点时，大部分时间会花在 IDE 上，因此尽可能将其调整为喜欢的样子是很有意义的。不要怕弄乱了，因为无论多乱，总是可以恢复为以前的设置。
- 花一些时间浏览在 Visual Studio 的 Options 对话框中找到的设置(可以通过 Tools | Options 菜单命令访问)。许多设置是容易理解的，而且确实有助于进一步将 IDE 调整为喜欢的样子。

## 1.8 本章小结

本章介绍了很多重要的基础知识，使我们对 ASP.NET 4.5 和 Visual Studio 有了初步了解。首先从总体上简单介绍了 Microsoft .NET Framework 的历史，并特别介绍了 ASP.NET。

然后说明了如何获得和安装 Visual Studio。Visual Studio 是创建 ASP.NET 4.5 Web 页面时使用最为广泛和最为通用的工具。为了更有效地使用这个工具，本章介绍了如何使用和定制 IDE 的主要功能。在接下来的几章中，将进一步介绍使用 VS 中的很多工具的方法。

重要的是要了解 Visual Studio 如何让页面进入 Web 浏览器。了解处理请求的 Web 服务器，了解如何处理页面以及最终发送到浏览器中的 HTML，对于了解 ASP.NET 是至关重要的。本章简要地介绍了 Web 页面的请求方式和提供给浏览器的方式。

第 2 章将更加详细地介绍如何创建 Web 站点。

## 1.9 练习

1. 解释 Visual Studio 中的页面标记与浏览器中的最终 HTML 页面之间的区别。
2. 假设你有若干 HTML 片段想在站点中大量使用。使这些片段在 Visual Studio 中可用的最佳方式是什么？

3. 重置部分或全部 IDE 自定义设置有哪 3 种方法?
  4. 如果要修改页面上某个控件的属性, 如按钮上的文本, 可以使用哪两种方法?
- 练习的答案见附录 A。

#### 本章要点回顾

特性	标记中用于定义或改变其行为的额外信息
元素	包含文本或其他内容的标记对
HTML	超文本标记语言: 浏览器使用该语言显示 Web 页面
HTTP	超文本传输协议: 用于 Web 浏览器和 Web 服务器之间的通信
IDE	集成开发环境: 一个由应用程序和工具所组成的集合, 用于开发应用程序
JavaScript	一种用于与客户端浏览器上显示的 Web 页面进行交互的编程语言
Tag	用尖括号括住的文本, 用来创建 HTML 元素
Visual Studio 2012	创建 .NET 应用程序的开发环境
Visual Studio Express 2012 for Web	Visual Studio 的免费版本, 用于创建 ASP.NET Web 应用程序

# 第 2 章

## 构建 ASP.NET Web 站点

### 本章要点

---

- 构建 ASP.NET Web 站点的不同项目类型
- 用于站点开发的不同项目模板
- ASP.NET 中众多可用的众多文件类型及它们的用途
- 创建易于现在和将来管理的结构化 Web 站点的方法
- 如何使用设计工具创建格式化的 Web 页面

要创建漂亮、实用的成功 Web 站点，就必须了解几种重要的技术与语言，包括 HTML、ASP.NET、CSS(Cascading Style Sheets, 层叠样式表)、服务器端编程语言(如 C#或 VB)以及客户端语言(如 JavaScript)。本章和接下来的几章将奠定这几种技术的坚实基础，因此在学完全书后你会非常熟悉这些最重要的概念。

除了这些技术外，你还必须了解在第 1 章中介绍的 Visual Studio IDE。需要知道如何创建站点，添加页面和管理 Visual Studio(VS)提供的所有工具栏和窗口。此外，还需要知道如何在 VS 中用 HTML 和服务器控件构建和设计 Web 页面。

本章将详细介绍如何创建与管理 Web 站点。还将介绍如何创建 ASP.NET Web 页面、向它们添加标记，从而使创建的页面能够向用户展示信息，并对他们的行为作出响应。

虽然在第 1 章中你已经创建了第一个 ASP.NET Web 站点，但本章将开始更深入地介绍如何新建一个 Web 站点。因为当你开始创建一个新站点时需要作许多选择，所以了解所有不同的选项并为你的场景选择正确的选项是很重要的。

### 2.1 使用 Visual Studio 2012 创建 Web 站点

第 1 章快速概览了如何使用 VS 创建 Web 站点。你只需简单地从 File 菜单中选择 New Web Site 命令，选择一种语言，选择标准的 ASP.NET Web Forms Site 模板，并单击 OK 按钮即可。然而，New Web Site 对话框中的内容远不止第 1 章中提到的那些。你可能已经注意到，可以从若干模板中选择



一个模板来创建不同类型的站点。但是在介绍用来创建新 Web 站点的各种模板前，我们将大致介绍一下 VS 中可用的不同项目类型。

### 2.1.1 不同的项目类型

在 Visual Studio 2012 中，可以为创建 ASP.NET Web Forms 网站选择两种项目：Web Application Project 和 Web Site Project。

#### 1. Web Application Project

Web Application Project 使得团队开发人员，以及那些需要对站点内容和编译及部署过程有更多控制的开发人员更容易使用 VS 构建 Web 站点。整个 Web 站点作为一个项目进行管理，用单个项目文件跟踪 Web 站点的所有内容。

在 VS 中，可以通过 File | New Project 对话框来创建一个新的 Web Application Project。在该对话框中，单击首选编程语言(可以是 Visual Basic 或 Visual C#)，然后单击 Web 类别，其中有若干个 ASP.NET Web 应用程序模板。其中一个可用的项目模板是 ASP.NET MVC 4 Web Application，它是根据模型-视图-控制器(Model View Controller, MVC)模式来创建应用程序的，该模式是 Web 应用程序开发的另一个流行方式。本书中不使用或讨论 MVC，如果要了解关于它的更多内容，可以参考 [www.asp.net/mvc](http://www.asp.net/mvc)。

#### 2. Web Site Project

Web Site Project 表示在 VS 中创建的 Web 站点项目。创建新的 Web Site Project 的方法是：从 Visual Studio 的主菜单中选择 File | New Web Site 命令，或者选择 File | New | Web Site 命令。

Web Site Project 站点只是一个 Windows 文件夹，以及其中的一组文件和子文件夹。在该 Web 站点中没有跟踪所有单个文件的集合文件(称为项目文件，扩展名是.vbproj 或.csproj)。只需将 VS 指向一个文件夹，它就会一直把这个文件夹作为 Web 站点打开。这样就可以非常容易地创建站点的副本、移动它们或者与别人共享，因为它不依赖于本地系统中的文件。由于缺少中心项目文件，因此 Web Site Projects 通常简称为 Web Sites，在本书的其余部分就使用这个术语。

#### 3. 在 Web Site Project 和 Web Application Project 之间选择

由于有两个选项可供选择，那么应选择哪个项目类型呢？一般来说，Web Site Project 更容易使用，因为它只是一个文件夹，所以更容易把文件复制到另一个位置，例如另一个开发工作站或生产服务器。另外，对代码文件的修改会由 Web 服务器提取，并自动应用，无需正规的部署过程。另一方面，如果一组开发人员在同一个站点上工作，Web Application Project 就比较好，因为它具备更正式的开发和部署过程，更好地支持源控制版本系统，例如 Microsoft 的 Team Foundation Server。

本书中使用的是 Web Site Project 模板，这是因为该模板对于 ASP.NET 初学者来说很容易使用。但是接下来你就会发现在构建站点时，使用 Web Application Project 模板与使用 Web Site Project 模板有很多共同之处。这意味着从本书获得的知识可以用于通过 Web Application Project 模板建立站点。如果希望跟随本书，则需要使用 Web Site Project 模板。如果没有指定特定的项目类型，那么本书在涉及一般的 Web 站点时，将互换使用 Web 站点和 Web 应用程序这两个术语。

了解了不同的项目类型后，接下来就要考虑不同的 Web 站点模板及它们的选项。

### 2.1.2 选择正确的 Web 站点模板

VS 中的 New Web Site 对话框包含不同的 Web 站点模板，各个模板的用途各不相同。

图 2-1 显示了 VS 中的 New Web Site 对话框。根据你的 VS 版本，可以选择 File | New Web Site 或者 File | New | Web Site 命令打开这个对话框。如果出现对话框与图 2-1 不同，则要确保选择的是 File | New Web Site，而不要误选了 File | New Project。

在对话框的左侧部分，可以从 Visual Basic 和 Visual C# 中选择一种作为站点的编程语言。中间的部分显示了默认安装的 ASP.NET Web 站点模板。其中的每一个模板都会在下一节中讨论。创建自己的模板时(第 6 章将介绍如何创建自己的模板)，或者安装其他方提供的模板时，它们也会显示在这个区域中。

在 Planet Wrox Web 站点中，ASP.NET Empty Web Site 模板是贯穿全书用到的。其他的模板只在下面几节中作简要描述，以便你了解如何使用它们。系统安装的模板的精确列表取决于 Visual Studio 的版本和安装的组件。即便其中有其他的模板也不要担心，只要有 ASP.NET Web Forms Site 和 ASP.NET Empty Web Site 模板即可。

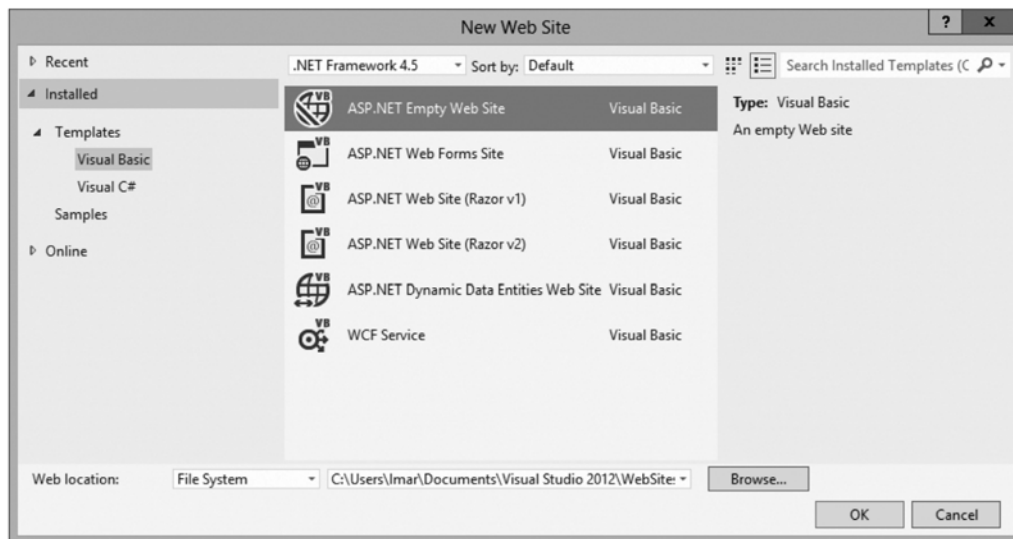


图 2-1

#### 1. ASP.NET Web Forms Site

这个模板允许配置一个基本的 ASP.NET Web 站点。它包含许多文件和文件夹用于开始站点的开发。各种不同的文件类型将在本章后面讨论。特殊的 App\_Data 文件夹和 Account 文件夹中页面的功能将在本书后面讨论。

一旦开始开发一个真实的 ASP.NET Web 站点，该模板会是一个很好的起点。

#### 2. ASP.NET Web Site(Razor v1 或 Razor v2)

通过 Microsoft 的 Web Pages 框架，使用这些模板可以创建站点。Web Pages 可参阅图书 *Beginning ASP.NET Web Pages with WebMatrix*(Wrox, 2011, ISBN: 978-1-118-05048-4)。

### 3. ASP.NET Empty Web Site

ASP.NET Empty Web Site 模板只包含一个配置文件(Web.config)。如果包含用于创建新的 Web 站点或希望从头开始创建站点时使用的许多已有文件,则 ASP.NET Empty Web Site 模板会很有用处。使用该模板作为本书中构建的示例 Web 站点的基础,并在学习本书的过程中添加文件和文件夹。

### 4. ASP.NET Dynamic Data Entities Web Site

这个模板用于创建灵活且强大的 Web 站点来管理数据库中的数据,而不需要手动输入许多代码。本书不讨论这些模板,但是将在第 14 章中更多地介绍该模板使用的 Microsoft ADO.NET Entity Framework。

### 5. WCF Service

该模板可用于创建包含一个或多个 WCF(Windows Communication Foundation) Service 的 Web 站点。WCF Service 模板与 Web Service 模板有点相似,它用来创建可通过网络调用的方法。然而,WCF Services 比这个简单的 Web 服务要复杂得多,而且提供了更大的灵活性。第 10 章将会介绍如何在浏览器中创建和使用 Web 服务。

虽然你似乎必须对正确的 Web 站点模板作出一个清晰的选择,但其实没有关系。由于 VS 中的 ASP.NET Web 站点实质上只是对文件夹的一个引用,因此很容易将一个模板的类型添加到另一个模板上。例如,向标准的 ASP.NET Web Forms Site 或 ASP.NET Empty Web Site 中添加 Web 服务文件,是完全可以接受的(而且非常常见),如第 10 章所述。

## 2.1.3 创建与打开新的 Web 站点

有几种不同的方式可以创建新的 Web 站点和打开现有 Web 站点。选择哪种方式在很大程度上取决于访问 Web 站点的方式(从安装了 VS 的本地计算机还是从远程计算机),以及你是想使用 VS 配备的内置 Web 服务器 IIS Express,还是使用 Windows 自带的 Web 服务器。

本书的所有示例都假定从本地硬盘驱动器中打开站点,并且使用 IIS 的删节版本 IIS Express,因为使用它开发站点非常方便。然而,第 19 章将介绍如何使用和配置 Internet Information Services,或者简称为 IIS。几乎 Windows 的所有版本都配备了这个高级 Web 服务器。IIS 主要用于在 Windows 的服务器版本上驻留 Web 站点的产品,因为它能在高通信量的情况下服务 Web 页面。

### 1. 创建新的 Web 站点

下一个“试一试”练习将带领你创建 Planet Wrox Web 站点,这个站点是本书将要使用的项目。除了特别说明外,本书余下部分的所有练习都假定你在 VS 中打开了此 Web 站点。该练习要求将 Web 站点存储在 C:\BegASPNET\Site 文件夹中。注意一下这个文件夹的名称,因为它在全书中都要用到。如果你决定使用其他文件夹,则一定要确保每次见到本书中的这个名称时,都使用你自己的位置。还要确保没有使用诸如井号(#)这样的特殊字符,也不要再在文件夹的名称中插入空格,以免在构建站点时出现问题。最后,确保不要在 Windows 的 Documents 文件夹(一般是 C:\Users\UserName\Documents)下创建这个文件夹,因为以后其他账户需要访问该站点时会出问题。

## 试一试

## 创建一个新的 ASP.NET 4.5 Web 站点

(1) 首先用 Windows 资源管理器或“我的电脑”在 C 盘的根文件夹中创建一个名为 BegASPNET 的文件夹。在这个文件夹内再创建一个文件夹，命名为 Site。最终应得到一个全名为 C:\BegASPNET\Site 的文件夹。如果你根据本书的“前言”中的指示做了，并且解压了本书的源代码包，那么你就已经有了 BegASPNET 文件夹。这个文件夹中又包含了 Source 和 Resources 文件夹，但还是需要创建 Site 文件夹。如果希望同时使用 VB.NET 和 C# 语言，则可以创建两个文件夹——BegASPNETVB 和 BegASPNETCS，并使用 Visual Studio 的两个实例。

(2) 启动 Visual Studio，并根据你的 VS 版本选择 File | New Web Site 或 File | New | Web Site 命令。



**常见错误：**不要使用 File | New Project 错误地创建一个新的 Web Application Project，因为该项目模板不适合本书中的练习。

(3) 在屏幕顶部的目标框架下拉列表中，选择 .NET Framework 4.5。

(4) 在左侧的 Installed Templates 区域中选择 Visual Basic 和 Visual C#。本书中所有的示例都是用这两种编程语言显示的，因此你可以选择一种最喜欢的语言。

(5) 在中间区域选择 ASP.NET Empty Web Site。

(6) 在 Web Location 的下拉列表中，确保选中了 File System。列表中还有其他两个选项，分别是 HTTP 和 FTP。HTTP 允许在本地计算机上，或使用 Microsoft FrontPage Server Extensions 在一台远程服务器上打开一个运行 IIS 的站点，而 FTP 则用来在 FTP 服务器中打开一个站点。

(7) 单击 location 文本框旁边的 Browse 按钮，浏览在本练习第一步中创建的文件夹 C:\BegASPNET\Site，并单击 Open 按钮。

最终的屏幕应如图 2-2 所示，只是你可能把它设成了 Visual C# 而不是 Visual Basic。

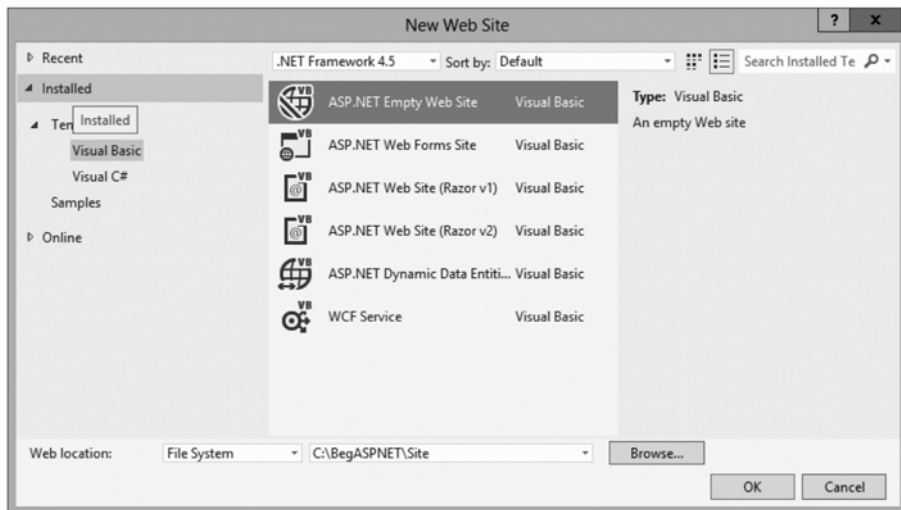


图 2-2

(8) 单击 OK 按钮后，VS 会创建一个新的站点。

工作原理

单击 OK 按钮后，VS 会创建一个新的空 Web 站点。这个新的 Web 站点中只包含一个配置文件(名为 Web.config)。在 Solution Explorer 中，Web 站点现在如图 2-3 所示。如果看不到顶部的 Solution 节点，可以在 VS 中选择 Tools | Options, 在 Projects and Solutions 类别中选择 Always Show Solution。

在 VS 看来，基于 Empty Web Site 模板的 Web 站点只是一个简单的 Windows 文件夹，它是在磁盘上包含同样文件的实际文件夹。从图 2-4 中可以看出，创建该站点不需要其他文件，该图是一个 Windows 资源管理器，显示了文件夹 C:\BegASPNET\Site 中的文件。

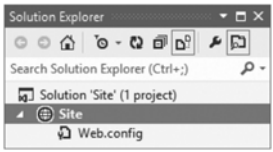


图 2-3

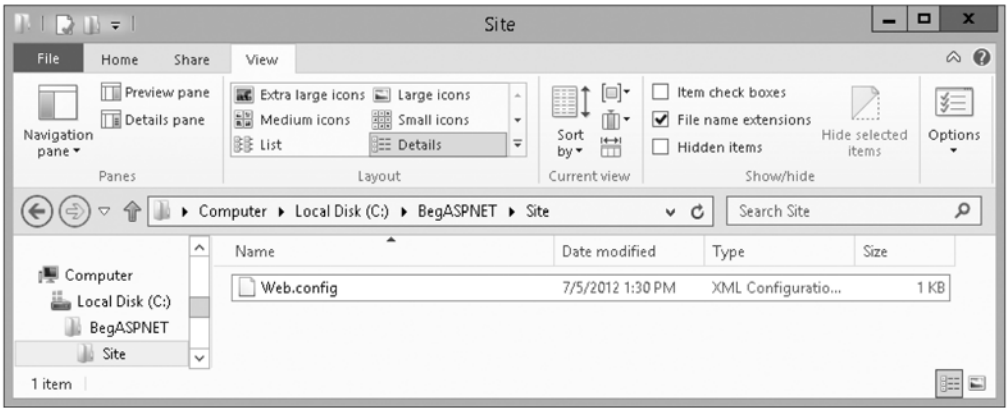


图 2-4

如果没有看见 Web 文件的.config 扩展名，不用担心。在稍后的练习中将会介绍如何查看文件的扩展名。

当你按部就班地照本书学下去时，自然会向这个站点中添加新的文件和文件夹。这些附加的文件和文件夹会显示在 Solution Explorer 中，也会出现在 Windows 文件夹 C:\BegASPNET\Site 中。

打开基于 Web Site Project 模板的 Web 站点与创建新站点的过程非常相似。下一节将介绍如何在 VS 中打开现有站点。

2. 打开现有 Web 站点

与创建新站点一样，在 VS 中打开现有站点时，根据 Web 站点的源位置，会有几种选择。可以选择从本地文件系统中打开站点，也可以从本地 IIS Web 服务器中打开站点，还可以用 FTP 从远程服务器中打开站点，或者用 Microsoft FrontPage Server Extensions 从远程站点中打开，或者从中心源控制系统如 Microsoft Team Foundation Server 中打开。图 2-5 显示了 VS 中的 Open Web Site 对话框。

