.NET 开发系列丛书

C# 开发人员指南

——ASP.NET、XML、Web服务与ADO.NET

Jeffrey P. McManus　Chris Kinsman　著
常晓波　朱剑平　译

Addison Wesley

中国电力出版社
www.infopower.com.cn

.NET 开发系列丛书

C# 开发人员指南

——ASP.NET、XML、Web服务与ADO.NET

Jeffrey P. McManus Chris Kinsman 著
常晓波 朱剑平 译

中国电力出版社

内 容 提 要

本书由资深专家撰写，主要向 C#开发人员介绍如何利用 ASP.NET、XML 与 ADO.NET 开发互联网应用程序。全书共分 11 章，详细讲述了页面框架、调试 ASP.NET 应用、状态管理和缓存、Web 服务、安全、使用 XML 等精彩内容，并通过大量实例代码阐述相关概念，以方便读者的理解和学习。

本书适合有一定基础或实践经验的 C#开发人员以及对这一领域感兴趣的读者阅读。

C# Developer's Guide to ASP.NET, XML, and ADO.NET(ISBN 0-672-32155-6)

Jeffrey P. McManus, Chris Kinsman

图书在版编目（CIP）数据

C#开发人员指南—ASP.NET、XML、Web 服务与 ADO.NET/（美）麦曼斯，克兹曼编；常晓波，朱剑平译．—北京：中国电力出版社，2002

ISBN 7-5083-1384-4

Ⅰ.C... Ⅱ.①麦...②克...③常...④朱...Ⅲ.①C 语言－程序设计－指南②主页制作－程序设计 Ⅳ.TP312-62

中国版本图书馆 CIP 数据核字（2002）第 099008 号

责任编辑： 姚贵胜

书　　名： C#开发人员指南—ASP.NET、XML、Web服务与ADO.NET

编　　著： （美）Jeffrey P. McManus、Chris Kinsman

翻　　译： 常晓波、朱剑平

出版发行： 中国电力出版社

地址：北京市三里河路6号 邮政编码：100044

电话：（010）88515918 传真：（010）88423191

印　　刷： 汇鑫印务有限公司

开　　本： 787×1092 1/16　　**印　　张：** 26　　**字　　数：** 624千字

版　　次： 2003年4月北京第一版

印　　次： 2003年4月第一次印刷

定　　价： 45.00 元

关 于 作 者

Jeffrey P. McManus 是一名致力于 Microsoft 工具的开发人员和评论家。作为一名开发人员，他在使用 Internet 和客户/服务器技术开发在线应用程序方面有过专门的研究。他编写了四本有关数据库和组件技术的著作，其中包括非常畅销的《Database Access with Visual Basic 6》（Sams 出版社出版）。Jeffrey 定期在 VBITS/VSLive、European DevWeek 和 VBConnections 讨论会上发表演讲。

Chris Kinsman 是一名致力于 Microsoft 工具的开发人员和评论家。作为一名开发人员，他完全凭借 Microsoft 工具来负责几个高流量的站点，同时他还是 DevX.com 的技术副总裁。此外，Chris 花了 10 年的时间与全球财富 500 强公司协商以期通过利用 Microsoft Visual Studio 和 Back Office 技术来解决他们的需求。Chris 定期在 VBITS/VSLive、Web Builder 和 SQL2TheMax 讨论会上发表演讲。

关于撰稿人

Anjani Chittajallu 在印第安技术学院主修控制系统工程专业并获得硕士学位。她在使用 Microsoft 技术设计和开发企业系统方面有专门的研究。Anjani 目前持有 MCSD 认证。可以通过 srianjani@hotmail.com 与她获得联系。

关于技术审校

Joel Mueller 是 DeLani Technologies（www.delani.com，该公司在 Web 开发软件行业居领先地位）的高级软件工程师，从 2000 年 7 月起他就成为该公司 Microsoft .NET 开发工作的先锋。在 ASP.NET 诞生之前，Joel 使用 Microsoft ASP（Active Server Pages，活动服务器页）和 Macromedia ColdFusion 技术做过大量的工作。他编写了几本关于 Macromedia ColdFusion 和 XML 的著作。Joel 目前的兴趣包括.NET 框架、C#和睡觉。