要打开这个对话框，可以选择 File | Open Web Site(不要误选了 File | Open Project, 因为这个菜单项用于打开 Web Application Project)。本书的所有示例都假定你总是使用窗口左栏的第一个按钮 File System 从本地文件系统中打开 Planet Wrox Web 站点。然后在右边的窗格中定位到 Web 站点(本例中是 C:\BegASPNET\Site)并单击 Open 按钮。

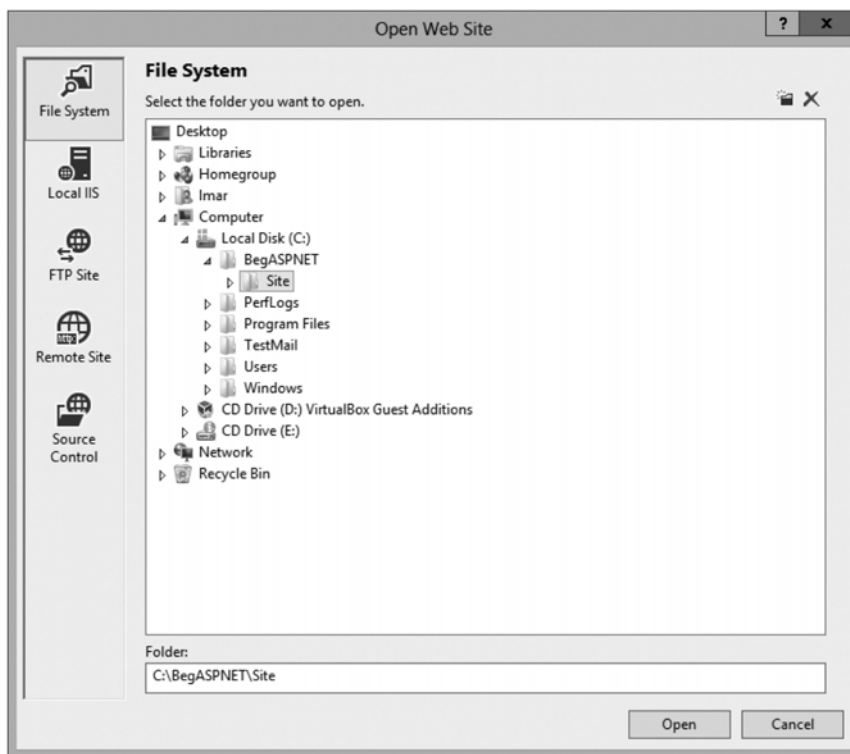


图 2-5

在上一个“试一试”练习中创建的站点是个空站点。要让站点更有用，就要向其中添加文件。下一节将着重讨论要添加到站点中的多种文件类型以及将文件添加到站点中的方式。

## 2.2 操作 Web 站点中的文件

虽然 ASP.NET 4.5 Web Forms 站点至少由一个 Web 窗体(扩展名为.aspx 的文件)组成，但是它常常是由更多文件组成的。VS 中有许多不同的文件类型可用，各个类型提供了不同的功能。下一节将介绍 VS 中用到的最重要的文件类型。此外，还会介绍向站点中添加这些文件的几种不同方法。

### 2.2.1 ASP.NET 4.5 Web 站点的文件类型

为了让你对各种不同类型的文件在 ASP.NET 中的使用有个印象，图 2-6 显示了允许向站点中添加新文件的对话框(要打开该对话框，可以右击 Solution Explorer 中的 Web 站点并选择 Add | Add New Item 命令，或者从主菜单中选择 Website | Add New Item 命令)。

为了便于找到希望的文件类型，可以使用对话框右上角的搜索框。只需输入所需文件类型的几个字母，并按下回车键，VS 就会过滤出一个匹配搜索短语的文件列表。

这些能够添加到站点中的文件可以分组到不同的类别中。下面将讨论其中最重要的文件(贯穿全书的示例都用到的文件)。



图 2-6

1. Web 文件

Web 文件是 Web 应用程序中特有的文件，可以由浏览器直接请求，也可以用来构建在浏览器中请求的 Web 页面的一部分。表 2-1 中列出了在 ASP.NET Web Forms 网站中常用的各种 Web 文件和它们的扩展名，并说明了各种文件的用法。

表 2-1

文件类型	扩展名	说明
Web Form	.aspx	这些文件是所有 ASP.NET Web 站点都要用到的文件。Web 窗体表示用户在他们的浏览器中浏览的页面
Master Page	.master	这些文件允许定义 Web 站点的全局结构和外观。在第 6 章中可以看到它们的用法
Web User Control	.ascx	包含可在站点的多个页面中重用的页面片段。第 8 章将专门介绍用户控件
HTML Page	.htm/ .html	可用来显示 Web 站点中的静态 HTML
Style Sheet	.css	包含允许定制 Web 站点的样式和格式的 CSS 代码。第 3 章将介绍关于 CSS 的更多信息
Web Configuration File	.config	包含用在整个站点中的全局配置信息。在本书后面，从第 4 章开始，你将了解如何使用 Web.config 文件
Site Map	.sitemap	包含一个层次结构，表示站点中 XML 格式的文件。Site Map 用于导航，并会在第 7 章讨论
JavaScript File	.js	包含可以在客户端浏览器中执行的 JavaScript
Skin File	.skin	包含 Web 站点中的控件的设计信息。Skin 将在第 6 章讨论

下面的“试一试”练习展示了如何向这个全书都会用到的站点中添加新的母版页。

试一试

向站点中添加文件

- (1) 如果站点还没有打开，则选择 File | Open Web Site 命令打开先前创建的 Planet Wrox Web 站点。一定要从 File System 中打开站点，定位到包含站点的文件夹(C:\BegASPNET\Site)，并单击 Open 按钮。
- (2) 在 Solution Explorer 中，右击站点并选择 Add | New Folder 命令。



**常见错误：**一定要单击实际站点而不是 Web.config 文件或顶部的 Solution 节点(如图 2-3 所示)，否则就得不到正确的菜单项。

- (3) 输入 MasterPages 作为文件夹的名称，并按下 Enter 键。然后右击这个新文件夹，并选择 Add | Add New Item 命令。另一种方法是从 Visual Studio 的主菜单中选择 File | New File 命令或 Website | Add New Item 命令。或者在 Solution Explorer 中单击新文件夹，使之突出显示，然后按下 Ctrl+Shift+A。
- (4) 在显示的对话框中，选择 Master Page 并将其命名为 Frontend。当添加文件时，VS 会自动地添加.master 扩展名。确保在 Installed Templates 中已经选择了要用于站点的语言，并选中了右下角的 Place Code in Separate File 复选框。最后，单击 Add 按钮，就将母版页添加到了站点中，并且已经在文档窗口中自动打开。

工作原理

这个简单的练习显示了如何将新项添加到 Web 站点中。虽然在这一阶段站点还不是很完善，但是你添加的文件已经形成了本书其余内容的基础。下一节将简单地介绍其余的文件类型。

2. 代码文件

将代码文件添加到站点中的方式与添加 Web 文件的方式相同。表 2-2 描述了各种类型的代码文件。

表 2-2

文件类型	扩展名	说明
WCF Service	.svc	可以被其他系统调用，包括浏览器，可以包含能在你的服务器上执行的代码。WCF Service 将在第 10 章介绍
Class	.cs / .vb	可以包含构建 Web 站点的代码。注意 Code Behind 文件(稍后讨论)也有这样的扩展名，因为它们实质上是类文件。C#使用带有.cs 扩展名的文件，而 Visual Basic 使用的是带有.vb 扩展名的文件
Global Application Class	.asax	可以包含为了响应站点中的有趣事情而触发的代码，例如应用程序的开头或者当在站点中某处发生错误时。在第 11 章和第 18 章你将看到如何使用这个文件

除了 Code Files 类别外，还有一组值得研究的文件：Data Files。



### 3. 数据文件

数据文件用来存储可以用在站点和其他应用程序中的数据。这组文件由 XML 文件、数据库文件以及与使用数据相关的文件组成，如表 2-3 所示。

表 2-3

文件类型	扩展名	说明
XML File	.xml	用来存储 XML 格式的数据。除了纯 XML 文件外，ASP.NET 还支持几种基于 XML 的文件，其中两种你在以前已经见过：Web.config 和 Site Map
SQL Server Database	.mdf	扩展名为.mdf 的文件是 Microsoft SQL Server 使用的数据库。在第 12 章及以后章节中将会讨论数据库
ADO.NET Entity Data Model	.edmx	用于声明性地访问数据库，不需要写代码。从技术上来讲，这并不是一个数据文件，因为它不包含实际数据。然而，由于它们与数据库绑定得如此紧密，因此把它们归组在这个标题下是有意义的。在第 14 章你将了解关于 ADO.NET Entity Framework 的更多内容

如前面的“试一试”练习所述，添加这些类型之一的任何文件真的很容易。只要轻松地将现有文件添加到站点中即可。

#### 2.2.2 添加现有文件

在 Web 站点中创建的文件不一定都要是全新的。在有些情况下可以重用其他项目中的一些文件。例如，在多个站点上重用 logo 或 CSS 文件。要添加现有文件，可以在 Solution Explorer 中右击 Web 站点，并选择 Add | Add Existing Item 命令。在出现的对话框中，可以浏览文件，也可以通过按住 Ctrl 键来选择多个文件。最后，当单击 Add 按钮时，文件就添加到 Web 站点中了。也可以使用复制和粘贴命令，把文件从本地磁盘的文件夹复制到 VS 的 Web 站点中。只需要在 Windows 资源管理器中突出显示文件，按下 Ctrl+C 来复制文件，再切换到 VS，单击 Solution Explorer 中的 Web 站点(或者站点中的一个子文件夹)，按下 Ctrl+V，文件就复制到 Web 站点的文件夹中了。

然而，向站点中添加文件还有一个更容易的方式，在需要向站点中添加多个现有文件和文件夹时，用这种办法会节省大量时间：拖放。下面的“试一试”练习就说明了具体做法。

试一试

向站点中添加现有文件

(1) 在 Windows 环境下，最小化所有打开的应用程序，右击桌面并选择 New | Text Document 命令。如果没有看到这个选项，则只需要用 Notepad 简单地创建一个新的文本文件并保存在桌面上。

(2) 将该文本文件重命名为 Styles.css。一定要用.css 替换.txt 扩展名。如果没有看到初始的.txt 扩展名，而且文件的图标也没有从文本文件转换成 CSS 文件(默认情况下它与文本文件的图标相同，只是上面有一个齿轮符号，不过你可能安装了可以修改 CSS 文件图标的软件)，那么就表示 Windows 配置为隐藏已知文件类型的扩展名了。如果是这种情况，则在 Windows 7 中打开 Windows 资源管理器，单击 Organize 按钮，然后选择 Folder and Search Options 选项。切换到 View 选项卡，并取消选中 Hide Extensions for Known File Types 选项。对于 Windows 8，在 File Explorer 的功能区中，可

以找到 View 选项卡中的 File Name Extensions, 如图 2-4 所示。现在你可能需要将文件名从 Styles.css.txt 改为 Styles.css。

当将文件名从.txt 改为.css 时, Windows 可能会给出一个警告, 指出如果继续进行下去, 文件可能会变得不可用。你可以对这个问题放心地回答 Yes 以继续转换。

(3) 重新排列 VS, 以便也能看到桌面上带 CSS 文件的部分。使用 VS Windows 标题栏上 Close 按钮旁边的 Restore Down 按钮, 可以使 VS 退出全屏模式。

(4) 单击桌面上的 CSS 文件, 按住鼠标按钮将该文件拖到 Solution Explorer 中。一定要把文件拖到 Solution Explorer 中, 而不要拖到 VS 中的其他部分, 否则文件就添加不进去。例如, 当把它拖到文档窗口时, VS 只是简单地打开文件, 而不是把它添加到站点中。

(5) 当在 Solution Explorer 中的 Web 站点节点或现有文件上释放鼠标时, CSS 文件就会添加到站点中。



**注意:** 如果使用 Windows 7 并以管理员的身份运行 VS, 那么将可能无法工作, 因为 Windows 不允许 Windows Explorer 和 VS 相互通信。在这种情况下, 可以使用之前讨论的 Add Existing Item 菜单或复制粘贴命令来添加现有文件。

### 工作原理

这个简单的“试一试”练习只使用了基本的 Windows 技巧, 但它说明了, 将文件添加到站点中时, VS 创建了该文件的一个副本。因此, 在对 VS 中的副本作修改时不会影响桌面上的原始 Styles.css 文件。这样, 就容易将现有 Web 站点之外的文件拖放到新站点中, 而不会影响原始文件。使用 VS 中的 Add Existing Item 对话框添加文件时也是如此。

如果在 VS 之外向 Web 站点的文件夹中添加文件, 它们可能不会立即显示出来。单击 Solution Explorer 的工具栏上的 Refresh 按钮, 可以得到文件列表的一个刷新副本。

### 2.2.3 组织站点

由于组成站点的文件有很多, 因此最好按功能将它们分组到单独的文件夹中。例如, 所有的 Style Sheet 文件都能归到 Styles 文件夹中, .js 文件可以归到 Scripts 文件夹中, 用户控件可以归到 Controls 文件夹中, 母版页可以存储在 MasterPages 文件夹中。这是个人习惯问题, 但是结构化和组织良好的站点更容易管理与理解。下面的“试一试”练习解释了如何将文件移动到新的文件夹中, 以便管理该站点。

#### 试一试

#### 组织 Web 站点

- (1) 右击 Solution Explorer 中的 Planet Wrox 站点, 并选择 Add | New Folder 命令。
- (2) 输入 Styles 作为新文件夹名并按下 Enter 键。
- (3) 创建另一个文件夹 Controls。本书的其余部分会用到这两个文件夹。
- (4) 把之前添加的 Styles.css 文件拖放到 Styles 文件夹中。

如果一切顺利, 那么现在的 Solution Explorer 应该如图 2-7 所示。

如果你的 Solution Explorer 看上去不同于图 2-7, 请再根据本“试一试”练习做一次, 直到你的站点看起来与图 2-7 完全相同为止, 里面的文件夹结构和文件都要相同。本书以后的“试一试”练习都假定你在 Web 站点中有了正确的文件夹和文件。

### 工作原理

结构和组织对于站点的管理很重要。虽然你可能想将所有文件都添加到项目的根文件夹中, 但是最好不要这么做。如果是非常小的站点可能看不出什么差别, 但是一旦站点开始变大, 你就会发现, 如果没有良好的结构, 文件就变得非常难以管理。将相关的文件放在单独的文件夹中是构建有组织的站点的第一步。而将相同类型的文件放在一个文件夹中只是优化站点的一种方法。在后面的章节中, 独立的文件夹也常用来分组功能类似的文件。例如, 所有只能由站点管理员访问的文件都放在一个名为 **Management** 的文件夹中。

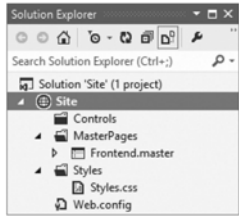


图 2-7

VS 的拖放功能可以轻松地重组站点。只要简单地选择一个或多个文件, 并把它们拖到新位置上即可。如果在扩展站点时继续用这些组织方法, 则第二天或者从现在起的 6 个月内, 在需要时就可以没有任何问题地找到正确的文件。

### 2.2.4 特殊文件类型

上一节列出的有些文件要求把它们放在一个特殊文件夹中, 这种特殊文件夹是相对于可选组织化文件夹结构而言的。当你试图将一个文件添加到它的特殊文件夹之外时, IDE 就会发出警告, 而且会提示创建该文件夹并将文件放在那里。例如, 当试图向站点中添加一个类文件时(扩展名为 .vb 或 .cs), 就会看到如图 2-8 所示的警告。



图 2-8

当看到这个对话框时, 总是单击 **Yes** 按钮。否则文件就不能正确地起作用。其他文件类型也有类似的对话框, 包括 **skin** 和数据库文件。

你已经对组成 Web 站点的各种文件类型有了很好的了解, 现在我们就来更具体地看一下其中的一种类型: **.aspx** 文件, 也称为 **Web 窗体**。

## 2.3 使用 Web 窗体

由.aspx 文件表示的 Web 窗体是任何一种 ASP.NET 4.5 Web Forms 网站的核心。它们是用户访问

站点时在其浏览器中看到的实际页面。

如第 1 章所述, Web 窗体可能由这些内容混合而成: HTML、ASP.NET 服务器控件、客户端 JavaScript、CSS 和编程逻辑。为了更容易地看到这些代码最终出现在浏览器中的状态, VS 提供了关于页面的几种不同的视图。

### 2.3.1 Web 窗体的不同视图

VS 允许从几个不同的角度查看 Web 窗体。当一个带有标记的文件(如 Web 窗体或母版页)在文档窗口中打开时,会看到在窗口的左下方有 3 个按钮。使用这 3 个按钮可以在不同的视图间切换,如图 2-9 所示。这个图显示了母版页,参见第 6 章。



图 2-9

Source 视图是打开页面时的默认视图。Source 视图显示了原始 HTML 和页面的其他标记,而且如果要调整页面的内容或想修改某处时,它就非常有用。第 1 章解释过,使用术语 Markup 视图来指代 ASPX 和 HTML 页面的标记,而不使用 Source 视图这一术语。

Design 按钮允许将文档窗口切换到 Design 视图,这样就能知道页面最终是什么样子。在 Design 视图中,可以用 View 主菜单下的 Visual Aids 和 Formatting Marks 子菜单来控制可视化标志,比如换行符、边界和空格。这两个子菜单都提供了一个 Show 菜单项,允许立即打开或关闭所有 Visual Aids。如果想看看页面最终出现在浏览器中的样子,就把这两者都关闭。然而,只应当使用 Design 视图来了解页面最终的样子。虽然 VS 有一个出色的呈现引擎,可以把页面相当漂亮地展现在 Design 视图中,但还是应当经常在不同的浏览器中检查页面,因为在 VS 中看到的是页面处理前的标记。页面上的服务器控件可能会给出修改浏览器中页面外观的 HTML。因此,建议尽可能频繁地在浏览器中浏览页面,这样就能检查出页面的外观是不是你想要的样子。同时建议你在所能使用到的众多不同的浏览器中检测站点,因为它们显示 Web 页面的方式可能略有不同。Planet Wrox Web 站点已经用 Microsoft Internet Explorer、Firefox、Google Chrome、Safari 和 Opera 检测过了。在本书的不同地方,将会看到这些浏览器的屏幕截图。

Split 按钮可以用来同时查看 Design 视图和 Markup 视图,如图 2-10 所示。

如果要查看向页面的 Design 视图中添加控件时 VS 生成的代码,那么 Split 视图就是相当不错的选择。另一种方法也非常有用,即在 Markup 视图中修改页面的标记时,可以查看它最终在 Design 视图中的样子。有时 Design 视图会变得与 Markup 视图不同步。如果发生了这种情况,Design 视图

上方会出现一条消息。只需要单击这条消息或者保存整个页面，就可以更新 Design 窗口。



图 2-10

使用 Ctrl+Page Up 和 Ctrl+Page Down 键可以遍历这三种不同的模式。

如果希望在 Markup 视图以外的视图中打开页面，则选择 Tools | Options 命令。然后展开 HTML Designer，并且在 General 类别中设置你喜欢的视图。或者在 VS 右上角的 Quick Launch 文本框(按下 Ctrl+Q 可以访问它)中输入 HTML Designer General，再单击所出现的列表中的项。

除了 HTML 及在 Markup 视图窗口中看到的其他标记外，Web 窗体也可能包含使用 C#或 Visual Basic .NET 编写的代码。代码所放的位置取决于所创建的 Web 窗体的类型。下一节将更详细地解释这两个选择。

### 2.3.2 在 Code Behind 和带内联代码的页面之间选择

Web 窗体有两种形式：一种是带 Code Behind 文件的.aspx 文件(根据带附加.vb 或.cs 扩展名的 Web 窗体命名的文件)，另一种是嵌套了代码的.aspx 文件，常称为带内联代码(inline code)的 Web 窗体。虽然在第 5 章之前你不会看到太多的代码，但是了解这些 Web 窗体类型之间的区别还是很重要的。首先，带内联代码的 Web 窗体看起来要稍微容易理解一些。由于构建 Web 站点所需的代码是相同的 Web 窗体部分，因此可以清楚地看到代码与文件是如何关联的。然而，随着页面变得越来越大，你也向页面中添加了更多的功能，那时把代码放在单独的文件中通常会更加方便。那样，就把代码完全从标记中分离了出来，你就可以集中精力解决手头的任务。

在下面的“试一试”练习中将添加两个文件，它们演示了 Code Behind 和内联代码的区别。

#### 试一试

#### 向站点中添加带代码的 Web 窗体

在本练习中添加的文件不是最终的应用程序所需要的文件。为了避免把项目弄混乱，应把它们放在一个单独的 Demos 文件夹中。

(1) 在 Solution Explorer 中，右击 Web 站点并选择 Add | New Folder 命令。将文件夹命名为 Demos 并按下 Enter 键。

(2) 右击 Demos 文件夹并选择 Add | Add New Item 命令。在出现的对话框的左方选择喜欢的编程语言，单击 Web 窗体模板并命名文件为 CodeBehind.aspx。确保选中了 Place Code in Separate File 复选框。最后，单击 Add 按钮。这个页面应当已在 Markup 视图中打开了，因此可以看到页面的 HTML。

(3) 在文档窗口下方，单击 **Design** 按钮将页面从 **Markup** 视图切换到 **Design** 视图。可以看到这个页面的背景是白色的，上方有一个小的虚线矩形。这个虚线矩形是在 **Markup** 视图中看到的 `<div>` 元素。

(4) 从 **Toolbox** 中，将 **Standard** 类别中的一个 **Label** 控件拖到页面的虚线框区域中。记住，如果 **Toolbox** 还没有打开，可以用快捷键 **Ctrl+Alt+X** 打开。在 **Design** 视图中，屏幕现在应如图 2-11 所示。



图 2-11

(5) 双击 `<div>` 元素虚线下方某处的空白区域，VS 就会从 **Design** 视图切换到文件的 **Code Behind** 模式，并添加向浏览器中加载页面时激活的代码。

### VB.NET

```
Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load
End Sub
```

### C#

```
protected void Page_Load(object sender, EventArgs e)
{
}
```

虽然这种奇怪的代码这时看起来有点吓人，但是不要太担心。如前所述，在大多数情况下，VS 会自动添加它。在以后的章节中，将具体看到这样的代码是如何工作的，不过目前重要的是了解将要放在这些代码行之间的代码，在 **Visual Basic** 中是以 **Protected Sub** 开头，以 **End Sub** 结束，在 **C#** 中是以花括号括起来的，当在浏览器中请求页面时就会运行它们。

从现在开始看到的所有代码示例都包括 **Visual Basic(VB.NET)**和 **C#**版本，因此你总是能选择一种适合你的编程语言的代码。

(6) 把光标放在 VS 创建的代码的起始行上，并将突出显示的指示今天日期与时间的代码添加到 **label** 中，这个 **label** 最终会出现在浏览器中。

### VB.NET

```
Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load
    Label1.Text = "Hello World; the time is now " & DateTime.Now.ToString()
End Sub
```

### C#

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "Hello World; the time is now " + DateTime.Now.ToString();
}
```

注意，一旦输入了表示 **Label1** 的 **L**，就会看到一个选项列表。这是 VS 的 **IntelliSense** 功能，它

是一个不错的工具，有助于快速编写代码。不需要输入整个单词 `Label1`，只需要简单地输入字母 `L` 或者字母 `La`，然后就可以从列表中选择正确的项，如图 2-12 所示。

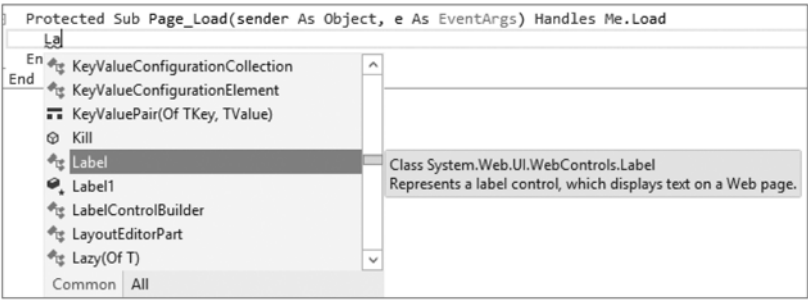


图 2-12

要完成选中的单词，可以按下 `Enter` 或者 `Tab` 键，甚至可以按下句点键。在后一种情况下，会立即看到另一个列表，它允许通过输入前几个字母来挑选单词 `Text`，然后按下 `Tab` 或 `Enter` 键完成这个单词。这是一个效率很高的工具，因为只需要稍微击几次键就能写出完整的代码。在许多其他的文件类型中也能使用 `IntelliSense` 功能，包括 `ASPX`、`HTML`、`CSS`、`JavaScript` 和 `XML`。在许多情况下，如果你开始输入，就会自动弹出这个带有选项的列表。如果没有弹出，可以按下 `Ctrl+空格` 键来激活它。如果该列表覆盖了代码窗口中的部分代码，那么一直按下 `Ctrl` 键就可以使窗口变得透明。

(7) 右击 `Solution Explorer` 中的 `CodeBehind.aspx` 页面，并选择 `View in Browser(Internet Explorer)` 命令。根据为计算机配置的默认浏览器，括号中的浏览器名称可能不同。下面只把这个菜单项表示为 `View in Browser`。

(8) 如果看到一个对话框询问是否要保存修改，单击 `Yes`，该页面就会出现在浏览器窗口中，如图 2-13 所示。

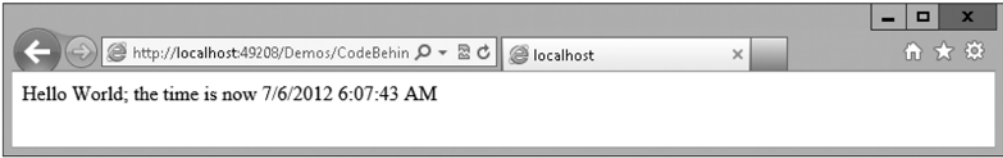


图 2-13

如果没有看到带有日期和时间的消息出现，或者在浏览器中看到了页面错误的消息，那么要确保已经将修改保存到了所有打开的页面中。要立即保存所有页面，按下 `Ctrl+Shift+S` 组合键或者单击工具栏上的 `Save All` 按钮(带有多个软盘符号的那个按钮)。此外，一定要确保输入了正确语言的代码。当创建这个新页面时，选择一种能应用到整个页面的编程语言。不能在一个页面中混合使用两种语言。因此如果一开始使用的是 `Visual C#` 页面，则一定要输入第(6)步中的 `C#` 代码段。

(9) 用内联代码配置页面的方法非常类似。首先向 `Demos` 文件夹中添加一个新的 `Web` 窗体文件。将其命名为 `CodeInline.aspx`，并确保取消选中 `Place Code in Separate File` 选项。

(10) 与第(3)、(4)、(5)步一样，将页面切换到 `Design` 视图中，拖动一个 `label` 控件到 `<div>` 元素内，然后在页面上双击现在包含该 `label` 的 `<div>` 之外的某处。VS 现在不是打开一个 `Code Behind` 文件，而是将页面切换到 `Markup` 视图中，并直接在页面中添加 `Page_Load` 代码。

(11) 在 VS 插入的代码块中的空行上, 输入在本练习第(6)步中突出显示的代码行。确保使用正确的编程语言。最后在.aspx 文件上应看到下列代码。

#### VB.NET

```
<script runat="server">
    Protected Sub Page_Load(sender As Object, e As EventArgs)
        Label1.Text = "Hello World; the time is now " & DateTime.Now.ToString()
    End Sub
</script>
```

#### C#

```
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "Hello World; the time is now " + DateTime.Now.ToString();
    }
</script>
```

(12) 右击 Solution Explorer 中的页面, 并选择 View in Browser 命令。也可以按下 Ctrl+F5 组合键在浏览器中打开页面。你应看到与第(7)步相同的页面。

#### 工作原理

在运行时, 带内联代码的页面的行为与使用 Code Behind 文件的页面相同。在这两种情况下, ASP.NET 运行库看到 Page\_Load 代码, 并执行它在其中发现的任何代码。在这个“试一试”练习中, 这意味着将 Label1 的 Text 设置为欢迎消息和今天的日期与时间。这两个选项的最大区别是代码的存储位置。对于带内联代码的页面, 所有的代码和标记都存储在磁盘的同一个文件中, 而使用 Code Behind 时, 所编写的 VB 或 C#代码层次在以 Web 窗体命名的另一个文件中。

因为使用 Code Behind 文件的页面更容易管理, 所以 Planet Wrox 网站将一直使用它。

注意在本例中, C#代码看起来与 VB.NET 代码非常相似。设置 Label 的文本的代码在两种语言中几乎相同。一个区别在于 VB.NET 用&符号把两块文本连在一起, 而 C#用的是加号(+)字符。还可以在 VB.NET 中使用加号将字符串连接起来, 但是会遇到第 5 章中将要介绍的一些警告。另一个区别是在 C#中所有的代码行都必须以分号(;)结尾, 表示代码单元的结束, 而 Visual Basic 用的是换行符。如果希望将使用 Visual Basic 语言编写的一个长的代码行分成多行, 则需要使用下划线字符。在之前的版本中, VB.NET 在很多不同的地方都需要使用下划线。但是, 在 Visual Basic 的最近版本中, 该语言的设计者大大减少了需要使用下划线的次数。

如果希望恰好在 Handles 关键字前将代码分为多行, 则需要使用下划线, 请参看练习第(6)步中的代码段:

```
Protected Sub Page_Load(sender As Object, e As EventArgs) _
    Handles Me.Load
    Label1.Text = "Hello World; the time is now " & DateTime.Now.ToString()
End Sub
```

注意在页面中, Visual Basic 示例第一行的末尾没有下划线(\_)。这里添加它是因为本书的页面宽



度不够在一行中显示整个代码语句。在本书其余部分的其他 Visual Basic 示例中你将会看到更多这样的下划线。如果你决定手动输入下划线来使代码更可读,则不要忘记在实际下划线之前多输入一个空格,否则代码将无法运行。

在 C#中就不需要这个字符,因为该语言本身允许按下 Enter 键来中断太长的行。这是因为 C#使用分号作为行结束符,而不是在源代码中使用换行符来表示结束。

可以通过右击 View in Browser 选项或者按下 Ctrl+F5 组合键,在浏览器中打开页面。使用 View in Browser 选项,总是可以打开右击的页面。使用 Ctrl+F5 快捷键,可以打开在文档窗口中当前活动文档的页面,即当前在 Solution Explorer 中选中的页面,或者打开设置为 Web 站点的起始页(start page)的文件。此外,所有打开的文件都会自动保存,并且在浏览器中打开请求页面之前会检查站点的错误。

在 Solution Explorer 中右击并选择 Set As Start Page 命令,可以将一个页面指定为起始页。如果想在后面的阶段控制这个行为,则右击 Solution Explorer 中的 Web 站点,并选择 Property Pages 命令。在 Start Options 类别中,可以指明希望打开的当前活动页面,也可以指定一个特定页面,如图 2-14 所示。

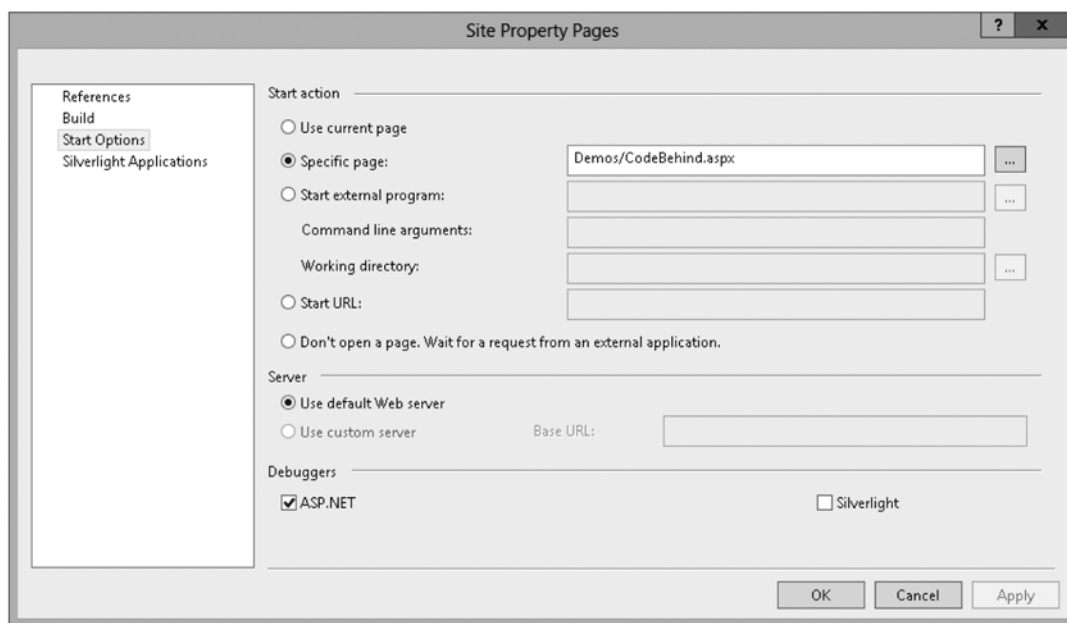


图 2-14

在上面的练习中,介绍了如何添加一个包含简单 Label 控件的页面。此外,你还看到了如何编写一些代码,来更新有今天日期与时间的 label。暂时可以忽略该代码,这里提到它只是为了演示 Code Behind 代码和内联代码之间的区别。在第 5 章将会介绍用 Visual Basic 和 C#编程的更多知识。

为了制作一个引人注目的页面,需要的内容显然远远不止一个显示今天日期和时间的简单 Label 控件。下一节将介绍如何向页面中添加内容和 HTML,以及如何定制它的样式和格式。

### 2.3.3 向页面添加标记

有几种方式可以向页面中添加 HTML 和其他标记。首先,可以简单地在 Markup 视图窗口中输

入。然而，这并不总是最好的选择，因为它会强制手动输入大量代码。要更容易地向页面中插入新 HTML 并向其应用格式，Design 视图窗口提供了几个有用的工具。这些工具包括 Formatting 工具栏、菜单项 Format 和 Table。要使这些工具起作用，需要在 Design 视图中包含它们的文档。如果是在 Split 视图模式下工作，则一定要确保焦点在 Design 视图部分，否则就会发现大多数工具都不可用。

### 1. 插入和格式化文本

在 Design 视图和 Markup 视图中都可以输入文本。只需要将光标放在所需的位置并开始输入即可。当切换到 Design 视图时，Formatting 工具栏就变得可用，其中的选项如图 2-15 所示。

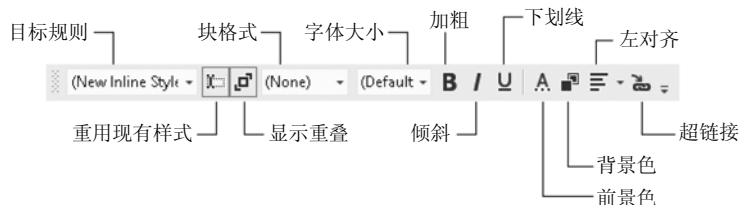


图 2-15

这个工具栏上的许多按钮的功能与其他编辑环境中的相应按钮功能完全相同。例如，B 按钮用粗体字格式化文本。类似地，I 和 U 按钮分别使字体倾斜和添加下划线。使用块格式下拉列表可以插入 HTML 元素，如<p>表示段落，<h1>~<h6>表示标题，<ul>、<ol>、<li>表示列表。可以直接从这个下拉列表中选择一项插入页面中，或者也可以先选择一些文本，然后从列表中选择适当的块元素，并把选中的文本包括在标记内。

在下面的“试一试”练习中，将看到如何使用这些工具来创建 Planet Wrox Web 站点的主页。

#### 试一试

#### 添加带格式的文本

在本“试一试”练习中，将创建一个名为 Default.aspx 的 Web 窗体并向其添加一些基础内容。

(1) 通过 Add New Item 命令在站点对话框的根位置添加一个新的 Web 窗体，并将其命名为 Default.aspx。确保选中的是 Place Code in Separate File 选项并单击 Add 按钮。使用文档窗口下方的 Design 按钮切换到 Design 视图。

(2) 在虚线矩形内单击，直到看到表示<div>元素当前是活动状态的“浮起”。与此同时，代码窗口下方的标记导航器应突出显示最后一个块，它上面有文本<div>，如图 2-16 所示。



图 2-16

(3) 输入“Hi there visitor and welcome to Planet Wrox”，并使用鼠标突出显示这段文字。从块格式下拉列表(参见图 2-15)中选择 Heading 1 <h1>。注意这时有一个带有文本 h1 的小浮块出现在这段文本的正上方，表示 VS 自动创建了一个标题。图 2-17 显示了有<h1>元素的 Design 视图。



图 2-17

(4) 把光标放在单词 Wrox 后的标题末尾，并按下 Enter 键。这时会插入一个新段(由一个小浮起表示，上面有字母 p)，因此就可以直接开始输入了。

(5) 输入图 2-18 中显示的文本(或是自己构思的文本)，对 Planet Wrox 站点的访问者发出欢迎消息。注意一输入逗号，文本 “www.PlanetWrox.com” 就变成了蓝色，以表明 VS 已经将其看作是一个 Web 地址，并且将其变成了一个链接。可以按 Enter 键来开始一个新的段落。选择文本 “paying a visit”，单击 Formatting 工具栏上的 Foreground Color 按钮，并在出现的对话框中选择一个不同的颜色。然后选择一些别的文本，如 “reviews and concert pictures”，并单击 Bold 按钮。此时，Design 视图的显示应如图 2-18 所示。



图 2-18

此主页的代码现在应如下面所示(代码已经略微重格式化，以适合本书的空间):

```
<div>
  <h1>Hi there visitor and welcome to Planet Wrox</h1>
  <p>
    We're glad you're
    <span class="auto-style1">paying a visit</span> to
    <a href="http://www.PlanetWrox.com">www.PlanetWrox.com</a>,
    the coolest music community site on the Internet.
  </p>
  <p>
    Feel free to have a look around; there are lots of interesting
    <strong>reviews and concert pictures</strong> to be found here.
  </p>
</div>
```

在文件的顶部应有一个<style>元素，如下所述。

(6) 要在浏览器中打开该页面，可以按下 Ctrl+F5 组合键，或者在 Solution Explorer 中右击该页面，然后选择 View in Browser 命令。

## 工作原理

当使用各种 Formatting 工具栏按钮时，如 Foreground Color，VS 会自动插入适当的 HTML 和 CSS 代码。例如，当单击 B 按钮时，VS 会在选中的文本两头插入一对<strong>标记。当单击 I 按钮时，它就会添加一对<em>标记使文本倾斜。在这个练习中，当改变文本颜色时，VS 还会插入一个 class 特性(前面的代码示例中出现过)，指向一个名为 auto-style1 的类。这种样式的代码将添加到文件的上方，代码应如下所示：