献　　词

敬献：Celeste。

Jeffrey P. McManus

本书献给我的父亲，他对我做的所有事都给予了支持和鼓励。

Chris Kinsman

鸣　　谢

Jeffrey 和 Chris 特别感谢 Anjani Chittajallu，她在关键时刻加入我们的队伍并非常出色地编写了本书本版的代码示例。我们非常感谢你的帮助！

目 录

1 简介：对 ASP.NET 的需求

本章内容

在深入讨论使用 C#进行开发的细节之前，综述 ASP.NET 的整体特征会对读者很有帮助。本章概述了 ASP.NET 的特性，其中包括在 ASP 以前版本基础上所进行的改进。

1.1 当前 ASP 遇到的问题

当 5 年前首次引入 ASP（Active Server Pages，活动服务器页）时，ASP 被看做是当时用于在 Web 上创建动态内容的笨拙技术的替代方法。那时 CGI（Common Gateway Interface，通用网关接口）程序或专用服务器插件是多数 Web 动态内容的创建方法。随着 ASP 1.0 的发布，Microsoft 改变了一切。ASP 1.0 提供了灵活健壮的脚本体系结构，使开发人员能够快速地创建动态 Web 应用程序。开发人员可以使用 VBScript 或 JScript 来编写脚本，Microsoft 还提供了大量的服务使开发工作更轻松。那时，ASP 1.0 正是开发人员所需要的开发工具。随着 Web 开发的成熟，该平台的几个缺点逐渐暴露出来，并且这种状况一直持续到今天。

1.1.1 代码与设计分离

随着 20 世纪 90 年代早期 Web 的逐渐流行，开发人员经历了开发范例的三次异乎寻常的浪潮。在第一次浪潮中，Web 开发人员创建静态 HTML 文档并将这些文档链接在一起。这是“小册子”Web 站点时代，除了看之外，什么也没有。第二次浪潮则为前卫的站点带来了动态内容的概念。开发人员开始创建注册表单和各种小的功能块并将它们添加到已有的 Web 站点中。第三次浪潮是在前两次的基础上形成的。Web 站点从底层的设计就注重交互性，当时的 Web 站点更像是一个应用程序，而很少像是包含订阅卡的杂志。在多数实例中，这种类型的交互式页面设计创建了类似这样的开发范例：

- 设计人员在 HTML 中创建页面模型。
- 设计人员向页面中添加代码。
- 当设计人员需要更改设计时，他们将已有的代码复制并粘贴到新的页面中，这就破坏了代码的原有功能特性。

这个问题的严重程度一般依赖于站点的规模、设计人员的智慧以及开发人员为避免这种

破坏所采用的技术。

随着 1998 年 9 月 Visual Studio 6 的发布，很明显，Microsoft 意识到了这个潜在的问题，并打算利用 Visual Basic 6 中的新增特性 Web Classes 来解决这个问题。Web Classes 试图将页面的设计与影响页面的代码分离。通过使用 HTML 模板实现了这种分离，并为在模板中进行标记（tag）替换提供了支持。Web Classes 存在着很多的问题，尽管 Web Classes 是一个很不错的想法。这些问题主要表现两个方面。首先，Web Classes 在 Visual Basic 中是完整实现的，这就要求传统的 ASP 开发人员为创建应用程序而转变思维模式。其次，Microsoft 在与 ASP 和 Visual Basic 线程模型有关的伸缩性方面遇到了问题。由于前面所述的原因以及其他许多更小的因素，Web Classes 实际上并没有得到开发人员的普遍支持。

1.1.2 基于脚本的语言

当 ASP 1.0 首次发布时，所有的开发工作都可以使用脚本语言来完成了，这确实是一个巨大的进步。这意味着开发人员不必像他们已经习惯的使用 CGI 或 ISAPI 风格的应用程序那样经历痛苦的重启/编译过程。随着应用程序的增大、用户数量的增多，开发人员开始使用 ASP 来对付越来越多的复杂问题。所有代码都需解释的事实成为潜在的性能瓶颈。当使用 VBScript 时，对错误处理仅有有限的支持。许多开发人员通过将代码移到已编译的 COM 对象中来回避这个问题。尽管这种代码转移解决了一些性能问题，但在部署和可伸缩性方面也产生了新的问题。