```
<style type="text/css">
    .auto-style1
    {
        color: #FF0000;
    }
</style>
```

如果选择的是不同的颜色，那么代码可能和上面略有不同。在这里所看到的代码将在第3章解释。目前，只需要记住这段代码将文本颜色设置为红色即可。如果左花括号与类在同一行上，则可以选择 Tools | Options | Text Editor | CSS | Formatting，把 Formatting Style 设置为 Expanded。这只是一种个人喜好，不会改变代码的效果。

注意 VS 用与 HTML 兼容的变体&#39;替换欢迎消息中的“we’re”里面的单引号(’)。使用这种代码可能向页面中插入浏览器显示起来有些麻烦的字符，或者在 HTML 中有特殊含义的字符，如&，它就写作&amp;。当在 Design 视图输入文本时，VS 会自动插入相关字符的等价代码；然而，如果直接在 Markup 视图输入，就必须亲自替换。

如果你的代码和这里显示的不一样，不要担心。VS 中的许多设置都会影响生成的代码。

到目前为止，这些练习都是关于在页面中添加和样式化文本的。然而，VS 也允许插入其他 HTML 元素，比如表和项目符号。下一节将说明它们的工作原理。

## 2. 添加表与其他标记

如果需要显示结构化或重复的数据，如购物车中的产品清单、相册中的照片或者表单中的输入控件，则 HTML 表相当好用。关于是否要用表来布置页面，网上也有许多争论。例如，如果页面中包含一个带 logo 的标题，一个主内容区，下方还有一个页脚，就可以使用一个 3 行的表来完成它。总体而言，为了这个目的而使用表并不太好，因为它们会向页面中添加大量无关的标记，而且常常难以维护。此外，使用 CSS 往往可以达到同样的效果，这将在第3章中讲解。尽管表可能带来这些缺点，但是在显示表格或其他结构化信息时，表仍然是 HTML 工具箱中的重要工具。

### 试一试

### 使用 Format 和 Table 菜单

本练习将指出如何用 Table 菜单向页面中添加表，以及如何添加行和列。此外，还将说明如何添加其他结构化元素，比如项目符号列表。

(1) 在 Demos 文件夹中，创建一个新的 Web 窗体，名为 TableDemo.aspx。通过选中 Place Code in Separate File 选项确保它使用 Code Behind 文件。

(2) 将页面切换到 Design 视图，在页面中表示标准<div>标记的虚线框内单击，并选择 Table | Insert Table 命令。这时会出现 Insert Table 对话框，如图 2-19 所示。

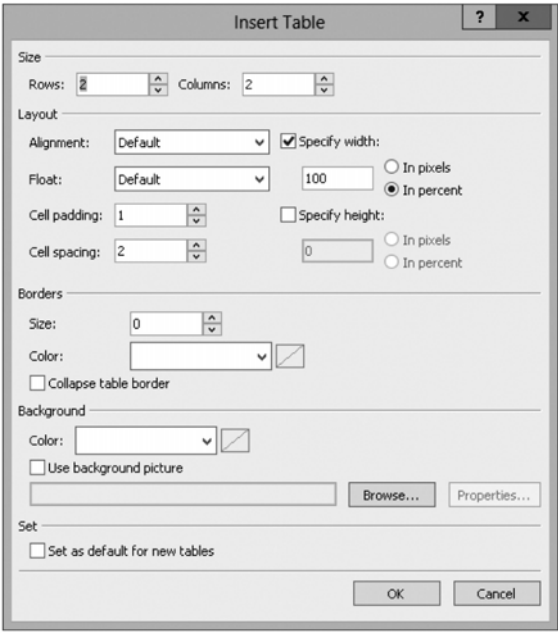


图 2-19

(3) 将 Rows 设置为 3，Columns 设置为 2。保持其他设定项为默认值，并单击 OK 按钮。这个表就会插入到页面中。

(4) 如果只看到一个表单元格，而不是有 3 行 2 列的完整表，就需要启用表的 Visual Aid。为此，从主菜单中选择 View | Visual Aids | Visible Borders 命令来打开边框。此时 Design 视图应如图 2-20 所示。

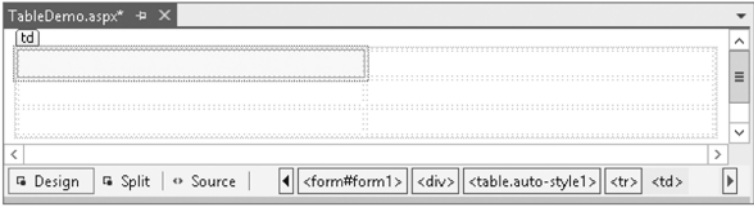


图 2-20

(5) 把表中第一个单元格的右边界拖到左边。这时就可以看到一个可视化指示器显示了单元格的宽度。把它往左边拖直到宽度为 200 像素，如图 2-21 所示。

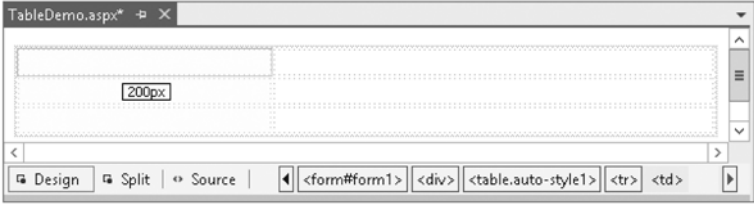


图 2-21

(6) 要将更多行或列添加到表中，可以右击现有单元格。从出现的弹出菜单中选择 Insert 命令在不同位置添加额外的行或列。类似地，可以用 Delete、Modify 和 Select 选项删除行或列、合并单元

格或进行选择。对于本练习，不需要添加额外的行或列，但如果已经添加了也没有关系。

(7) 把光标放在第一行的第一个单元格中，输入 **Bulleted List**。

(8) 用鼠标把光标放在第一行的第二个单元格中，也可以按下 **Tab** 键，将光标移动到下一个单元格中，从 **Format** 菜单中选择 **Bullets and Numbering** 命令。

(9) 切换到 **Plain Bullets** 选项卡，单击带圆形实心项目符号的图片(如图 2-22 所示)，并单击 **OK** 按钮。

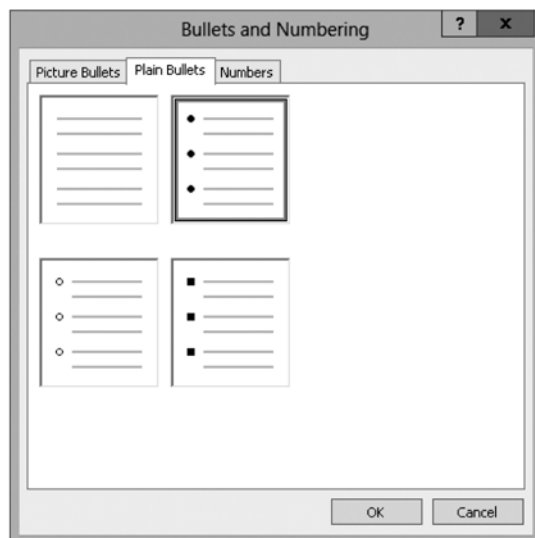


图 2-22

(10) 输入一些文本，例如喜欢的音乐类型(Punk、Rock 及 Techno 等)，然后按下 **Enter** 键。VS 会自动插入一个新的项目符号，从而可以继续向列表中添加新的项。再添加两种类型，这样就有 3 个项目符号了。

(11) 重复第(7)~(10)步，但这次创建一个编号列表。首先，在第二行的第一个单元格中输入 **Numbered List**，然后将光标放到同一行的第二个单元格中，并选择 **Format | Bullets and Numbering** 命令。切换到 **Numbers** 选项卡(如图 2-22 所示，位于 **Plain Bullets** 后边)，并单击第一行中的第二个图，它显示了一个标准的编号列表，然后单击 **OK** 按钮。为列表输入一些项，每输入一个项就按一次 **Enter** 键。

(12) 按下 **Ctrl+F5** 组合键在浏览器中打开页面。这时应看到一个如图 2-23 所示的屏幕。

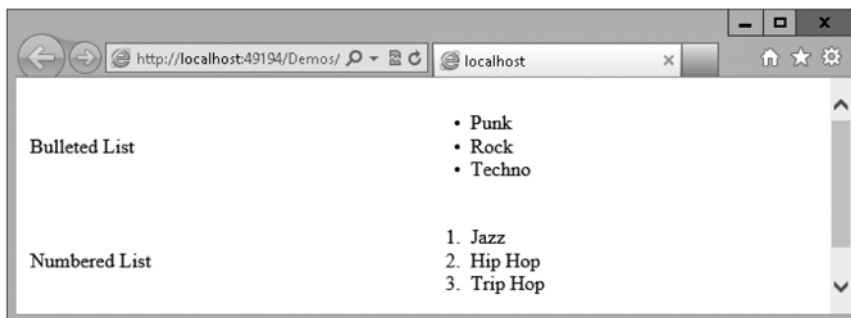


图 2-23

## 工作原理

当通过可用的菜单可视化地插入了一些页面元素,比如表或列表时,VS 就会在 Markup 视图中插入必需的标记。当插入一个表时,VS 会添加一个<table>元素和几个<tr>及<td>元素,来分别定义行和单元格。它还应用了一个指向 CSS 样式的 class 特性来控制表的宽度。如果把列的宽度设为 200 像素,它还会为<td>元素创建另一种样式。类似地,当插入一个列表时,VS 会插入一个<ol>元素表示编号或有序列表,插入一个<ul>元素表示项目符号列表或无序列表。在这些元素内,用<li>元素定义列表中的每个项。

除了到目前为止所看到的 HTML 标记外,还需要了解另一个重要标记:<a>标记,它用来在页面之间创建链接。

### 2.3.4 连接页面

Web 站点的一个重要功能是链接站点中的页面。链接允许访问者从一个页面进入同一站点中的另一个页面,或者进入到 Internet 上完全不同站点中的页面。在页面之间创建链接的方式有几种,包括:

- HTML <a>元素,已在本章解释。
- 使用<asp:HyperLink>控件,将在第 7 章讨论。
- 通过代码编程,这将在本书后面讨论。

下面的练习显示了如何轻松地从一个页面链接到另一个页面。

#### 试一试

#### 链接页面

在本“试一试”练习中,将通过添加链接到另一个页面的文本来修改早先创建的页面 TableDemo.aspx。一旦在浏览器中运行该页面并单击那个链接,新页面就会替换老页面。

(1) 从 Demos 文件夹中打开页面 TableDemo.aspx。

(2) 如有必要,切换到 Design 视图。

(3) 在第三行的第一个单元格中,输入文本 Link。

(4) 在同一行的第二个单元格中,输入文本 Go to the homepage of Planet Wrox,并用鼠标突出显示它。

(5) 在 Formatting 工具栏上,单击 Convert to HyperLink 按钮。它是工具栏上的最后一个按钮,上面有一个链接图标和一个小箭头。

(6) 在出现的对话框中,单击 Browse 按钮,并定位到站点根文件中的 Default.aspx 页面,然后单击 OK 按钮。再次单击 OK 按钮关闭 Hyperlink 对话框。此时页面的 Design 视图应如图 2-24 所示。

(7) 切换到 Markup 视图并注意链接的 HTML 是如何插入的:

```
<a href="../../../Default.aspx">Go to the homepage of Planet Wrox</a>
```

注意 href 特性指向要链接到的页面。

(8) 如果要从 Markup 视图中修改要链接的页面,单击 href 特性的起始和结束引号之间的某处,并按下 Ctrl+空格键。这时会弹出一个对话框,允许选择站点中的另一个页面。此外,也可以单击 Pick URL 选项,并浏览站点中某处的新页面。



图 2-24

(9) 右击 Solution Explorer 中的页面 TableDemo.aspx，并选择 View in Browser 命令。当页面完成加载后，单击 Go to the homepage of Planet Wrox 链接。请求将被发送到 Web 服务器，作为响应，现在得到了 Web 站点的主页。

### 工作原理

页面之间的链接很可能是 Web 页面中最重要的一个元素，因为它们允许在站点中的一个页面和另一个页面之间创建连接，不论那个页面是在自己的站点中，还是在 Internet 上某处一个完全不同的服务器上。对于应出现在页面上某处的简单链接来说，最容易创建的是带有一个 href 特性集的 HTML `<a>` 标记。当用户单击这样一个链接时，浏览器从服务器上请求新页面并显示它。href 值中的两个点 (..) 指的是父目录。完整的 href 特性表示“返回文件夹的层次结构中的上一层，然后选择文件 Default.aspx”。在第 7 章中将会看到关于链接的更多内容及其工作原理。

注意并不只能链接到自己站点的页面。如果要链接到外部页面，只需用下面示例中显示的完整页面地址替换 href 特性即可。

```
<a href="http://www.wrox.com">Go to the Wrox homepage</a>
```

对于外部链接，重要的是要包括 http:// 前缀。否则浏览器就会在 Web 站点上寻找名为 www.wrox.com 的文件或文件夹。

第 3 章将用到本章关于页面创建与格式化的知识来处理用 CSS 设计 Web 页面的事务。

除了可视化工具(如 Formatting 工具栏和 Table 菜单)外，Visual Studio 还有另一种在页面中快速插入代码的好方法：代码段。代码段允许只使用很少的击键次数，就可以向页面中插入大的代码块。在第 3 章中将会看到这种方式。

## 2.4 使用 Web 窗体的实用提示

下面是使用 Web 窗体的一些提示：

- 总是试着使用 Code Behind 模式窗体而不是内联代码设计 Web 窗体。虽然一开始你可能不会注意到两者之间的明显区别，但是随着站点和页面的扩大，你将会发现使用代码与标记分离的页面会轻松得多。



- 花一些时间熟悉 **Format** 和 **Table** 菜单中不同的菜单项。它们中的大部分都会生成可以插入到页面中的 **HTML** 元素。看一下自动生成的 **HTML** 元素与特性，并试着在代码中直接修改它们，然后再通过菜单和工具栏做一遍。通过这种方式，可以很好地了解各种可用的标记及它们的行为。
- 试验连接站点页面的链接。注意根据要链接页面的位置，VS 会创建不同的链接。第 7 章会更详细地介绍链接和定位站点中的页面的各种方式。

## 2.5 本章小结

本章介绍了一些帮助构建可维护和结构化的 ASP.NET Web 站点的重要主题。了解不同项目类型和模板之间的区别后，可以只用所需的文件构建一个 Web 项目。

这也适用于可以添加到站点中的不同文件类型。由于各个文件类型都有特定的用途，因此重要的是要知道文件的用途是什么，以及可以如何使用该文件。

在构建 ASP.NET Web 页面时会执行的一个常见操作是向页面中添加标记。如本章和第 1 章所述，标记有几种形式，包括纯 **HTML** 和 **ASP.NET** 服务器控件。知道如何用 VS 提供的大量菜单选项和工具栏向页面中添加标记，对于构建漂亮的 Web 页面是至关重要的。

现在你已经对于创建和修改 Web 窗体有了扎实的理解，下一步将了解如何把那些只有几个控件的黑白页面变成有吸引力的 Web 页面。第 3 章将介绍如何使用 VS 中的许多 CSS 工具来创建所需的效果。

## 2.6 练习

1. 指出 Web Files 类别中 3 个可以添加到站点中的重要文件。描述各文件的用途。
2. 如何使 Web 页面中的一块文本既加粗又倾斜？最终的 **HTML** 是怎样的？
3. 指出在 VS 中向 ASP.NET Web 站点添加现有文件的 3 种不同方式。
4. VS 为 ASPX 页面提供了哪些不同的视图？VS 中有没有提供别的视图？

练习的答案见附录 A。

### 本章要点回顾

Code Behind	一种在独立的代码文件中存储服务器端代码的页面模式
Design 视图	提供页面的图形表示
File Types	ASP.NET 支持多种不同的文件类型，包括 Web Forms(.aspx)，母版页(.master)，CSS 文件(.css)，JavaScript(.js)以及 SQL Server 数据库(.mdf)
内联代码	一种页面模型，将服务器端代码与标记存储在相同的文件中
Markup 视图	允许查看页面标记
项目模板	通过针对特定的应用场合建立站点，启动 Web 开发
项目类型	Visual Studio 提供了两种项目类型：Web Application Project 和 Web Site Project
Split 视图	允许同时查看 Markup 视图和 Design 视图
Web 窗体	在客户端上显示 Web 站点的用户界面

# 第 3 章

## 设计 Web 页面

### 本章要点

---

- CSS 的定义及需要 CSS 的原因
- CSS 语言简介及如何编写 CSS
- 向 ASP.NET 页面和外部文件添加 CSS 代码的各种不同方法
- VS 提供的大量快速编写 CSS 的工具

在前两章中创建的页面看起来相当单调且无趣。那是因为它们缺少样式化信息，并默认应用了浏览器的标准布局。要把页面打扮得漂漂亮亮的，需要使用某种方法来改变它们在浏览器中的表现 (presentation)。最常用的方法是使用 CSS (Cascading Style Sheet, 层叠样式表) 语言。CSS 是在 Web (包括 ASP.NET Web 页面) 上格式化和设计信息的实际语言。使用 CSS 可以快速地改变 Web 页面的外观，创建你自己设计的或是公司统一要求的完美外观。

对使用 CSS 的支持最初在 Visual Web Developer (VWD) 2008 中添加，VWD 2008 是 VS Express 2012 for Web 的一个前任。在 Visual Studio 的以前版本 VWD 2010 中进一步提升了这个支持。新的 VS 2012 在它的基础上从很多方面提高了对 CSS 的支持，使 CSS 编辑器称为 VS 中的优秀成员，该编辑器的功能类似于其他语言，如 C# 和 VB。CSS 工具有助于可视化地创建 CSS 代码，使页面的样式化更加容易，而不需要知道或记得 CSS 的所有细枝末节。

要了解 CSS 在 ASP.NET Web 站点中的关联性和必要性，需要先了解 HTML 的缺点。下一节将指出纯 HTML 表现存在的问题，以及 CSS 如何克服这些问题。

### 3.1 需要 CSS 的原因

从 Internet 出现伊始，Web 页面主要由文本和图像组成。文本是使用纯 HTML 格式化的，用 `<b>` 这样的标记使文本加粗，并用 `<font>` 标记影响字体、大小和颜色。Web 开发人员很快就发现他们需要更强大的功能来格式化页面，因此诞生了 CSS 以弥补 HTML 在样式方面的缺陷。

### 3.1.1 HTML 格式化的问题

使用 HTML 进行格式化的问题之一是它提供的样式化页面的选项很有限。可以用<i>、<b>及<font>这样的元素来改变文本的外观，用 bgcolor 这样的特性来改变 HTML 元素的背景颜色。还有几个特性可用来改变链接出现在页面中的方式。

显然，这个功能集不足以创建符合用户期望与需求的生动 Web 页面。

HTML 影响 Web 页面构建的另一个问题是样式化信息应用到页面的方式。在设计时，HTML 会强制要求在 HTML 文档中嵌入格式化信息，使得以后难以对设计进行重用或修改。如下面这个示例：

```
<p><font face="Arial" color="red" size="+1">  
  This is red text in an Arial type face and slightly larger than the default text.  
</font></p>
```

这段代码的问题在于实际数据(<p>元素中的文本)与表现(在本例中是用<font>标记格式化的文本)混淆在一起。理想情况下，这两者应当分开，以便各自能方便地修改而不会互相影响。

假设在站点的各个页面中用<p>和<font>元素来标识第一段。在决定把字体的颜色从红色改为深蓝色时，会发生什么状况呢？或者，如果公司统一要求使用 Verdana 字体而不是 Arial 字体时，会出现什么情况呢？结论是在做必需的修改时要访问站点的每个页面。

除了维护性问题外，HTML 格式化的另一个问题是，在用户的浏览器中不能轻松地在运行时修改格式。至于上面代码段中的 HTML，没有什么方法可以让访问者修改字体大小或颜色这样的属性，但视觉有障碍的人常常有这样的要求。如果想给访问者提供另一种采用较大的字体和不同颜色的版本，就需要为原始页面创建一个副本，再进行必要的修改。

HTML 格式化的最后一个问题是，页面中的附加标记大大增加了页面的大小。这样，由于需要从 Web 站点中的各个页面上下载信息，下载和显示就会变慢。而且，当需要滚动大型的 HTML 文件来查找需要的内容时，页面也会变得难以维护。

简言之，使用 HTML 格式化存在以下这些问题：

- 它的有限功能集远远满足不了页面的格式化需求。
- 数据与表现混合在相同的文件中。
- HTML 无法在浏览器中于运行时轻松地切换格式。
- 必需的格式化标记与特性使页面更大，因此加载、显示和维护更慢。

幸运的是，CSS 能够解决所有这些问题。

### 3.1.2 CSS 如何解决格式化问题

CSS 几乎能在所有可能的方面格式化 Web 页面。它提供了一套丰富的选项，可以修改 Web 页面的各个细微方面，包括字体(大小、颜色、字体等)、颜色和背景色、围绕 HTML 元素的边框、元素在页面中的位置以及其他很多方面。当今所有的主流浏览器都能接受 CSS，它就是 Web 页面可视化表现的语言，而且在 Web 开发人员中非常流行。

CSS 允许在外部文件中定义所有的格式化信息，从而解决了数据与表现混在一起的问题。然后，ASPX 或 HTML 页面就能引用这些文件，并且浏览器会应用正确的样式。有了这样的分离，HTML 文档中就仅含有要显示的内容，而 CSS 文件会仅定义显示的方式。因而，可以修改或替换这两个文档之一，而让另一个文档保持不变。此外，CSS 可以直接放在 HTML 或 ASPX 页面中，这样就可以

添加真正必要的 CSS 小代码块。直接把 CSS 放在 HTML 或 ASPX 页面上时要谨慎，因为那样就不再能从一个集中的地方控制样式信息了。

由于所有的 CSS 代码可以放在单独的文件中，因此很容易让用户在不同的样式之间选择，例如，采用较大字体的样式。可以创建外部样式表的一个副本来进行必要的修改，然后向用户提供这个可选的样式表。在第 6 章讨论 ASP.NET 主题时将介绍它的工作原理。

独立样式表文件的另一个好处是它降低了站点的带宽需求。由于样式表不会随着每个请求而改变，因此在第一次下载时浏览器会保存样式表的一个本地副本。从那时起，它就会采用这个缓存副本，而不是一再地从服务器上请求它。有时当浏览器还没有下载修改过的最新的 CSS 文件时，这种缓存就能派上用场。如果发现浏览器没有显示对 CSS 文件所做的修改，请在浏览器中(不是在 VS 中)使用 Ctrl+F5 或 Ctrl+R 键，从服务器中获得一个刷新的副本。

上面介绍了 CSS 之所以重要的原因，下面介绍它的呈现方式以及如何使用它。

## 3.2 CSS 简介

就语法而言，CSS 是一种容易学习的语言。它的“语法”仅由几个概念组成，使得它相当容易入门。CSS 的难点在于所有主流浏览器呈现页面的方式。尽管实际上每种现代桌面浏览器都能够理解 CSS，但当根据 CSS 标准显示页面时，它们都有各自的“怪癖”。这个标准是由提出 HTML 标准的同一个组织 W3C(World Wide Web Consortium)提出的，它出现过 3 个版本：1.0、2.1 和 3.0。在这 3 个版本中，2.1 版本是现在人们最普遍接受的，它包含了版本 1.0 中的所有内容，但是在其基础上又增加了大量功能。这个版本也是 VS 默认使用和生成的版本。版本 3.0 目前正在开发中，预计不久的将来，主流浏览器就会大力支持它。

在学习 CSS 的实际语法之前，最好先看一个示例。在下面的“试一试”练习中，将会写一个简单的 ASPX 页面，该页面包含了一些 CSS 以格式化页面内容。这个练习有助于理解 CSS 语言，在练习之后接着将会全面讨论这个语言。

### 试一试

### 编写第一个 CSS

在本“试一试”练习中将编写一些 CSS 来修改一个标题和两个段落的外观。首先手动为页面编写代码；本章的后半部分会介绍如何在 VS 中使用可用的 CSS 工具。

(1) 在 Planet Wrox 项目中的 Demos 文件夹中新建一个 Web 窗体，命名为 CssDemo.aspx。对于本练习来说，选择内联代码或 Code Behind 都没有关系。

(2) 确保该页面在 Markup 视图中，然后在源代码中定位</title>结束标记。把光标放在这一行的最后，按下 Enter 键，在<title>和<head>标记中间空出一行。在这一新行上输入 style 然后按下 Tab 键。VS 将会自动完成<style>元素的添加。再按两次 Enter 键，在标记之间留出一些空间。最终可以得到如下的粗体代码：

```
<title></title>
<style type="text/css">

</style>
</head>
```



**注意：**代码完成功能使用代码段把一部分代码(如<style>元素)和一个标识符(在本例中是 style)关联起来。这个代码段对于快速插入一段代码非常有用，只需要输入一个简写的标识符即可。有可用的许多代码段，在学习本书时，会在适当的地方指出它们。

除了使用样式代码段，还可以手动输入整个代码。注意，一旦输入了起始尖括号(<)，就会弹出一个列表供你选择<style>标记。type 特性也是如此，只要输入字母 ty，列表中就会预选了 type 特性。只需要按下 Tab 或 Enter 键完成单词即可。此外，特性值 text/css 也有这样的帮助功能，只需要在列表中选择它，并按下 Tab 或 Enter 键，它的值就会自动插入，而且带有完好的双引号。

(3) 接下来，在起始和结束<style>标记之间，输入下面突出显示的 CSS 代码：