1.1.3 状态管理

早期 Web 开发新手所面对的最大难题之一是要处理 Web 开发的无状态特性。借助 ASP 1.0，Microsoft 引入了 Session 对象的概念，该对象的目的是轻松地将状态与特定用户相关联。这个新增特性无可置疑地成为 ASP 1.0 最引人注目的特性之一。当开发人员开始创建更大的应用程序时，可伸缩性和可靠性也就开始变得很重要。为适应这种需要，开发人员开始将应用程序部署为 Web 群（Web farm）。Web 群使用多个服务器并几乎均等地为各个服务器的页面传播请求。这有利于可伸缩性，除非开发人员使用了很“酷”的 Session 对象。该对象对应于 Web 群中的特定计算机，如果用户跳到另一台服务器上，则该对象不会起作用。因此，部署到 Web 群的应用程序不能使用 Session 对象。

1.2 ASP.NET 简介

ASP.NET 是 Microsoft 对前面讲述的和许多尚未明确指出的问题的解决方案。这是对已经服役了两年多的 ASP 的根本性改写。ASP 团队密切关注 Web 开发人员所面临的问题，并以传统 ASP 的精髓创建了一个全新的平台来解决这些问题。经过长时间地使用 ASP.NET，我们最后得出结论：这一版本达到了预定的设计目标。

1.2.1 平台体系结构

ASP 的以前版本是一种 ISAPI（Internet Server Application Programming Interface，Internet

服务器应用程序编程接口）过滤器，专门用来与 IIS（Internet Information Server，Internet 信息服务器）交互。ASP 以前版本在本质上是单一独立的，很少依赖于外部服务。

注意：

在 IIS 5.0 时代，ASP 使用 MTS（Microsoft Transaction Server，Microsoft 事务服务器）作为外部服务。

ASP.NET 仍然是一种 ISAPI 过滤器。然而与 ASP 以前版本不同的是，ASP.NET 依赖于大量的"外部"服务：.NET 框架。ASP.NET 和.NET 框架结合得如此之紧密，以至于很难将.NET 框架看做是外部服务。然而，既然可以从 ASP.NET 范畴之外的应用程序进行访问，所以.NET 框架还是应该被看做是"外部"服务。因此，.NET 框架的诞生是 ASP.NET 开发人员的巨大胜利。开发人员不再必须从零开始编写所有的功能模块，取而代之的是，.NET 框架提供了一个非常巨大的预先编写好的功能库。

可重新发布的.NET 框架由以下三个主要部分组成：CLR、.NET 框架基类和 ASP.NET。

1．公共语言运行库

CLR（Common Language Runtime，公共语言运行库）是.NET 框架应用程序的执行引擎。然而，尽管通常被误解，但 CLR 不是解释器。.NET 应用程序是完全已编译的应用程序，在执行过程中使用 CLR 提供大量的服务。这些服务包括：

- 代码管理（加载和执行）
- 应用程序内存隔离
- 类型安全验证
- IL 到本地代码的转换
- 对元数据的访问
- 垃圾收集
- 强制代码访问安全性
- 异常处理
- 互用性
- 自动对象布局
- 支持调试和分析

CLR 是从操作系统中抽取功能函数的平台。从这种意义上讲，如果目标平台上具有 CLR 的实现，那么针对 CLR 编写的代码就是"平台无关"的。

2．受管执行

CLR 不仅仅是可被执行程序调用的函数库或函数框架。CLR 可在许多级别上与运行代码进行交互。CLR 提供的加载器在每次加载程序段时执行验证、安全检查和许多其他任务。内存的分配和访问也由 CLR 控制。当你听到"受管执行"（Managed Execution）时，这是人们在谈论 CLR 和执行代码之间的交互，这种交互用以创造可靠的应用程序。

3．跨语言的互用性

对当前基于 COM 或基于 API 的开发实践最令人受挫的一点是，接口通常采用特殊的语言编写。当编写由 Visual Basic 程序使用的组件时，开发人员创建接口的方式与用 C++创建接口的方式不同。这意味着要实现这两种组件，开发人员必须使用一种公共的方法来开发接

口或者必须为每类用户分别开发接口。这显然不是最有效的编写组件的方法。当今多数开发人员感到很勉强的第二个问题是多数组件需要采用单一的语言来编写。如果使用 C++来创建表示 employee 对象的组件，就不能从 Visual Basic 编写的对象中继承其特性来创建 Developer 对象。这意味着对于多数开发项目来说，为使对象可以重用，一般都选择使用单一的语言。

.NET 改变了这一切。跨语言的互用性是从头开始创建的。所有的.NET 语言都必须符合 CLS（Common Language Specification，公共语言规范），该规范规定了每种语言必须实现的底层功能函数，以便相互之间的通信。CLS 以这样的方式进行编写：在 CLR 中每种语言都保持其独特风格，但仍然能够与其他语言正确地互操作。CLS 包括大量的数据类型，所有遵守此规范的语言都必须支持这些数据类型。这种限制的作用是为开发人员排除共同的问题：创建接口所使用的数据类型不被其他语言支持。它还支持二进制和源代码级的继承，使开发人员能够在 C#中创建 Employee 对象，并在 Visual Basic 中继承该对象。

这对开发人员来说意味着代码重用变得简单多了。只要代码是针对.NET 编写的，就不需要考虑是用什么语言编写的。实际上，语言的选择更多地变成了生活方式的选择，而不是能力的选择。.NET 中所有的语言理论上都是同等创建的，因此使用 Visual Basic 来代替 C# 并不会带来性能或功能上的益处。你可以选择对自己来说最高效的语言。

1.2.2 ASP.NET 中的新增特性

前面所提到的所有特性都源自.NET 框架顶层的 ASP.NET 宿主。然而，这些特性只是冰山的一角。如前所述，ASP.NET 是一个完全重写的版本，具有来自.NET 框架固有的重要特性。下面将扼要地讲述 ASP.NET 中的新增特性，同时说明如何使用这些特性来解决代码和设计的分离、脚本语言和状态管理等方面的问题。

1. Web 表单

Web 表单是 Microsoft 对解决代码和设计分离问题的尝试。ASP.NET 现在提供一种 code-behind（代码后置）模型，这使人联想起 Visual Basic 中的窗体设计器。这意味着现在可以在独立的文件中放置代码，而同时还能保持页面的交互性。这种独立是这样实现的：在页面执行的顶部提供新的事件驱动模型，同时在该页面的 HTML 顶部提供对象模型。不是采用自顶向下的线性执行模型，事件是在页面的执行过程中引发的。你的代码将接收这些事件并通过与位于 HTML 顶部的对象模型交互来响应事件。

这非常巧妙地解决了设计人员修改 HTML 和中断代码的问题。

除了新的执行模型之外，Microsoft 还创建了新的服务器端控件模型。与 Visual Interdev 中功能不够完整的 Design Time Controls（设计阶段控件）不同，这些新的模型是对公共显示范例的封装非常有用。它们不会依赖于任何浏览器，并在服务器端运行，而不是在客户端。在少数产生浏览器相关代码的情况下，它们可嗅探到浏览器并恰当地降级。第 2 章“页面框架”中介绍了有关 Web 表单的更多信息。

2. Web 服务

在第一代和第二代 Web 应用程序之后，有一点变得很明显，那就是可以扩展 Web 范例来解决在范例形成之前的问题。过去企业使用 VAN（Value Added Networks，增值网络）上的 EDI（Electronic Data Interchange，电子数据交换）来交换信息。这种方式工作得很好，但是访问 VAN 的成本以及实现各种行业特定的 EDI 协议的复杂性阻碍了几乎所有的企业参与

信息交换，只有那些最大的公司除外。

Web 服务是一种使用工业标准的协议（例如 HTTP、XML 和 TCP/IP）在 Internet（而不是昂贵的 VAN）上传输相同类型信息的方法。Web 服务在.NET 中十分易于创建，而且.NET 适用于任何规模的个体和公司。Web 服务并不只限于代替传统的 EDI 协议。它们还提供了 EDI 所从未提供的机会。有关详细信息，请参阅第 6 章“Web 服务”。

3．数据访问

第一次发布 ASP 1.0 时，Microsoft 的数据访问历史正处于不断改变之中。那时，RDO（Remote Data Objects，远程数据对象）是为 Visual Basic 开发人员提供的技术选择。ADO（ActiveX Data Objects，ActiveX 数据对象）则随着 1997 年 2 月 Visual Basic 5.0 的发布而引入。Microsoft 计划将 ADO 作为新的针对所有类型数据的数据访问模型，并与另一种新技术 OLE DB 结合使用。

尽管 ADO 能够很好地实现其设计目的，即互联的数据访问，但在非连接的场合下却存在缺陷。在后续版本中的新增特性允许 ADO 以非连接的方式工作。ADO 1.0 不支持 XML。在发布 ADO 1.0 时，Microsoft 并不能预料到 XML 作为数据描述语言的成功，因此 XML 支持也只是在后续版本中加入的。所有这些特性都不是一开始就设计好的。