```
<style type="text/css">
  h1
  {
    font-size: 20px;
    color: Green;
  }

  p
  {
    color: Blue;
    font-style: italic;
  }

  .RightAligned
  {
    text-align: right;
  }
</style>
```

输入这段代码时要非常小心，因为 CSS 的语法相当严格。列表中的第一项是样式化一级标题的 h1 标记，因此它的大小为 20 像素，并显示为绿色字体。注意 font-size 和 20px 之间用冒号隔开，且该行以分号结束。

列表中的第二项仅含有字母 p，它定义了页面中所有<p>元素的外观。

最后一项有一个前缀句点(.)，后面跟着文本 RightAligned。这个项用来右对齐页面中的文本。因为 CSS 是区分大小写的，所以必须使用大写的 R 和 A，才能正确输入它们，否则 CSS 代码就不匹配下一步中的 HTML。

注意一旦在 color 属性后面输入#，就弹出一个颜色选择器，帮助选择颜色，如图 3-1 所示。现在，只需要按下 Esc 键关闭颜色选择器，手动完成代码的输入。本章后面将介绍颜色选择器。

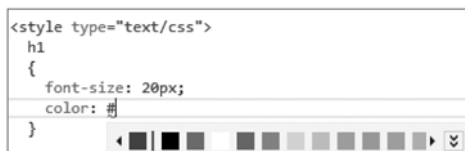


图 3-1

(4) 把页面稍微向下滚动一点，直至看到起始<div>标记。在这个标记后面输入下面的突出显示

代码:

```
<div>
  <h1>Welcome to this CSS Demo page</h1>
  <p>CSS makes it super easy to style your pages.</p>
  <p class="RightAligned">
    With very little code, you can quickly change the looks of a page.
  </p>
</div>
```

如果不想直接输入这段代码,也可以在 Design 视图中,使用 Formatting 工具栏来创建像<h1>和<p>这样的元素。在这里,需要切换到 Markup 视图来输入 class="RightAligned",但是在本章以后的练习中,就可以看到如何让 IDE 自动写出这个代码。

(5) 如果切换到 Design 视图(或者 Split 视图)中,将看到设计器会用在页面的<style>元素中定义的格式显示文本。图 3-2 显示了 Split 视图中的页面,这里可以同时看到代码与设计效果。

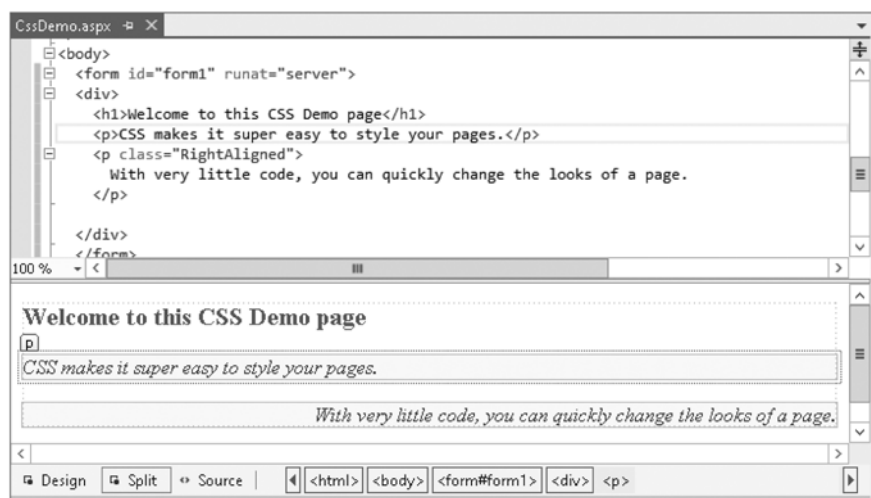


图 3-2

尽管本书是黑白版的,不太容易看出不同的字体颜色,但是从图 3-2 中可以清楚地看出<h1>的字体更大。图 3-2 还表明,所有段落现在都是用斜体字显示的。最后,可以看到最后的段落会与窗口右对齐,这是因为这个标记上的 class 特性设置为 RightAligned。



**常见错误:** 如果没有看到最后一段贴合在文档窗口的右边,请确保在<style>标记中和在 class 特性中输入的 RightAligned 完全一样。因为 CSS 是区分大小写的,所以 RightAligned 和 rightaligned 之间是有很大的区别的。

(6) 若要在浏览器中显示 CssDemo.aspx 页面,则按下 Ctrl+F5 组合键。在浏览器中看到的页面与在 VS 的 Design 视图中看到的预览完全相同。

### 工作原理

虽然在本练习中输入的代码相当简单,但是在后台浏览器(以及 VS 的 Design 视图)中却要

少工作才能实现它。首先，要向页面的<head>部分添加一些样式：

```
<style type="text/css">
  h1
  {
    font-size: 20px;
    color: Green;
  }

  ...
</style>
```

<style>标记用来包装一个嵌在页面中的样式表，它的 type 特性设置为 text/css。这个 text/css 值是<style>块目前唯一可用的类型，它告诉浏览器，把后面的代码解释为 CSS。

在<style>标记之间从 h1 开始到闭合花括号的代码块称为规则集，或者仅称为规则。这段代码段中的规则定义了页面中所有<h1>元素的外观。代码块上方的 h1 称为选择器(selector)，用来表示应当向什么元素应用该格式化信息。在这种情况下，选择器直接映射到 HTML 元素上。还有很多其他选择器可以使用(在下一节将会介绍)。图 3-3 显示了元素相互之间的关系。

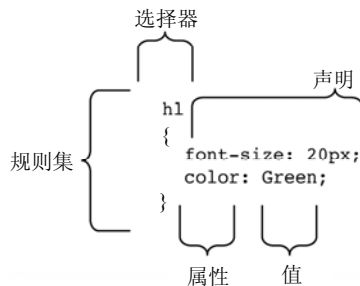


图 3-3

在花括号之间可以看到要应用到标题的样式信息。花括号中的每一行称为声明(declaration)。声明依次由一个属性、一个冒号以及一个值组成。声明末尾的分号(;)把它与下一个声明分隔开，除了规则集中的最后一个声明以外，其他所有的声明都需要使用分号隔开。然而，为了一致性，最好把分号添加到所有声明上，在本书的其余部分就是这么做的。

当浏览器加载这个页面时，它也读取在<style>标记之间定义的样式。然后，每当它遇到一个匹配该选择器的 HTML 元素时，就会向那个元素应用 CSS 规则。因此，对于<h1>和<p>元素，就会应用它们各自的规则。这样就导致标题的字体颜色变成绿色，字体更大；而段落会变成蓝色，字体为斜体。

但是最后一段为什么变成了蓝色并且右对齐呢？在 CSS 中，可以有来自不同源代码的规则。最后一个<p>元素从样式定义的 p 选择器中获得它的样式信息。因此，p 规则规定了段落使用蓝色且倾斜的字体。然而，它还定义了一个类。这个类为 RightAligned，链接到样式表中的一个 Class 选择器.RightAligned(注意前面的句点)上，它使文本与窗口右边对齐。下一节将介绍 Class 和其他选择器。在末尾，最后一个<p>元素同时从两个选择器中得到它的规则。可以定义自己的类并为其指定名称(如 RightAligned 类)，从而以更加灵活的方式设计页面和元素。

下一节将进一步研究 CSS 的语法，给出关于选择器、属性和值的详细介绍。

### 3.2.1 CSS 语言

如上一个“试一试”练习所述，层叠样式表实际上是规则的集合。规则是选择器和一个或多个声明的组合，而声明又可以分解为属性和值。你很可能对上面介绍的这些术语感到迷惑，因此在下一节中将再次介绍其中的大部分术语，并提供详细的解释和代码示例，以显示它们的用途和工作方式。

### 3.2.2 样式表

样式表中包含应当应用到页面元素的所有相关样式信息。样式表最简单的形式如下所示：

```
h1
{
  color: Green;
}
```

从前面的“试一试”练习中可以看出，样式表也可以包含多个规则。同时，每个规则可以包含多个声明，允许将它们组合在一个选择器下，比如：

```
h1
{
  font-size: 20px;
  color: Green;
}
```

这段代码在功能上等同于：

```
h1
{
  font-size: 20px;
}
h1
{
  color: Green;
}
```

在精简形式中，两个声明组合在同一个选择器下，因此阅读、理解和维护要容易得多。我们建议尽可能采用这种语法。

要能够样式化页面上的元素，浏览器必须知道 3 件事情：

- 必须样式化页面上的什么元素？
- 必须样式化元素的什么部分？
- 希望选中元素的那部分看起来是什么样子？

这些问题的答案由选择器、属性和值给出。

#### 1. 选择器

顾名思义，选择器用来在页面内选择或指向一个或多个特定元素。有若干个不同的选择器可用，通过它们可以对要样式化的元素进行很细化的控制。选择器回答了第一个问题：必须样式化页面的什么元素？接下来将介绍 4 种最重要的选择器类型。



### 1) Universal 选择器

Universal 选择器由星号(\*)表示,它适用于页面中的所有元素。Universal 选择器可以用来进行一些全局设置,比如字体。下面的规则集将页面中所有元素的字体改为 Arial:

```
*
{
    font-family: Arial;
}
```

### 2) Type 选择器

Type 选择器允许指向一个特定类型的 HTML 元素。有了 Type 选择器,那种类型的所有 HTML 元素都会被相应地样式化。

```
h1
{
    color: Green;
}
```

这个 Type 选择器现在应用到代码中的所有<h1>元素上,并把它们设为绿色。Type 选择器是不区分大小写的,因此既可以用 h1 也可以用 H1 来表示相同的标题。但我喜欢给 Type 选择器使用全小写形式,所以本书使用这种方式。

### 3) ID 选择器

ID 选择器总是以井号(#)为前缀,允许查阅页面中的单个元素。在 HTML 或 ASPX 页面内,可以使用 id 特性给每个元素赋予一个独一无二的 ID。使用 ID 选择器,就能改变那个元素的行为,如:

```
#IntroText
{
    font-style: italic;
}
```

由于可以在站点的多个页面上重用此 ID(只必须在一个页面内独一无二),因此可以用这个规则来快速地修改在每个页面上使用一次(但是在站点中使用多于一次)的元素的外观。如下面的 HTML 代码:

```
<p id="IntroText">I am italic because I have the right ID.</p>
<p id="BodyText">I am NOT italic because I have a different ID.</p>
```

在这个示例中, #IntroText 选择器将修改第一段的字体——它有匹配的 id 特性,但是让其他段落保持不变。ID 选择器是区分大小写的,因此,要保证 id 特性和选择器总是使用相同的大小写。

注意该选择器在其名称中使用#,但在 id 特性中不使用#。

### 4) Class 选择器

Class 选择器可以通过 class 特性样式化多个 HTML 元素。当需要给若干个不相关的 HTML 元素赋予相同的格式化类型时,使用这个选择器就非常方便。下面的规则将所有 HTML 元素的 class 特性设置为 Highlight,从而将它们的文本改为红色粗体字:

```
.Highlight
{
```

```
color: Red;
font-weight: bold;
}
```

下面的代码段使用 **Highlight** 类创建 `<span>` 元素的内容，并使链接 `<a>` 显示为粗体字：

```
This is normal text but <span class="Highlight">this is Red and Bold.</span>
This is also normal text but
<a href="CssDemo.aspx" class="Highlight">this link is Red and Bold as well.</a>
```

注意这个选择器在它的名称中用了句点，但是当涉及 `class` 特性中的选择器时，不要使用这个句点。`class` 特性是非常有用的，因为它允许对多种不同用途重用一块 CSS，而不必考虑使用该类的 HTML 元素是什么。`Class` 选择器是区分大小写的，所以应确保正确输入它们(或者尽量让 **IntelliSense** 帮助从列表中选择类)。

CSS 支持很多选择器类型，以便对目标元素进行更多的控制，不过最常用的就是刚才介绍的 4 种类型。

### 5) 组合和合并选择器

CSS 还允许组合多个选择器，用逗号将它们分开。如果要向不同的元素应用相同的样式，这是很方便的。下面的规则将页面中的所有标题都改为红色：

```
h1, h2, h3, h4, h5, h6
{
    color: Red;
}
```

而且，用 CSS 也可以合并选择器，这样就能有层次地指向一个页面中的特定元素。为此，可以用一个空格分开选择器。下面的示例把属于 `id` 为 `MainContent` 的 `<section>` 元素的所有 `<p>` 元素作为目标，这个规则仅影响落在 `MainContent` 元素内的段落，而保持所有其他段落不变。另外注意，`section` 与 `#MainContent` 之间没有空格，因此选择器的该部分指向 `id` 为 `MainContent` 的 `<section>` 元素。

```
section#MainContent p
{
    font-size: 18px;
}
```

注意，合并与组合有很大的区别。组合只是一个避免反复输入相同声明的快捷键，而合并允许将文档中的特定元素作为目标。

使用合并，不只能使用 `ID` 和 `Type` 选择器，也可以通过其他选择器来使用它，如下面的示例所示：

```
section#MainContent p.Attention
{
    font-weight: bold;
}
```

这个规则会修改 `id` 为 `MainContent` 的 `<section>` 元素内类为 `Attention` 的所有段落，而不影响其他段落。下面的 HTML 代码段使用此规则显示效果：

```
<section id="MainContent">
  <p class="Attention">My class is Attention, so my text is bold.</p>
  <p>My text is not bold, as it lacks the Attention class.</p>
</section>
<p class="Attention">I am NOT bold because I don't fall within MainContent.</p>
```

在页面中应用某种样式时需要回答的第二个问题是必须样式化元素的哪一部分。它通过属性来回答。

2. 属性

属性是元素的一部分，可通过样式表修改。CSS 规范定义了一个很长的属性列表(VS 的 IntelliSense 列表显示了 100 多项)，但在大多数 Web 站点中不会用到所有项。表 3-1 列出了部分最常见的 CSS 属性，并说明了它们的应用场合。

表 3-1

| 属 性              | 说 明  | 示 例   |
|------------------|--|---|
| background-color | 指定元素的背景色或图像                                | background-color: White;  |
| background-image |  | background-image: url(Image.jpg);   |
| border           | 指定元素的边框                                    | border: 3px solid Black;  |
| color            | 修改字体颜色                                     | color: Green;   |
| display          | 修改元素的显示方式，允许隐藏或显示它们                        | display: none;<br>这种设置使元素被隐藏，不占用任何屏幕空间  |
| float            | 允许用左浮动或右浮动将元素浮动在页面上。<br>其他的内容则被放在相对的位置上    | float: left;<br>该设定使跟着一个浮动的其他内容放在元素的<br>右上角。在本章后面将介绍它的工作原理                    |
| font-family      | 修改页面上使用的字体外观                               | font-family: Arial;   |
| font-size        |  | font-size: 18px;  |
| font-style       |  | font-style: italic;   |
| font-weight      |  | font-weight: bold;  |
| height           | 设置页面中元素的高度或宽度                              | height: 100px;  |
| width            |  | width: 200px;   |
| margin           | 设置元素内部(内边距)或外部(页边距)的可用<br>空间               | padding: 0;   |
| padding          |  | margin: 20px;   |
| visibility       | 控制页面中的元素是否可见。不可见的元素仍然会<br>占用屏幕空间，只是看不到它们而已 | visibility: hidden;<br>这会使元素不可见。然而，它仍然会占用页面<br>的原始空间，就好像元素仍然在那里，只不过是<br>完全透明的 |

幸运的是，VS 会通过它的许多 CSS 工具帮助找到恰当的属性，因此不必全部记住它们。



**注意：**CSS 中可用的选择器和属性要比这里描述的多得多。要查看更多关于 CSS 的详细信息，请查看 VS 附带的 IntelliSense 列表或者访问 [www.w3schools.com/cssref/](http://www.w3schools.com/cssref/) 链接。

要使一个属性有用，需要给它赋值，这就回答了第三个问题：你希望选中元素的部分看起来是什么样的？

### 3. 值

与属性一样，值也有很多风格。可用的值取决于具体的属性。例如，`color` 特性采用表示颜色的值。它可能是颜色名称(如 `White`)，也可能是代表红、绿、蓝(RGB)成分的十六进制数(如 `#FF0000`)，还可以用 CSS `rgb` 表示法来设置。下面的示例在功能上是完全等价的，都是把 `h1` 元素设置为红色：

```
h1
{
  color: Red;
}

h1
{
  color: #FF0000;
}

h1
{
  color: rgb(100%, 0%, 0%);
}
```

还可以使用 `rgba` 表示法指定颜色的透明度，在 `rgba` 表示法中，第 4 个参数是一个 0(完全透明)~1(完全不透明)之间的小数，如下所示：

```
color: rgb(255, 0, 0, 0.50);
```

使用颜色名称能增强 CSS 代码的可读性，但是由于可以指定名称的颜色相当少，所以常常需要用十六进制表示法来得到想要的精确颜色。在本章后面，将看到如何使用内置的颜色选取器来选择需要的颜色。

还有很多其他值，包括大小单位(`px`、`em` 等)、字体、图像(采用 `url(SomeImage.jpg)` 的形式)，或者所谓的枚举，如边框样式，允许将边框样式设置为 `solid`、`dashed`、`double` 等。

### 4. 使用缩略版本

CSS 的很多属性允许写一个缩略版本以及一个较扩展的版本。例如，`border` 属性。在最短形式中，可以将 `border` 属性设置为这样：

```
border: 1px solid Black;
```

这个 `border` 属性应用到 HTML 元素的 4 条边上。边框大小将为 `1px`，样式将为 `solid`(一些其他选项包括 `dashed`、`dotted` 和 `double`)，边框颜色设置为 `Black`。

这是一种容易的方式，可以快速地将 HTML 的 4 条边框设置为相同的值。然而，如果希望对各条边框和它们的属性有更多的控制，可以使用扩展版本，如下所示：

```
border-top-width: 1px;
border-top-style: solid;
border-top-color: Black;
border-right-width: 1px;
border-right-style: solid;
border-right-color: Black;
border-bottom-width: 1px;
border-bottom-style: solid;
border-bottom-color: Black;
border-left-width: 1px;
border-left-style: solid;
border-left-color: Black;
```

这个长版本将使各边应用完全相同的样式：4 条边都是 1 像素宽的黑色实线边框。在大多数情况下，人们可能更喜欢缩略版本而不是扩展版本，因为它更容易阅读和维护。然而，如果需要对边框有绝对的控制，例如，如果希望 HTML 元素的左边和上边是 2 像素的虚线边框，右边和下边是绿色实线边框，那么最好单独设置所有 4 个方向的各个 border 属性。还可以混合这些选项。下面的示例把 4 条边都设置为 1 像素宽的黑色实线边框，再重写左边框的颜色：

```
border: 1px solid Black;
border-left-color: Blue;
```

支持缩略版本的其他 CSS 属性包括 font、background、list-style、margin 和 padding。如果不确定某个属性是否支持缩略版本，就使用 IntelliSense 弹出列表。当在 CSS 文件或<style>块中输入一个属性时，按下 Ctrl+空格键，就会弹出这个列表。

尽管有时候看起来需要通过测试和检错反复地编写 CSS，而这只是为了得到正确的结果，但事实上在 CSS 的后台有一个十分精确的模型，它决定了页面上元素的布局方式。这个模型就是 CSS Box Model(CSS 方框模型)。

## 5. CSS 方框模型

CSS 方框模型描述了将 3 个重要的 CSS 属性应用于 HTML 元素的方式，这 3 个属性是 padding、border 和 margin。图 3-4 是这个方框模型的图形表示：

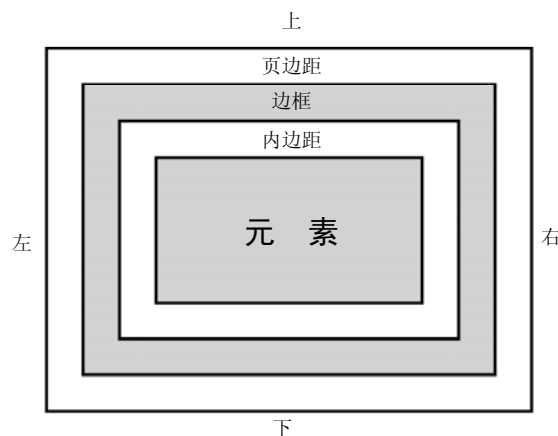


图 3-4

图 3-4 的中间部分是一个 HTML 元素，如一个具有特定高度和宽度的<p>或<div>元素。与它最近的一层是内边距(padding)，即边框内部包围 HTML 元素的白色部分。padding 的外层是边框(border)。最后，该图的最外层是页边距(margin)，它定义了某个元素(包括它的内边距和边框)与其周围元素之间的空间。元素外层的 3 个属性(padding、border 以及 margin)所占的空间总计等于元素在页面中所占的空间。要想知道这是如何工作的，请看下面的 CSS 和 HTML 代码：