ADO.NET 是一种新的数据访问技术，充分利用了 Microsoft 从 ADO、RDO、OLEDB、ODBC 和其他早期数据访问实现中积累的所有经验。ADO.NET 从一开始就被设计成与 XML 紧密结合，并且能够以非连接的方式高效工作。有关详细信息，请参阅第 11 章“使用 ADO.NET 创建数据库应用程序”。

4．部署

ASP 开发人员一直争论不休的一个问题是有多少代码要迁移到 COM 对象中。一些作者提倡所有的代码都驻留在 COM 对象中，而 ASP 仅应该包含单一方法调用来调用 COM 对象。尽管这些在理论上非常美妙，但它抛弃了 ASP 一个最大的功能：快速创建应用程序和即时更改的能力。由于所有逻辑和 HTML 都与 COM 对象密切相关，所以一个简单 HTML 标记的更改都会要求重新编译和重新部署 COM 对象。我们心中最大的问题都是由于使用这种方法而引起的。COM 对象是由 IIS 动态加载的 DLL（Dynamic Link Libraries，动态链接库）。加载时，COM 对象不能被替换。要部署开发人员用以关闭 IIS 的 COM 对象，需要关闭该 COM 对象所在的 MTS 包，替换它，然后重新启动 IIS。本摘要实际上是该过程的简化，但可以看出这种技术所带来的问题。每当部署新的版本时，Web 服务器就必须关闭。该技术引起的停机时间可通过创建 Web 群和执行滚动升级来控制；然而在大型 Web 群中，这意味着在首次展示新对象时，一个简单的更改都要花费数小时的时间来部署。

由于代码后置模型是 ASP.NET 所固有的，因此上述情况似乎会更加严重。然而事实却恰恰相反，Microsoft 极大地简化了部署模型。组件（现在叫做部件，assembly）不再需要在计算机上注册以进行部署。部件是 COM 对象的.NET 替代物。部件是自描述性的，包含一个清单，其中包含了有关该部件的元数据。该元数据包含着诸如版本、所依赖的部件以及潜在的安全标识之类的信息。

部署就像将部件复制到应用程序根目录的/bin 目录中一样容易。ASP.NET 会注意到已复制了新的版本，然后卸载旧版本并加载新版本。部署变得与 XCOPY /S 或将新文件上传到 Web 服务器的递归 FTP 一样容易。有关详细信息，请参阅第 5 章“配置和部署”。

5. 配置

在过去，所有的 ASP 配置信息都存储为 IIS 元数据的一部分。这是一个类似于注册表的二进制文件，包含着 IIS 和 ASP 的所有设置。影响更改的惟一方法原本是使用 Internet 服务管理器（Internet Services Manager）或利用 ADSI（Active Directory Services Interfaces，Active Directory 服务接口）编写脚本来使更改自动化。这种方法使得对 Web 群中多个服务器的设置进行同步变得很困难。

ASP.NET 针对所有设置引入了一种新的范例。不是将其存储在不透明的元数据中，现在配置信息被存储为 XML 配置文件的分级集合。这些文件驻留在应用程序根目录和子目录中。因此，现在开发人员使用 XCOPY 来部署源文件时，这些设置也就同时部署好了！不再需要编写长长的配置脚本了。有关详细信息，请参阅第 5 章。

6. 状态管理

状态管理在 ASP.NET 中得到了极大的改进。现在，要保持服务器上的状态可以有三种选择方案。在单一服务器上保持内存中状态这一 ASP 3.0 中的经典方法仍然存在。此外，在 SQL Server 中还提供进程外状态服务器和持久状态的选项。

ASP 以前版本中状态服务的其他限制是为将用户连接回到他们的状态而产生的对 cookie 的依赖。ASP.NET 为无 cookie 状态引入了一种新的选择方案，该方案执行 URL 以便将用户连接至状态信息。有关详细信息，请参阅第 4 章“状态管理和缓存”。

7. 缓存

多数开发人员使用 ASP 的原因是要将动态特性应用到 Web 上。这本来需要访问后端数据库数据或将数据从非传统的后端拉入。该动态内容的问题是尽管人们可以使用在多个 Web 服务器上应用的收缩方法来轻松地缩放 Web 层，但这种缩放没有在数据层中那么容易。数据库收缩的方法只是刚刚出现。在这些方法趋于完美之前，Web 开发人员如何缩放应用程序呢？

对那些经常更改的数据来说，缓存是一种很好的解决方案。ASP.NET 提供两种形式的缓存。输出缓存获取完整的页面并将执行的结果存储在内存中以便以后传送。数据缓存获取那些难以创建的项目（例如 DataSet）并将其缓存在服务器端。有关详细信息，请参阅第 4 章。

8. 调试

调试 ASP 应用程序一直非常困难。尽管在 Visual Studio 的以前版本中提供了远程调试解决方案，但只有极少数开发人员能够使之始终如一地工作。因此，多数调试由 Response 语句组成。Write 语句遍布于整个代码中或者用于开发人员所创建的某些类型的日志机制中。

ASP.NET 不仅改进了远程调试，还使之相容。ASP.NET 还提供了新的强大的 Trace 工具，用以处理日志或 Response 之类的事务。过去使用的是 Write。有关详细信息，请参阅第 3 章“调试 ASP.NET 应用程序”。

9. 可用性

任何目前在 Web 上拥有站点的人都知道站点的可用性是关键。如果你的站点当机了，那么访问者将可能转到其他许多站点上。更大的问题是：他们可能永远也不会回来！

ASP.NET 引入了大量的进程控制，其目标直接面向提高可用性。现在，ASP.NET 进程可以基于诸如内存使用率、特定时刻甚或请求句柄数目等原因而自动重新启动，这将有助于处理那些 ASP 过去常常造成堵塞的情况。有关详细信息，请参阅第 5 章。

2 页面框架

本章内容

编写 ASP.NET 应用程序与 ASP 编程有很大的不同。这种不同就像从 QuickBasic 编程变为 Visual Basic 编程一样。

ASP.NET 中的变化可以分成三类：控件模型、事件模型以及代码与表现分离。

2.1 ASP.NET 的控件模型

在 QuickBasic 中，处理输出屏幕的代码就像一篇很长的论文。你可能曾经用过屏幕库，它封装了许多这样的功能并用更好的配置使之成为一个更高层的操作。

随着 Visual Basic 的出现，我们进入了一个可以重用控件的世界。通过把控件拖放到一个设计界面中来设计 UI，而不是将文本输出到屏幕或在屏幕上画多边形。

这些控件是对象，它们具有用于定制它们的外观和功能的方法和属性。ASP 的以前版本在许多方面类似于 QuickBasic。整个页面本质上被看成一篇长长的用于放置代码的纸张。没有任何对象模型可用于访问代码周围的 HTML——对你来说，只是一个基于你的代码位置来输出附加 HTML 的方法。

ASP.NET 通过引入服务器控件的概念来使之改观。如果你曾用 Visual Interdev 来创建 ASP Web 应用程序，那么你可能会这样想："太妙了！它们只是重命名了那些繁重的设计时控件"。事实并非如此。服务器控件不是另一种形式的设计时控件。服务器控件也不需要任何特殊类型的客户端，换句话讲，服务器控件不是 ActiveX 控件或者客户端行为。服务器控件是在页面执行期间用于在页面上放置用户界面元素的功能的高层抽象。

下面看一下清单 2.1，其中显示了一个 ASP 以前版本的表单的 HTML。

清单 2.1 一个简单的 ASP 以前版本的页面，SimplePage.asp

```
<html>
<head>
    <title>SimplePage.asp</title>
</head>
```

```
<body>
    <form name="WebForm1" method="post">
        <p>
        <table border=0>
            <tr>
                <td>Name:</td>
                <td><input type=text name=txtName></td>
                <td><input type=submit name=Button1 Value="Send"></td>
            </tr>
            <tr>
                <td valign=top>Hobby:</td>
                <td>
                    <select name=lbHobbies Multiple>
                        <option Value="Ski">Ski</option>
                        <option Value="Bike">Bike</option>
                        <option Value="Swim">Swim</option>
                    </select>
                </td>
                <td> </td>
            </tr>
        </table>
        </p>
    </form>
</body>
</html>
```

当用户填写名字，选择了一个爱好（hobby），然后按下 Send 按钮时发生了什么？该页面先被发送回服务器。此时表单中没有代码，所以用户在 Select 标记中所做的选择[我们称为表单状态（form state）的信息]都丢失了。该页面然后被返回给浏览器。在 ASP 以前版本中，如果想保存表单状态，就必须编写代码来完成表单状态的保存工作。