```
.MyDiv
{
    width: 200px;
    padding: 10px;
    border: 2px solid Black;
}
...
<div class="MyDiv">Element</div>
```

运行上面的代码会在浏览器中显示出一个具有 2 像素黑色边框的矩形框，其中有一个<div>元素，如图 3-5 所示：

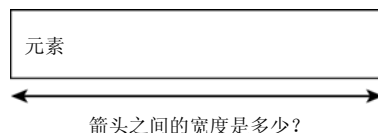


图 3-5

现在，先试着回答这个问题：<div>元素下面的箭头之间的宽度是多少？

如果你猜的是 224 像素，就猜对了！这个箭头的宽度是 3 个值的和，即实际元素的宽度(200 像素)、两侧包围元素的内边距的宽度(两个 10 像素)和两侧边框的宽度(两个 2 像素)；这 3 个值加起来就是 224 像素。因此，如果希望整个方框的宽度是 200 像素，只需要将 MyDiv 选择器的 width 属性设置为 176 像素即可。在元素边框的外部，margin 可以进一步影响这个元素及其周围元素之间的空间。

这个例子只演示了宽度属性的效果，但是这些规则同样适用于元素的高度属性。在布局页面时要记住这个方框模型。如果得到的元素宽于或高于期望的值，就要核对 CSS 样式表中 width、height、padding、border 以及 margin 属性的值。

在下面的“试一试”练习中，将修改在第 2 章中构建的站点的主页。在此练习中将为站点添加基本布局，然后用样式表格式化页面。在第 6 章把它升级为母版页时，将再次用到这个页面。

### 试一试

### 样式化 Planet Wrox 主页

在这个练习中将修改两个文件：首先，向 Default.aspx 页面中添加基本布局元素，来为标题、菜单、主内容区域、滚动条及页脚创建空间。然后编辑 Styles 文件夹中的 Styles.css 文件，修改这些元素的大小和位置。最后，把该样式表附加到页面上，这样，在设计器或浏览器中浏览页面时，就会应用这些样式信息。

- (1) 打开 Web 站点根文件夹中的文件 Default.aspx，如有必要，切换到 Markup 视图。
- (2) 修改<form>元素中的代码，结果如下所示：

```
<form id="form1" runat="server">
```

```

<div id="PageWrapper">
  <header>Header Goes Here</header>
  <nav>Menu Goes Here</nav>
  <section id="MainContent">
    <h1>Hi there visitor and welcome to Planet Wrox</h1>
    ...
  </section>
  <aside id="Sidebar">Sidebar Goes Here</aside>
  <footer>Footer Goes Here</footer>
</div>
</form>

```

一定要保证在第 2 章中添加的欢迎消息在 MainContent <section> 元素的起始和结束标记中出现。如果已经熟悉了 HTML5 之前的 HTML，就可能对新元素如 header、section 和 footer 感到惊讶。这些元素在 HTML5 中引入，给文档提供了更好的语义结构。HTML5 简介参见第 1 章。

(3) 从 Styles 文件夹中打开文件 Styles.css。如果之前在这个文件中添加了一些代码，则要先把这些代码删除掉。

(4) 在 CSS 文件的顶部，输入下面的代码，使用 ID 选择器选择 header 元素：

```

header
{
}

```

(5) 把鼠标放在花括号之间，并右击，从上下文菜单中选择 Build Style，这时将出现如图 3-6 所示的 Modify Style 对话框。

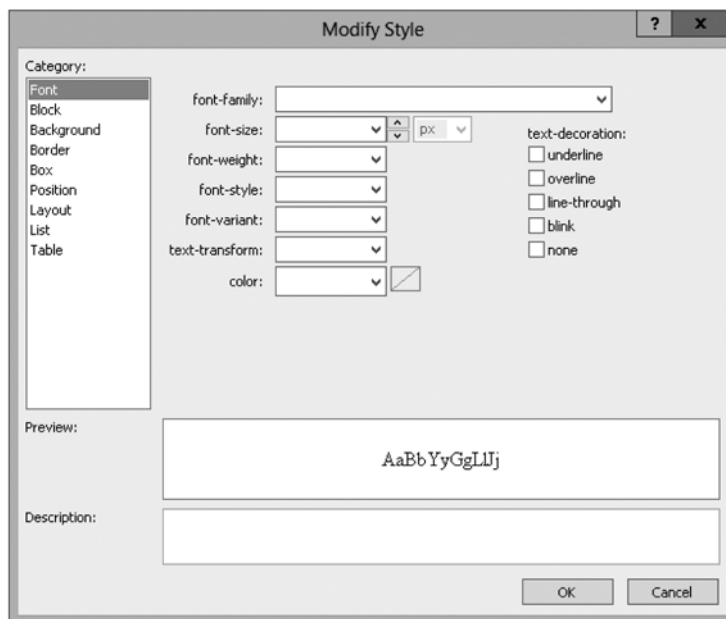


图 3-6

(6) 在左边的 Category 列表中，单击 Background 选项，然后打开背景色的下拉列表。从出现的颜色选取器中，单击 Silver 颜色，如图 3-7 所示。

此外，还可以直接在背景色文本框中输入颜色的十六进制代码，如 Silver(#C0C0C0)。

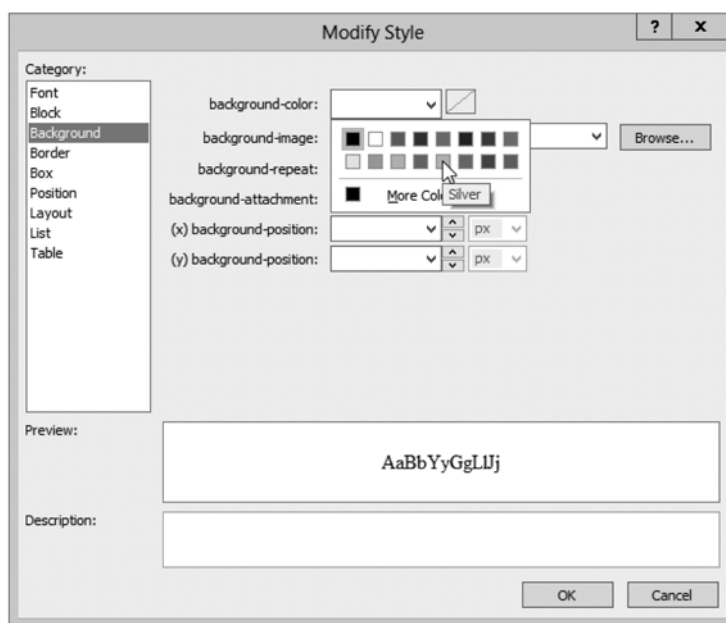


图 3-7

(7) 通过单击左边的列表切换到 **Position** 类别。出现的面板用来设置与位置相关的信息，包括高度与宽度。在宽度下面，输入 844，一定要选中右边下拉列表中的 px。高度输入 86。单击 OK 按钮关闭对话框，并向代码中插入声明，它现在应该如下所示：

```
header
{
    background-color: #C0C0C0;
    width: 844px;
    height: 86px;
}
```

(8) 重复第(4)~(7)步，创建下面的规则。如果愿意，也可以直接在代码编辑器中输入这些代码：

```
*
{
    font-family: Arial;
}

h1
{
    font-size: 20px;
}

#PageWrapper
{
    width: 844px;
    margin: auto;
}
```



```

nav
{
    width: 844px;
}

section#MainContent
{
    width: 664px;
    float: left;
}

aside
{
    background-color: Gray;
    width: 180px;
    float: left;
}

footer
{
    background-color: #C0C0C0;
    width: 844px;
    clear: both;
}

```

float 和 clear 属性在 Modify Style 对话框的 Layout 类别中,而 text-decoration 属性在 Font 类别中。因为 Universal 选择器(\*)应用到站点的所有元素上,所以通常将它移动到 CSS 的顶部,甚至在 header 选择器的前面,现在就可以这么做。

(9) 完成规则的创建后,保存并关闭文件 Styles.css,因为现在对它的操作已经完成。

(10) 再次打开文件 Default.aspx,并切换到 Design 视图。从 Solution Explorer 中,把 Styles 文件夹中的文件 Styles.css 拖到页面上,Design 视图就会发生改变,以反映在样式表中编写的代码。当把样式表放到页面上时,VS 会在 Markup 视图中页面的<head>部分插入向文档附加样式表的代码。

```

<head runat="server">
    <title></title>
    <style type="text/css">
        .auto-style1
        {
            color: #FF0000;
        }
    </style>
    <link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
</head>

```

也可以在 Markup 视图中页面的<head>部分直接从 Solution Explorer 中拖动一个现有样式表。这时,VS 会添加相同的<link>元素,但特性的顺序略有区别。

(11) 因为站点使用了 HTML5,所以需要有一个支持该最新 HTML 版本的浏览器,来正确显示页面。对于旧浏览器,例如 IE 8 及以前版本,可以使用一个漂亮的 JavaScript 库 Modernizr。这个库的一个功能是通过 JavaScript 动态添加对新 HTML 元素的支持,例如 nav、section 和 aside。

要把 Modernizr 添加到站点上,可以使用 VS 附带的 Package Manager Console。为了添加这个库,

可执行下面的步骤：

- (1) 选择 Tools | Library Package Manager | Package Manager Console。
- (2) 输入下面的命令，按下回车键：

```
Install-Package Modernizr
```

(3) 过一会儿，站点就添加了一个文件 `packages.config` 和一个文件夹 `Scripts`，文件 `packages.config` 用来跟踪已安装的包，文件夹 `Scripts` 现在包含文件 `modernizr-2.6.2.js`。注意：如果你阅读本书时，Modernizr 发布了新版本，版本号就会不同。为了把这个库添加到页面上，应在 Markup 视图中打开 `Default.aspx`，把该文件从 Solution Explorer 拖放到代码的 head 部分，位于样式表链接的后面。代码应如下所示：

```
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="Scripts/modernizr-2.6.2.js"></script>
</head>
```

即使有了现代的 Web 浏览器，也应把 Modernizr 库添加到站点上。因为我们不能控制访问站点的浏览器，但希望确保每个人都能看到期望的站点，添加 Modernizr 可以更正旧浏览器的 HTML5 兼容问题，且不带来什么系统开销。Modernizr 详见其网站 <http://modernizr.com/>。第 11 章会介绍 Package Manager Console。

(12) 最后，保存对所有打开的文档的修改(按下 Ctrl+Shift+S 组合键)，然后在浏览器中请求 `Default.aspx` 页面。这时屏幕应如图 3-8 所示，它显示了 Mozilla Firefox 中的页面。

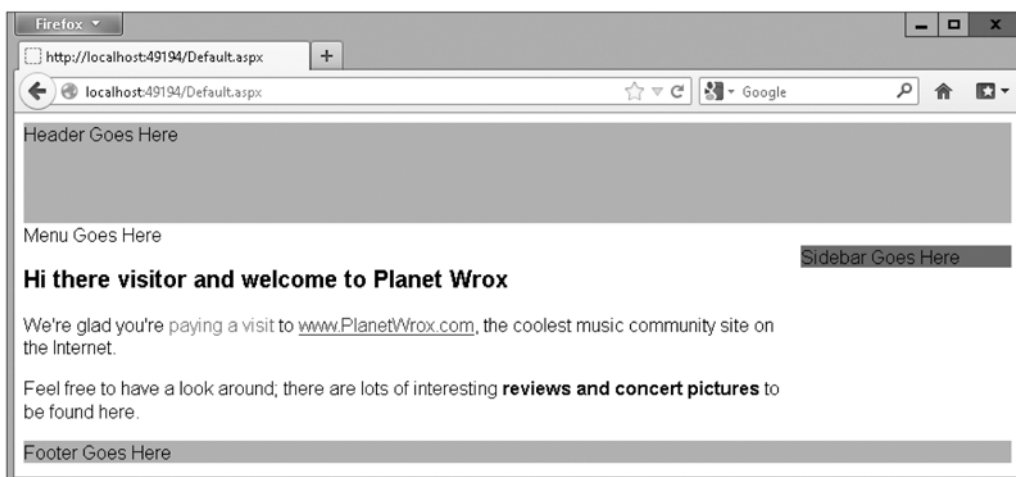


图 3-8

## 工作原理

Style Builder 使开发人员很容易选择 CSS 属性和修改它们的值。不需要记住关于 CSS 的所有细枝末节，而可以可视化地创建 CSS 代码。一旦很好地理解了 CSS 及其许多属性和值，则直接在 CSS 文件中输入代码常常会更快。

注意，header、PageWrapper、nav 和 footer 元素的精确宽度是 844 像素。这样，站点就可以很好地适合于 1024×768 像素的屏幕宽度(这种屏幕大小如今很普遍)，因而不用在窗口的边界中进行缩放。

屏幕较大的系统仅把站点居中放在可用的空间中。这个居中通过 `PageWrapper` 元素来实现，该元素把其 `margin` 设置为 `auto`。这表示左右边的可用空间(但不是顶部和底部的空间)会均分，有效地把 `PageWrapper` 元素居中在浏览器窗口的中心。

还要注意，`MainContent` section 和 `aside` 彼此相邻。这是用 CSS 的 `float` 属性实现的：

```
section#MainContent
{
    width: 664px;
    float: left;
}

aside
{
    background-color: Gray;
    width: 180px;
    float: left;
}
```

这段代码可以让 `MainContent` “浮动”在内容的左边，有效地将 `aside` 放在它的右边。还需要让 `aside` “浮动”。如果不进行这样的设置，它将放在页面的左边，而这里正好没有为它应用 CSS 样式。如果站点中有多个 `aside` 元素(这很常见)，就可以添加一个 `id` 特性(例如 `Sidebar`)，把这个用作边栏的 `aside` 元素作为目标，这与处理 `MainContent` section 元素相同，再更新 CSS，如下所示：

```
aside#Sidebar
{
    ...
}
```

`MainContent` 和 `aside` 元素的宽度合起来共 844 像素，恰好等于它们的父元素 `PageWrapper` 的宽度。

要结束浮动，并让 `footer` 元素放在 `MainContent` 和 `aside` 元素下面，可使用 `clear` 属性清除可能有效的任何浮动(左或右)：

```
footer
{
    background-color: #C0C0C0;
    width: 844px;
    clear: both;
}
```

灰色背景只是临时添加到代码中的，因此很容易看出 `<div>` 最终位于何处。在将来的练习中，会再次修改这个 CSS 文件，来适应 Planet Wrox Web 站点的模式。

要告诉浏览器应用什么样式，需在页面的标题中链接样式表：

```
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
```

这行代码会通知浏览器到文件夹 `Styles` 中查找一个名为 `Styles.css` 的文件，并将该文件中的所有规则应用到当前文档中。一旦浏览器下载了这个 CSS 文件，它就会将在那里找到的所有样式应用到 HTML 元素上，最终产生如图 3-8 所示的布局。

---

在这个练习中，可以看出如何用<link>标记向一个页面链接样式表。然而，在 Web 页面中包括样式表的方法有好多种。

### 3.2.3 向页面中添加 CSS

向 Web 页面中添加 CSS 样式表的首选方法是通过<link>元素，就像前面的练习中那样，它指向一个外部 CSS 文件。看看下面这个<link>，通过它可以了解向页面中嵌入样式表时有哪些选择：

```
<link href="StyleSheet.css" rel="Stylesheet" type="text/css" media="screen" />
```

href 属性指向站点内的一个文件，就像第 2 章在两个页面之间创建链接时那样。rel 和 type 特性告诉浏览器链接的文件实际上是一个层叠样式表。media 特性很有趣：它允许以不同的设备为目标，包括显示器、打印机、手持设备，甚至包括为视觉有障碍的访问者准备的盲文点字设备和助听器。media 的默认特性是 screen，因此如果目标是标准桌面浏览器，就完全可以省略这个特性。

在本章开头曾经简要介绍过包括样式表的另一种方法：使用嵌入的<style>元素。<style>元素应放在 ASPX 或 HTML 页面上方的<head>标记之间。在<style>标记内，可以编写与以前完全相同的 CSS 代码。例如，要单独修改当前页面中<h1>元素的外观，可以向页面的<head>标记内添加下列代码：

```
<head runat="server">
  <title></title>
  <style type="text/css">
    h1
    {
      color: Blue;
    }
  </style>
</head>
```

向 HTML 元素应用 CSS 的第三种方法是使用带有第 2 章指出的 style 特性的内联样式。由于 style 特性已经应用到了特定的 HTML 元素，因此不需要选择器，而可以直接把声明写在这个特性中：

```
<span style="color: White; background-color: Black;">
  This is white text on a black background.
</span>
```

#### 在外部、内嵌和内联样式表之间进行选择

既然有这么多种选择可以向站点中添加样式表，那么使用哪种方法最好呢？一般而言，外部样式表优于内嵌样式表，内嵌样式表优于内联样式表。外部样式表允许通过单个文件改变整个站点的外观。只要对外部样式表文件做一次修改，使用这个样式表的所有页面就会自动进行这样的修改。

然而，在某些情况下使用内嵌样式表和内联样式表也是完全可以接受的。如果要修改单个页面的外观，而不想影响到站点中的其他页面，那么内嵌样式表就是最佳选择。内联样式表也是如此，如果只想修改单个页面中单个元素的行为，且相当确定其他 HTML 元素不需要同样的声明时，就可以使用内联样式。

需要考虑的一件重要事情是不同类型的样式表互相覆盖的方式。如果有多个带不同属性值的同等选择器，那么会优先采用最后定义的那个选择器。例如，在名为 Styles.css 的外部样式表中定义了一个规则，它将所有<h1>标记的颜色都设置为绿色：

```
h1
{
    color: Green;
}
```

现在假设要将这个样式表应用到一个页面中,该页面中对同样的 h1 还有一个嵌入规则,但设置了不同的颜色:

```
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
<style type="text/css">
    h1
    {
        color: Blue;
    }
</style>
```

运行这段代码,页面中实际的<h1>标记的颜色就会是蓝色。这是因为将颜色定义为蓝色的内嵌样式表在页面中定义得较晚,因此它覆盖掉了外部文件中定义的设置。如果采用这样的样式:

```
<style type="text/css">
    h1
    {
        color: Blue;
    }
</style>
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
```

则标题将会是绿色的,因为此时外部样式表中的设置否决了内嵌样式表中的值。

内联样式表也是采用这样的机制。由于它们是直接在 HTML 元素上定义的,因此与内嵌和外部样式表相比,会优先采用它们的设置。

也要注意 CSS 通常否决 HTML 元素上的特性。例如,如果有一个 CSS 规则设定了某个图像的宽度和高度,站点就会忽略 img 元素上的 height 和 width 特性,在这个例子中最终图像会显示为 100 像素的方形:

```
img
{
    height: 100px;
    width: 100px;
}
...

```



**注意:** 关于 CSS 的知识远不止这里提到的这些。要了解关于 CSS 的更多信息,请参见清华大学出版社引进并出版的《CSS 入门经典(第 3 版)》一书(ISBN: 978-7-302-27624-1)。

一般而言,建议在<head>部分的上方应用外部文件,后跟内嵌样式表。这样,外部文件可以定义元素的全局外观,然后可以用内嵌样式来否决外部设置。

用 VS 可以轻松地将内嵌样式表移到外部 CSS 文件中。下面一节将讨论 VS 中其余的 CSS 工具,

在其中会介绍具体操作。

### 3.3 在 Visual Studio 中使用 CSS

VS 的以前版本中有几个使用 CSS 的便利工具，如下所示：

- **CSS Properties 面板** 用来修改属性值。
- **Manage Styles 窗口** 用来组织站点的样式，将它们从内嵌样式表改为外部样式表，反之亦然；对它们重新排序；将现有样式表链接到一个文档；创建新的内联、内嵌或外部样式表。
- **Apply Styles 窗口** 用来从站点中选择所有可用样式，并将它们快速地应用到页面中的不同元素上。
- **Style Builder** 用来可视化地创建声明，如前所述。

在 VS 2012 中，CSS 工具有了很大的改进，现在也支持如下选项：

- 层次缩进，使 CSS 代码更容易理解。
- 更智能的 IntelliSense 功能，更便于手工输入 CSS 代码。
- 许多有益的编辑器功能，例如更容易注释和取消注释代码，在可折叠的区域中放置代码，颜色选择器和代码段。

下面几节会详细解释这些工具。

#### 3.3.1 使用 CSS 编辑器

VS 中的 CSS 文本编辑器包含许多强大的功能，如下面的“试一试”练习所述。

##### 试一试

##### 试用 CSS 编辑器

在下面的“试一试”练习中，将修改前面创建的 Styles.css 文件，添加几个新样式。在这个过程中将介绍前面列出的一些 CSS 编辑器功能。

(1) 从 Styles 文件夹中打开文件 Styles.css，定位到 section#MainContent 选择器。在其结束花括号的下方，添加如下 CSS 选择器，它指向 MainContent 区域中的链接(a 元素)：