清单 2.2 包含 SimplePage2.asp，它显示了使用 ASP 以前版本完成此工作所需编写的典型代码。

清单 2.2 用于在 ASP 以前版本中保存表单状态的 SimplePage2.asp

```
<html>
<head>
    <title>SimplePage2.asp</title>
</head>

<SCRIPT LANGUAGE="VBScript" RUNAT=SERVER>
    function IsOptionSelected(strControlName, strOption)
        for iCount = 1 to Request(strControlName).Count
            if request(strControlName)(iCount) = strOption then
                response.write " SELECTED "
            end if
        next
    end function
```

```
</SCRIPT>

<body>
    <form name="WebForm1" method="post">
        <p>
        <table border=0>
            <tr>
                <td>Name:</td>
                <td><input type=text name=txtName value="<% = Request("txtName")
%>"></td>
                <td><input type=submit name=Button1 Value="Send"></td>
            </tr>
            <tr>
                <td valign=top>Hobby:</td>
                <td>
                    <select name=lbHobbies Multiple>
                        <option <% IsOptionSelected "lbHobbies", "Ski" %>

Value="Ski">Ski</option>
                        <option <% IsOptionSelected "lbHobbies", "Bike" %>
Value="Bike">Bike</option>
                        <option <% IsOptionSelected "lbHobbies", "Swim" %>
Value="Swim">Swim</option>
                    </select>
                </td>
                <td> </td>
            </tr>
        </table>
        </p>
    </form>
</body>
</html>
```

随着服务器控件的出现，ASP.NET 向 HTML 拥有的用户界面控件里添加功能，让它们完成你希望的工作，即保存用户刚才费时输入的数据。

你需要做三件事情来使 ASP.NET 服务器控件工作：

（1）ASP.NET 服务器控件是用 ID 属性而不是（或外加）Name 属性来识别的。可以两个都用。如果有需要引用该控件的客户端脚本时，就应该使用 Name 属性。

（2）ASP.NET 服务器控件要求添加 runat=server 属性。这个属性向 ASP.NET 表明此标记不是一个内置的 HTML 标记。

（3）ASP.NET 服务器控件需要一个结束标记。服务器控件使用 XML 名称空间来实现，这很像 XML，它要求每个元素都要有一个匹配的封闭元素。你可以使用 XML 风格的语法作为创建标记的快捷方式，例如<input type=text runat=secver />。

清单 2.1 中的代码就是这样做的。清单 2.3 显示了 simplepage.aspx，它是 simplepage.asp 的 ASP.NET 实现。

清单 2.3 SimplePage.aspx——在 ASP.NET 中对清单 2.1 的改写

```
<html>
<head>
    <title>SimplePage.aspx</title>
</head>

<body>
    <form id="WebForm1" method="post" runat="server">
        <p>
        <table border=0>
            <tr>
                <td>Name:</td>
                <td><input type=text id=txtName runat=server /></td>
                <td><input type=submit id=Button1 Value="Send" runat=server /></td>
            </tr>
            <tr>
                <td valign=top>Hobby:</td>
                <td>
                    <select id=lbHobbies Multiple runat=server>
                        <option Value="Ski">Ski</option>
                        <option Value="Bike">Bike</option>
                        <option Value="Swim">Swim</option>
                    </select>
                </td>
                <td> </td>
            </tr>
        </table>
        </p>
    </form>
</body>
</html>
```

所做的改变是分别在 form 标记、input 标记和 select 标记中增加了 runat=server 属性。我们也将每个 name 属性更改为 ID 属性。就做了这些。如果运行该页面，填写名字，选择一个爱好，然后单击 Send 按钮，那么在此页面被撤消并在到服务器的往返程中被重新创建后，你所输入的数据仍然保存在那里。服务器控件知道所希望的默认行为是保持输入，也就是说，这些控件保持所输入数据的状态，并且这种行为是自动进行的。

如果不希望某个服务器控件保持自己的状态，则可以使用一个新的称为 EnableViewState 的属性，任何服务器控件均具有该属性。通过将该属性设置为 false，可以覆盖张贴后保持表单状态这一默认行为。

两类服务器控件分别是 HTML 控件和 Web 控件。HTML 控件反映 HTML 部分。HTML 控件包含下列内容：

HTML 控件类	HTML 标记
HtmlAnchor	锚