```
section#MainContent a
{
}
```

(2) 在类型 color 的起始和结束花括号之间，输入冒号(:)后跟一个井号(#)。一输入“#”，VS 就会显示一个颜色选择器。如果单击颜色选择器右边的向下箭头，它就会展开，显示更多的选项，如图 3-9 所示。

这个颜色选择器有几个有趣的选项。首先，顶行彩色方块包含样式表中定义的一组颜色，这是快速选择已经用于另一个选择器的颜色的

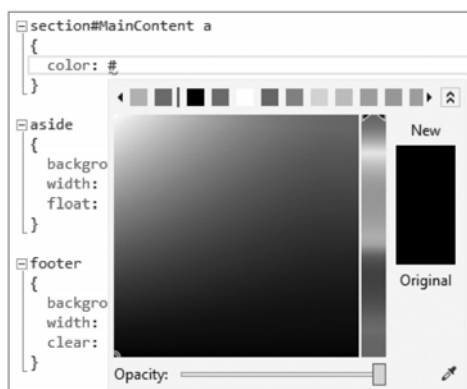


图 3-9

极佳方式。识别出的颜色列表后跟一个竖杠，再后面是 VS 自动添加的一些默认颜色。在图 3-9 中有两种识别出的颜色：淡灰色和深灰色。其他颜色都是默认的。其次，把鼠标拖过中间的大色块和右边的彩色竖条，就可以混合出自己的颜色。最后，使用屏幕右下角的小颜色选择器图标，可以从其他 Windows 应用程序或桌面中快速选择出某种颜色。这是在某个图形程序中打开某个图像后，从该图像中检索颜色的好方法。

如果希望输入自己的颜色信息，只需要忽略颜色选择器(或者按下 Esc 键来关闭它)。

对于这个练习，单击绿色块，这会自动插入该颜色的 16 进制值(例如#4cff00)。输入一个分号，再按下回车键，完成这行代码。

(3) 在这个新行上输入字母 t，打开 IntelliSense 列表，它会显示所有以 t 开头的属性，如图 3-10 所示。

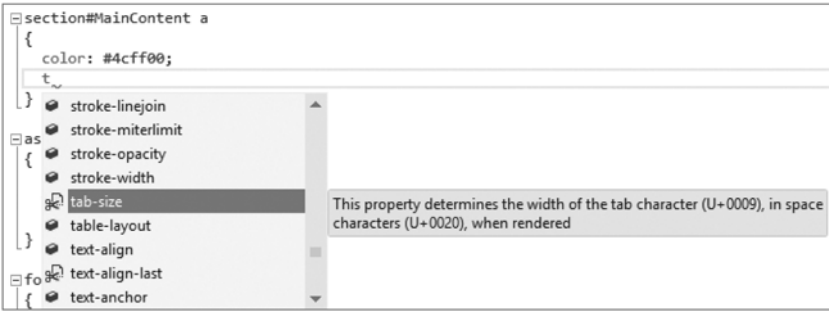


图 3-10

接着，输入字母 d，列表就会进一步过滤，如图 3-11 所示。



图 3-11

注意这个过滤操作是如何以智能方式完成的。过滤器不仅将列表限制为以 td 开头或包含字母 td 的属性(本例不存在包含 td 的属性)，而且支持所谓的标题搜索(它会列出以指定字母开头的属性中的每个单词)。对于本例，td 匹配的项有 text-decoration 和 transition-delay。如果输入 tdc，该列表就只显示 text-decoration-color，这也适用于所有其他 CSS 属性，例如输入 bc，该列表就会列出 background-color；输入 bbw，该列表就会列出 border-bottom-width，等等。这将非常便于通过几次按键来编写 CSS 代码。

注意，列表中的一些项带有剪刀图标，而不是标准的属性图标。剪刀图标表示，按下 Tab 键两次，就可以展开代码段。这么做可以把浏览器特定的 CSS 插入文件，用于还没有由所有浏览器都实现的 CSS 属性。

继续这个练习。从列表中选择 text-decoration。可以双击该值，按下回车键或者 Tab 键，VS 就会完成这个属性的编写。接着，输入一个冒号和 underline 作为 text-decoration 属性的值。用一个分号结束该行。最终的代码如下所示：

```

section#MainContent a
{
    color: #4cff00;
    text-decoration: underline;
}

```

(4) 在刚才创建的样式规则下面，再添加两个样式规则，如下面的代码所示：

```

section#MainContent a:visited
{
    color: #FF0000;
    text-decoration: underline;
}

section#MainContent a:hover
{
    color: #FFA500;
    text-decoration: underline;
}

```

(5) 接着从主菜单中选择 Edit | Format Document。或者按下 Ctrl+K，接着按下 Ctrl+D，这个快捷键和菜单命令也可以用于其他类型的文件，例如 HTML、ASPX、VB 和 C# 文件。执行时，它们会根据特定的文件类型设置格式化文档。对于 CSS，它格式化代码的方式是：把起始花括号放在一致的位置上，缩进并格式化各个规则。它还会执行所谓的层次缩进。为了更好地理解其工作原理，可以看一下图 3-12，它列出了格式化文档后，用于前面添加的 a 元素的 section#MainContent 规则集和新选择器。

注意在 section#MainContent 规则集的下面，section#MainContent a 规则集已经进行了缩进。这有助于理解规则集使用的两个选择器之间的关系(section#MainContent a 只针对 id 为 MainContent 的父 section 元素中的 a 元素)。同样，section#MainContent a:visited 和 section#MainContent a:hover 规则集在 a 元素的下面也进行了缩进，因为它们可以看作 a 元素的子元素。如果不喜欢这个行为，可以关闭它，方法是：使用 Tools | Options | Text Editor | CSS | Formatting；或者按下 Ctrl+Q，进入 Quick Launch 框，输入 CSS Formatting，再按下回车键。这会打开相同的 Options 对话框，在这个对话框中，可以花点时间浏览 Text Editor 节点下的项，了解如何改变 VS 支持的许多语言的格式。

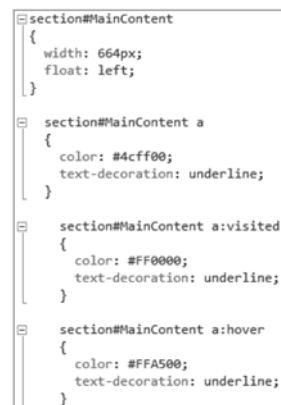
(6) 这里要讨论的最后两个浏览器特性是区域和注释。如果 CSS 代码比较长，把其中一些彼此相关的代码放在一个区域中可能会有帮助。VS 中的区域可以折叠起来，隐藏代码，需要修改代码时再展开它。要创建区域，可以使用 CSS 注释语法把代码放在 region / endregion 对中。例如：

```

/*#region Name Of Region */
... Your CSS code here
/*#endregion*/

```

对于这个练习，在 section#MainContent 规则集的上面添加起始区域语句(/\*#region Main Content \*/)，其名称是 Main Content，在 section#MainContent a:hover 规则集的下面添加结束语句(/\*#endregion



```

section#MainContent
{
    width: 664px;
    float: left;
}

    section#MainContent a
    {
        color: #4cff00;
        text-decoration: underline;
    }

    section#MainContent a:visited
    {
        color: #FF0000;
        text-decoration: underline;
    }

    section#MainContent a:hover
    {
        color: #FFA500;
        text-decoration: underline;
    }

```

图 3-12



\*/)。创建了区域后,就可以在文本编辑器旁边看到一个减号图标(-),如图 3-13 所示。

现在单击减号图标,就可以折叠该区域。VS 会隐藏代码块,而显示区域的名称,如图 3-14 所示。

在处理大代码文件时,这就很容易显示某些代码。注意:其他文件类型也支持区域,例如 C# 和 VB 文件,但每种语言使用的语法略有区别。

VS 中的区域功能使用了标准的 CSS 注释语法。这意味着,浏览器会忽略区域代码及其名称。

要注释掉其他代码(这样浏览器就不会解释它们),可以选择一些文本,按下 Ctrl+K,再按下 Ctrl+C(表示 Comment)。代码就会用/\*和\*/注释掉,如图 3-15 所示。

要取消对代码的注释,可以按下 Ctrl+K,再按下 Ctrl+U(表示 Uncomment)。这两个命令也可以在 Edit | Advanced 菜单和 Styles 工具栏上找到。还可以输入/\*,手工开始一个注释。接着 VS 会自动插入结尾的\*/。

(7) 保存对 CSS 文件的修改。可以留下区域代码,但应确保取消注释上一步注释掉的任何代码。

(8) 切换到页面 Default.aspx,如果需要,切换到 Design 视图,在段落中选择文本“look around”,如果在上一个“试一试”练习中输入的是别的内容,就选择那个文本。在这一阶段,最重要的是把一些文本转化为链接。

(9) 在 Formatting 工具栏上,单击 Convert to Hyperlink 按钮(带有链接标志和箭头),在出现的对话框中单击 Browse 按钮,并在站点的根位置中选择文件 Default.aspx。这样,链接就会指向在其中定义它的那个页面,对于本练习来说这样很好。单击 OK 按钮两次,关闭对话框。

(10) 保存对所有打开的文档所做的修改(从主菜单中选择 File | Save All 命令或按下 Ctrl+Shift+S 组合键),然后按下 Ctrl+F5 组合键,在浏览器中请求 Default.aspx 页面。这时应看到出现一个页面,其上有一个带有下划线的“look around”链接,如图 3-16 所示,它在 Google Chrome 中显示页面。

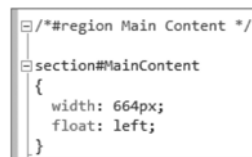


图 3-13

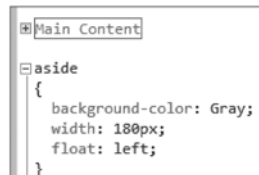


图 3-14

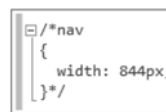


图 3-15

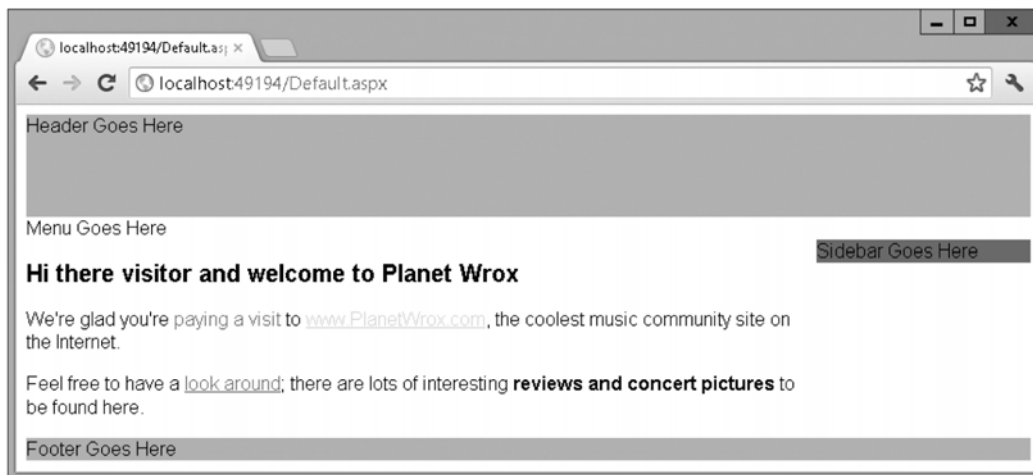


图 3-16

(11) 把鼠标悬停在“look around”链接上,注意它变成了橙色。

(12) 注意到 [www.PlanetWrox.com](http://www.PlanetWrox.com) 的链接变成绿色(假定以前还没有访问过这个站点),而到 `Default.aspx` 的链接变成红色(因为当前在访问 `Default.aspx`,所以浏览器将这个页面标记为“已访问”)。如果访问 Planet Wrox 网站,再返回 `Default.aspx` 页面,到 [www.PlanetWrox.com](http://www.PlanetWrox.com) 的链接也就变成红色。把鼠标悬停在它上面,仍会把它变成橙色。如果要在另一个浏览器中测试页面,就在 Solution Explorer 中右击 `Default.aspx`,并从上下文菜单中选择 **Browse With** 命令。如果另一个浏览器已经列在那里,则从列表中选择它,然后单击 **Browse** 按钮。也可以通过单击 **Set as Default** 按钮将这个浏览器设置为默认浏览器。

如果没有列出想要的浏览器,则单击 **Add** 按钮,然后单击 **Program Name** 框旁边的省略号来查找喜欢的浏览器。当浏览器显示在列表中时,单击它以选择它,然后单击 **Browse** 按钮在那个浏览器中打开页面。这个页面现在应出现在另一个浏览器中。

### 工作原理

这个练习首先展示了 VS 中 CSS 编辑器的一些功能。**IntelliSense** 列表过滤和标题搜索功能在编写 CSS 代码时可以节省大量时间。读者需要花费一些时间来熟悉它们,但一旦习惯了使用它们,就会觉得没有它们就无法工作。

`a` 选择器上的 `:hover` 和 `:visited` 部分你可能不太熟悉。这些选择器称为伪类选择器。`a:visited` 选择器仅用于已经在浏览器中访问过的链接,`a:hover` 选择器仅用于当用户将鼠标悬停在链接上时的 `<a>` 标记。

在浏览器中打开页面时,就已经下载了更新后的样式表,然后浏览器会将 `a:visited` 规则集应用到之前访问过的 `MainContent` section 的所有链接上。将鼠标悬停在一个链接上时,就应用了规则集 `section#MainContent a:hover`,使链接变成橙色。

在另一个浏览器中查看页面是个不错的方法。尽管现代浏览器对页面的显示越来越相似,但是仍有一些细微的差别需要注意,并在 HTML 和 CSS 代码中针对这种差别进行处理。在系统中安装几个不同的浏览器(例如 IE、Firefox、Safari、Opera 和 Chrome 等),按照这个“试一试”练习中的说明把它们分配到 **Browse With** 对话框中,并在这些浏览器中尽可能多地测试页面,这将有助于确保页面的显示在主流浏览器中完全一致。

尽管外部样式表可能很有用,但是仍然会有一些时候确实想使用内嵌或内联样式。创建并管理这些样式相当容易,下一节将会介绍。

### 3.3.2 创建内嵌和内联样式表

在操作 **Design** 视图中的页面时,常常需要对页面的部分内容作些细微调整,如样式化一块文本,对齐一个图像,或者向一个元素应用边框。在本阶段,需要决定是创建一个内联、内嵌还是外部样式表。如前所述,如果预测到以后会重用一個样式,则应选择外部或内嵌样式表。但在 VS 中采用何种样式表并不太紧要,它允许创建所有 3 个级别的样式表,而且它还允许轻松地将内嵌样式升级为外部样式,或者将内联样式信息复制到不同位置,因此提供了很大的灵活性,以后也可以随时改变主意。

在下一个“试一试”练习中,将介绍如何创建内联和内嵌样式表。以后将介绍如何将这

移动到外部样式表中，使其他页面可以重用相同的样式。

试一试

在一个页面中创建内嵌和内联样式

在本“试一试”练习中，将向页面的<h1>元素添加一个样式规则，以删除浏览器在这个标题周围保留的默认页边距。此外，还将用一个 class 特性来样式化第一段，给它设计一个与众不同的外观，使之突出于页面上的其他段落。

(1) 回到 VS，并确保 Design 视图中的页面 Default.aspx 是打开的。

(2) 在文档窗口中的<h1>元素上单击一次以选中它，然后选择 Format | New Style 命令。这时会出现 New Style 对话框(如图 3-17 所示)，它与前面的 Modify Style 对话框十分相似。

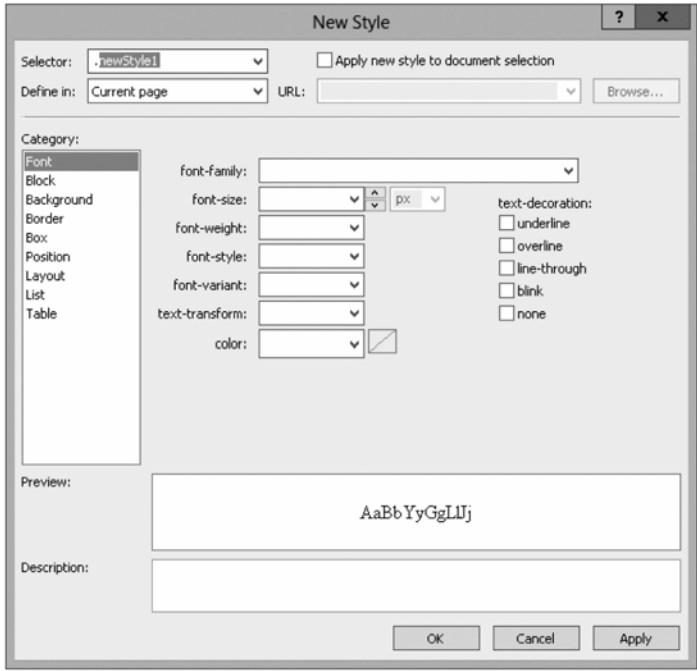


图 3-17

(3) 在屏幕的上方，打开 Selector 下拉列表并选择内联样式。它是列表中的第一项，这样就可以确保将新样式作为内联样式应用到<h1>元素上。

(4) 切换到 Box 类别，如图 3-18 所示。

这个对话框中有一个方便的关系图，可作为 CSS 方框模型的补充，它显示了 CSS 属性最终出现的位置，如 Padding、Border 和 Margin。

默认情况下，浏览器会在<h1>元素上方或下方绘制一些白色空白，但空白的实际大小因不同的浏览器而异。为了给各个浏览器以相同的设置，可以把内边距重置为 0，然后在标题下方应用一点页边距，这样就可以在它和紧邻它的元素之间生成一些距离。为此，在 top 文本框中把内边距设置为 0，从文本框旁边的下拉列表中清除值。保持选中 Same for all 选项，VS 会创建一个缩略声明。然后取消选中 margin 部分的 Same for All 选项，在 top、right 和 left 文本框中输入 0，从每个值旁边的下拉列表中清除值。在 bottom 文本框中输入 10，确保在下拉列表中选择 px，此时屏幕应如图 3-18 所示。

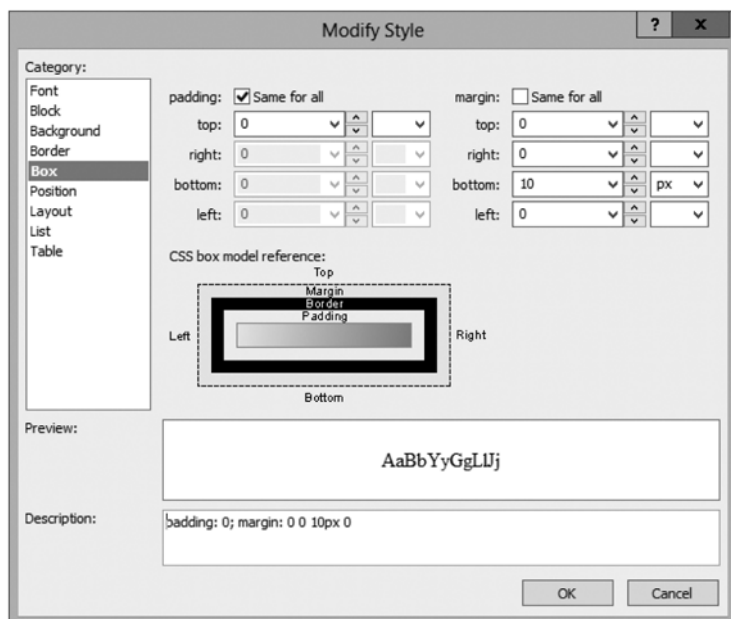


图 3-18

单击 OK 按钮关闭对话框，将修改应用到标题上。最后会在 Markup 视图中得到下面带内联样式的<h1>元素。

```
<h1 style="padding: 0; margin: 0 0 10px 0">
  Hi there visitor and welcome to Planet Wrox
</h1>
```

(5) 接下来，在 Design 视图中单击选中第一段。这时会出现一个小浮起，表示选中了一个<p>元素，如图 3-19 所示。文档窗口底部的标记选择器应突出显示<p>元素。

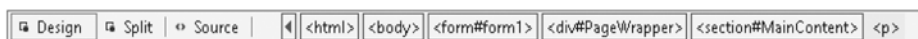


图 3-19

(6) 在这一段仍然被选中的状态下，从主菜单中选择 Format | New Style 命令。这时，不是创建内联样式，而是在 Selector 框中输入文本.Introduction，如图 3-20 所示。不要忘记输入选择器名称前面的句点(.)。

(7) 在屏幕上方，选中 Apply new style to document selection 前面的复选框。在这个设定打开的情况下，要创建的新类就会应用到已选中的<p>元素上。

(8) 从 font-style 下拉列表中选择 italic。New Style 对话框现在看起来应当如图 3-20 所示。

(9) 最后，单击 OK 按钮。注意整个段落现在已经显示为斜体。

(10) 在<p>元素仍然被选中的情况下，通过选择 View | CSS Properties 命令打开 CSS Properties 面板(如图 3-21 所示)。从这个面板中可以纵览所有的 CSS 属性，并显示页面中哪些属性当前是活动的。

该 CSS Properties 面板的上半部分显示了应用的规则列表，下半部分用来显示这些规则的 CSS 属性。在图 3-21 中可以看到应用到.Introduction 选择器的规则。

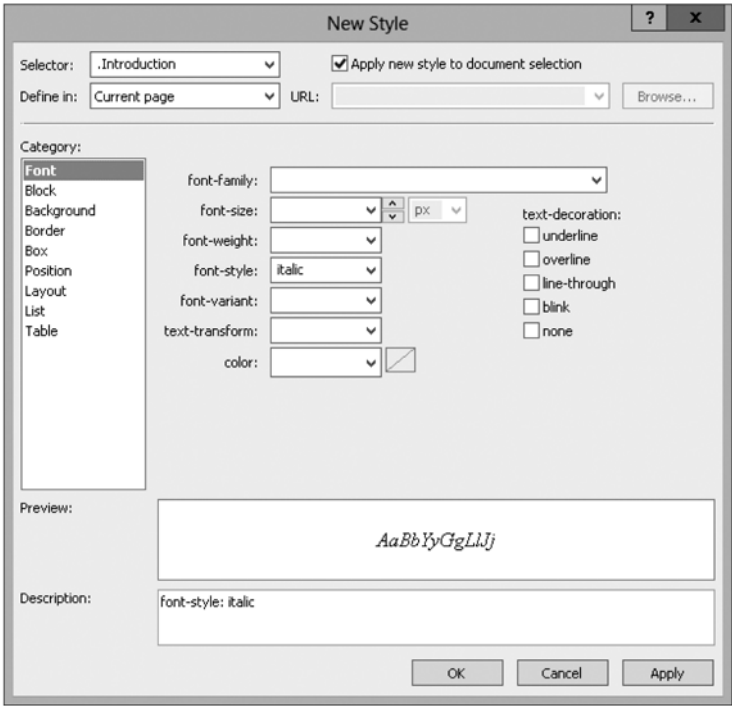


图 3-20

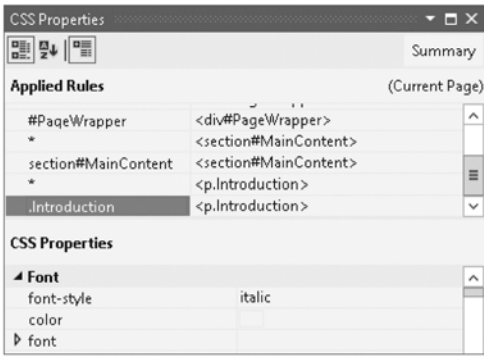


图 3-21

(11) 在 CSS Properties 列表中，定位到 color 属性并把它设置为深蓝色，如#003399。为此，需要打开属性值的下拉列表，并从颜色选取器中选择一种颜色。如果所寻找的颜色不可用，那么请单击 More Colors 按钮以扩展颜色选取器，如图 3-22 所示。

也可以不使用颜色选取器，直接在 Properties 面板中输入一个值。下面是 CSS Properties 面板中许多属性的工作方式：它们允许直接输入值，或者使用下拉列表或属性的值框后面的省略号按钮来可视化地修改属性值。图 3-23 显示了一个方便的下拉列表中用于 font-style 属性的不同选项。

要特别注意窗口上方的 3 个按钮，因为它们容纳了一些有用的功能。前两个按钮允许在按类别的模式和按字母顺序的模式之间切换，更容易找到正确的属性。第三个按钮用来在列表上方或者在列表中的默认位置显示应用的属性。

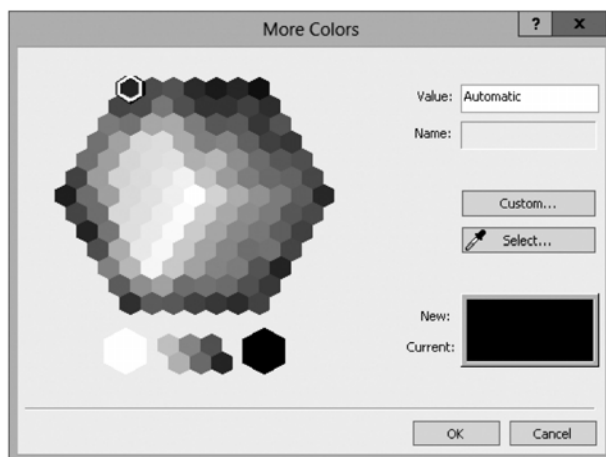


图 3-22

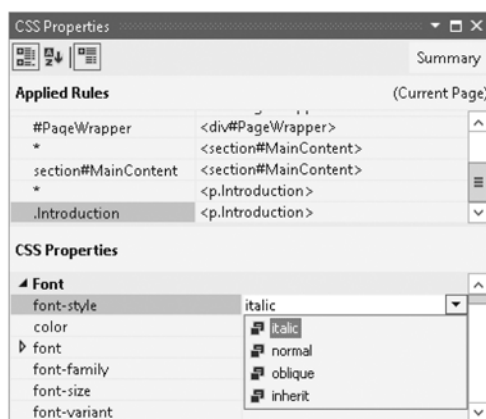


图 3-23

(12) 最后，保存所有修改，并在浏览器中打开 Default.aspx 页面(如图 3-24 所示)。可以发现第一段现在除了文本中的链接根据以前是否访问过该站点而显示为绿色或红色外，其余内容都显示为蓝色斜体字。此外，如果跟着第 2 章的所有指令做了，那么文本“paying a visit”就是红色的，它是由嵌入的 CSS 类设置的。

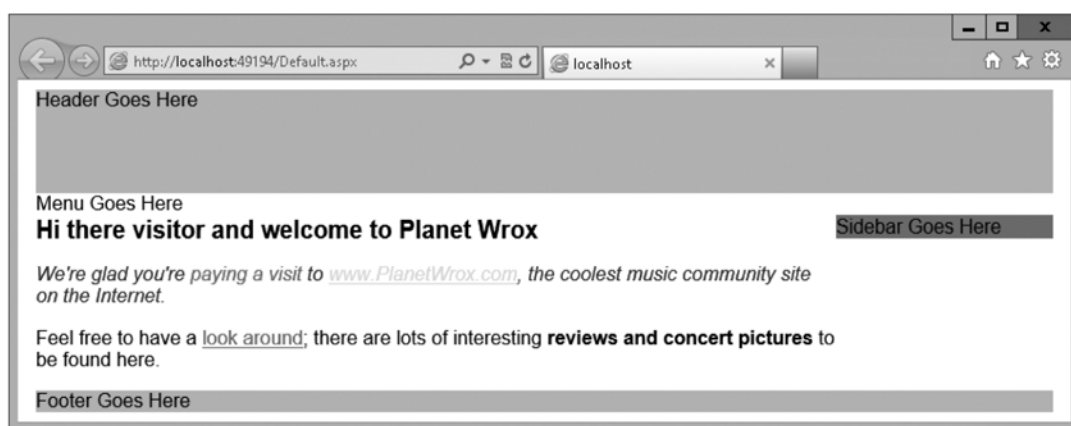


图 3-24

(13) 切换回 VS 并在 Markup 视图中查看页面。在页面的<head>部分，应当看到下面的内嵌样式表：

```
.Introduction
{
    font-style: italic;
    color: #003399;
}
</style>
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
```

### 工作原理

VS 提供的众多工具使得为 Web 站点编写 CSS 很简单。不需要手写任何代码，也不需要记得 CSS 标准所支持的各种属性。相反，只需要从 CSS Properties 面板上的不同列表中选择它们。这个面板允许手动输入值，也提供了方便的工具可以从下拉列表中选择颜色、文件和项。

在 Properties 面板中进行的所有修改都将应用到相关的样式表中，不管使用的是内联、内嵌还是外部样式表。与此同时，还会更新 Design 视图以反映设置的新 CSS 选项。

当查看<h1>元素时，可以看到 VS 将 padding 设置为 0，创建了一个内联样式，以反映 4 边的情况，并将 margin 设置为 0 0 10px 0 来逐个控制 4 条边。

在创建了一组有用且可以重用的样式后，需要有一种方法将现有样式应用到其他页面或 HTML 元素中。下面将介绍其工作原理。

### 3.3.3 应用样式

如果使用过 Microsoft Word，则很可能习惯 Styles 对话框，它列出了所有可用的样式，并允许把它们应用到文本的选中部分。用这种方式可以快速地文本块应用同样的格式。在 VS 中的工作方式与此类似。从主菜单中选择 View | Apply Styles 命令打开 Apply Styles 窗口，就可以轻松地向页面中的元素应用样式规则。

#### 试一试

#### 使用 Apply Styles 窗口

在这个练习中，将重用.Introduction 类，并将它也应用到页面的第二段。这样，这两个段落最后的外观就是相同的。

(1) 仍然使 Default.aspx 处于打开状态，并确保处在 Design 视图中，然后通过单击来选中页面中的第二段。确保文档窗口下方的标记选择器显示<p>元素被选中，而不是<p>元素中的其他标记，如<strong>。如果文本中只有一段，那么要先创建一个新段落(在 Design 视图中第一段后面按下 Enter 键)，输入一些文本，然后选中那个段落。

(2) 选择 View | Apply Styles 命令，打开 Apply Styles 窗口。确保该查看没有停靠在主文档窗口中，而是浮动的，或者放在文档窗口的边上。这个窗口显示了它在当前页面中找到的所有选择器和附加的任何样式表。如果没有看到如图 3-25 所示的所有样

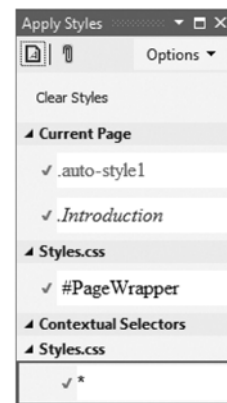


图 3-25

式，则单击 Options 按钮，并选择 Show All Styles 选项。

(3) 单击 CSS Styles 列表中的 Introduction 类，VS 会向<p>标记添加一个 class 特性：

```
<p class="Introduction">
  Feel free to have a <a href="Default.aspx">look
  around</a>; there are lots of
  interesting <strong>reviews and concert
  pictures</strong> to be found here.
</p>
```

如果要应用多个类，则在单击列表中的其他类之一时，按住 Ctrl 键，这样就会给元素的 class 特性应用一个类列表，类之间由一个空格隔开。也可以对 Markup 视图中选中的样式应用同样的步骤。

(4) 单击 Clear Styles 按钮，就可以快速地从标记中删除现有类和内联样式。考虑第 2 章中的一个 HTML 片段，那时用了 Formatting 工具栏格式化页面中的文本。如果使用的是 Foreground Color 按钮，那么最后代码会如下所示：

```
We&#39;re glad you&#39;re <span class="auto-style1">paying a visit</span>
```

要删除 class 特性，可选择标记选择器中的<span>标记，或者简单地在 Markup 视图中单击<span>标记，然后在 Apply Styles 窗口中单击 Clear Styles，这个项可以从图 3-25 上方看到。最后的 HTML 会是这样：

```
We&#39;re glad you&#39;re paying a visit
```

因为文本两端的空<span>没有用，所以 VS 也删除了它。另外，VS 还删除了 auto-style1 规则，因为页面上的代码不再使用它。

从 HTML 元素中删除样式特性的工作方式也是这样。

### 工作原理

再次说明，VS 能保持所有的相关窗口同步：Design 视图、Markup 视图和各种 CSS 设计工具。当从 Apply Styles 窗口中应用一个类时，VS 会将请求的类添加到 Markup 视图中选中的 HTML 元素中，然后它也会更新 Design 视图窗口。类似地，当从 Design 视图的内嵌样式中删除一个选择器或声明时，Design 视图和 CSS Tools 窗口都会更新。

本章要介绍的最后一个 CSS 功能位于 Manage Styles 和 Apply Styles 窗口中。除了可以向文档中附加 CSS 文件外，这些窗口还有助于轻松地管理样式。

### 3.3.4 管理样式

由于添加新的内联和内嵌样式是如此容易，因此页面可能很快就变得混乱不堪。要实现可重用性，应将尽可能多的内联和内嵌样式移到一个外部样式表中。这就是 Apply Styles 和 Manage Styles 窗口的作用所在。

试一试

使用 Manage Styles 和 Apply Styles 窗口管理样式

在本章前面，修改了<h1>元素，并向标题应用了内边距和页边距。然而，Default.aspx 不是在标



题中从该样式获益的唯一页面，因此有必要把它移到 `Styles.css` 文件中。类似地，`Introduction` 类的可重用性似乎够好，因此可以把它包括到 `Styles.css` 文件中，以便其他页面能访问它。本“试一试”练习显示了如何在站点中移动样式。

- (1) 确保 `Default.aspx` 仍然是打开的，如有必要，切换到 Markup 视图。
- (2) 定位到 `<h1>` 元素并再单击一次。VS 突出显示了位于文档窗口底部的标记选择器中的标记，以表明它是活动标记。

(3) 从主菜单中选择 **View | Apply Styles** 命令，打开 **Apply Styles** 窗口。如果窗口与其他窗口停靠在一起，则只需要简单地单击它的选项卡来让它成为活动状态。请确保窗口不是偶然地与主文档窗口停靠在一起，而是浮动的或是放置在文档窗口的一侧。**Apply Styles** 窗口下方显示了内联样式(如图 3-26 所示)。



图 3-26

(4) 右击 **Inline Style**，并选择 **New Style Copy** 命令。这时会出现 **New Style** 对话框，允许创建一个基于当前选项的新样式。在这个窗口的上方，从 **Selector** 下拉列表中选择 **h1**，并从 **Define in** 下拉列表中选择 **Existing style sheet** 选项。从 **URL** 下拉列表中，选择 **Styles/Styles.css** 选项。如果这个项不可用，则单击 **Browse** 按钮来定位并选择它。对话框最后应如图 3-27 所示。

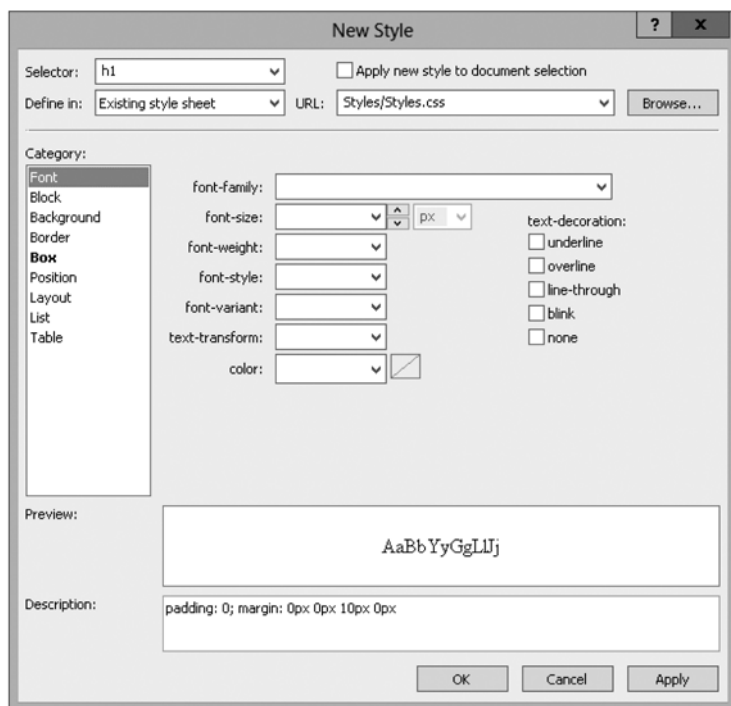


图 3-27

- (5) 单击 OK 按钮关闭对话框。VS 会创建 h1 样式的一个副本，并把它放在文件 Styles.css 中。注意 VS 在 Styles.css 文件中为 h1 创建了一个新的选择器，而不是向现有规则集中添加内边距和页

边距信息。如果愿意，可以将这两个选择器手动合并为一个。

(6) 在 Apply Styles 窗口中，再次右击 Inline Style，这次从上下文菜单中选择 Remove Inline Style 命令。这样会将样式特性从 h1 元素中删除。

(7) 从主菜单中选择 View | Manage Styles。请再次确认窗口是放在文档窗口的一侧，而不是停靠在文档窗口内部。在 Current Page 项下，定位 Introduction 选择器。

(8) 单击 Introduction 选择器，然后把它拖到 Styles.css 的区域中，例如把它放在 h1 选择器后面。注意，当鼠标悬停在选择器上时，VS 会在选择器之间画线，表示选择器最后将出现的位置。图 3-28 显示了如何将 Introduction 选择器从当前页面拖到 h1 和 #PageWrapper 选择器之间的 Styles.css 区域中。

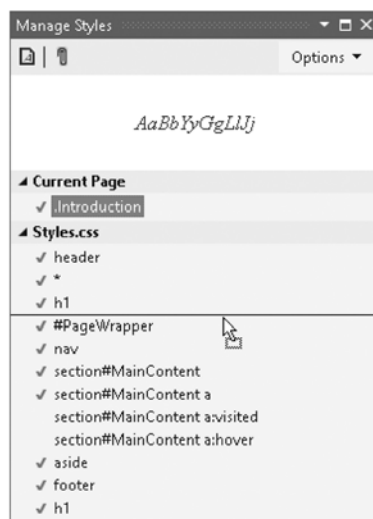


图 3-28

(9) 一旦把选择器放到 Manage Styles 窗口的 Styles.css 部分中，则与该选择器关联的样式就会从当前页面中去除，然后插入到 Styles.css 中。由于那个 CSS 文件包括在使用 <link/> 元素的当前页面中，因此在 Design 视图中看不到区别。现在，可以把空 <style> 元素从 Default.aspx 中删除，因为不再需要它了。

(10) 如果还没有合并两个 h1 选择器，则打开 Styles.css，向下滚动到文件的末尾。把 padding 和 margin 属性所在的两行复制到剪贴板上，再删除整个选择器。在文件中向上滚动，再把两个 CSS 规则粘贴到其他 h1 选择器中，代码最后如下所示：

```
h1
{
    font-size: 20px;
    padding: 0;
    margin: 0 0 10px 0;
}
```

(11) 保存所做的任何修改，然后通过按下 Ctrl+F5 组合键，在浏览器中打开 Default.aspx 页面。注意，这些段落没有改变，使用的仍然是蓝色斜体字。

## 工作原理

遗憾的是，VS 不允许把内联样式移到外部样式表文件中。然而，可以通过创建现有样式的副本，然后删除原始内联样式来获得同样的效果。在文件之间移动内嵌或外部样式表就容易得多。可以简单地把样式从一个文件拖到另一个文件中，并且 VS 会自动移动代码。这使得组织 CSS 极其容易。不要让所有嵌入的 CSS 都留在页面中，因为那样就可能会碰到它，现在可以简单地把它拖放到一个外部文件中。这使得在其他页面中重用那些样式容易得多，从而降低了页面大小和页面膨胀，并使站点更容易管理。显然，重要的是 CSS 所移到的文件被附加到了正在使用的页面中。

## 3.4 关于使用 CSS 的实用提示

请根据这些提示来制作大部分 CSS：

- 花一些时间熟悉 CSS 支持的众多属性。最佳方法是在 Demos 文件夹中创建一个崭新的页面，再创建几个像<div>和<p>标记这样的 HTML 元素，然后简单地配备几个不同的属性。通过在 CSS Properties 面板上尝试其中的很多属性，就能对可用的选项心中有数。这样，以后如果要向一些内容应用某种效果，就会容易得多。
- 当创建自定义 CSS 类时，试着命名一些能描述规则的行为而不是描述外观的名称。例如，使用一个名为.Introduction 的类样式化页面的第一段，它就是一个不错的描述。通过它可以修改底层值，而不会影响名称的实际意思。但是像.BlueAndItalic 这样的类名以后肯定会带来问题，如果以后决定将蓝色改为黑色怎么办呢？只好要么用不是描述自身行为的非常别扭的类名，要么就要重命名类，然后更新整个站点，将对旧类的引用改为.BlackAndItalic。
- 尽量创建一个较小而且可以重用的规则集。在必要时可以合并，而不是创建庞大完整的规则，只能在单个 UI 元素上使用。例如，不要创建这样的样式：

```
.ImportantHeading
{
    color: Red;
    font-size: 20px;
    font-weight: bold;
}
```

最好创建一些容易重用的轻量级规则：

```
h1
{
    font-size: 20px;
}

.Attention
{
    color: Red;
    font-weight: bold;
}
```

在将.Attention 类应用到标题<h1 class="Attention">时，将会得到与赋予它 ImportantHeading 类时

完全相同的行为。然而，使用单独的 **Attention** 类时，就创建了一个可重用的规则，可以应用到其他需要用户注意的元素上，如<p>或<span>元素。

## 3.5 本章小结

本章详细介绍了 CSS——样式化 ASPX 和 HTML Web 页面最重要的语言。

CSS 克服了使用 HTML 样式化 Web 页面时存在的局限性，因为它可以用来最小化页面膨胀，提供对页面外观更好的控制，而且通常有助于创建加载更快且更容易维护的 Web 站点。

很好地了解了 CSS 术语后，将会发现使用 VS 提供的很多 CSS 工具变得容易了。像 **Manage Styles** 和 **Apply Styles** 窗口、**Style Builder** 和代码编辑器中的智能 **IntelliSense** 等工具，都使 CSS 的编写和管理轻松愉快。

CSS 不仅可以像本章介绍的这样应用到 HTML 中，而且也可以应用到 ASP.NET 服务器控件上。应用到这些控件的 CSS 最终在浏览器中会显示为纯 HTML，这里采用的原理与本章提到的相同。第 4 章将详细介绍很多可用的 ASP.NET 服务器控件。

## 3.6 练习

1. 使用外部样式表优于内嵌样式表的地方主要是什么？
2. 写一个 CSS 规则，将站点中所有的一级标题(h1)的外观改为：
  - 标题使用 **Arial** 字体
  - 标题应是蓝色的
  - 标题必须是 18 像素的字体大小
  - 标题上边框和左边框应为蓝色细边

对于最后一个要求，可以在 CSS 文件中查看 VS 的 **IntelliSense** 列表来找到 **border** 属性的另一个缩略版本。

3. 在下面两个规则中，哪个规则比较容易在 Web 站点中跨页面重用？请解释原因。

```
#MainContent
{
    border: 1px solid Blue;
}

.BoxWithBorders
{
    border: 1px solid Blue;
}
```

4. VS 允许以多种不同的方法向页面中附加外部样式表。请指出其中 2 种不同方法。  
练习的答案见附录 A。

## 本章要点回顾

CSS	层叠样式表，在浏览器中布局 Web 页面的语言
CSS 方框模型	用于计算元素尺寸的模型，包括高度、宽度、内边距、边框和页边距
声明	为要应用该声明的元素决定样式的属性和值的组合
内嵌样式表	在页面的<style />元素中定义的 CSS 代码
外部样式表	在一个单独的文件中定义的 CSS 代码，之后通过<link />元素包括在页面中
内联样式表	使用 style 特性直接在元素上定义的 CSS 代码
规则集	一个选择器和一个或多个包含在一对花括号中的声明的组合
选择器	一个 CSS 结构，指向页面中的一个或多个元素。现在有多种选择器，包括全局选择器、ID 和类选择器以及元素选择器