

"九五"国家重点电子出版物规划项目·希望计算机知识普及系列
体验办公新工具丛书(2)



Microsoft

ASP

Microsoft
.net

北京希望电子出版社 总策划
何志学 编 写

深入 编程

深入编程系列电子书



北京希望电子出版社
Beijing Hope Electronic Press
www.bhep.com.cn

“九五”国家重点电子出版物规划项目·计算机知识普及系列
体验办公新工具丛书(2)

Microsoft ASP .NET 深入编程

北京希望电子出版社 总策划
陈英学 编写

本书特点

- 容易理解和掌握, 范例丰富
- 讲练结合, 内容精心设计
- 事半功倍, 思考周到、体贴

适合对象

- 面向初级和中级层面读者
- 高级用户、高等院校相关专业师生参考
- 科技图书馆的馆藏书

本盘内容包括本版电子书



北京希望电子出版社

Beijing Hope Electronic Press

www.bhp.com.cn

2001

内 容 简 介

这是一本专门介绍如何用微软最新网络开发工具 ASP.NET 进行深入编程的书。全书由 15 章构成，从 ASP.NET 的基础知识到高级应用，作者从始至终通过大量的范例教授用 ASP.NET 进行各各层次的编程方法和技巧。

各章主要内容包括：第 1 章 ASP.NET 的发展战略和发展概况；第 2 章环境与 Visual Studio.NET，介绍 .NET 的安装和运行环境、Visual Studio.NET 的使用；第 3 章 ASP.NET 的语言基础。介绍 C# 的语法；第 4 章 Web Form 页面，介绍 Web Form 的概念、应用 相关的例子讲解；第 5 章 ASP.NET 的验证，介绍在 ASP.NET 技术中应用广泛的验证技术；第 6 章服务器端控件，介绍服务器控件技术；第 7 章自定义与 HTML 控件：介绍自定义控件和 HTML 控件技术，是深入了解 ASP.NET 的基础；第 8 章 ASP.NET 的数据库编程，介绍数据库编程的基础、ADO.NET 数据库编程的基础、ADO.NET 数据库基本操作、Dataset 的用法和数据绑定技术；第 9 章应用程序，先介绍如何配置 config.Web，如何编写 global.asax，再结合一个实例“会员系统”来对应用程序进行深入的讲解；第 10 章 Web Service，对 Web Service 进行了详细的讲解。通过例子学习 Web Service。同时，深入地讲解了数据交换和存取站点对象；第 11 章性能优化，ASP.NET 有两种用于 Web 应用的缓冲技术：对输出缓冲和数据缓冲进行详细的讲解；第 12 章高级应用，介绍 XML 在 ASP.NET 中的应用、三层结构及其应用、以及微软消息队列 (MSMQ)；第 13 章 ASP.NET 实战，大量的例子，各种情况的综述，相关的技巧；第 14 章延伸，向 Window 编程发展，举一个例子进行讲述；第 15 章网上资源：网上的相关的 ASP.NET 的资源。

本书特点：范例丰富，将 ASP.NET 的功能与实际编程相结合，边讲边练，即学即用。内容精心设计，由浅入深，层层深入地讲解了 ASP.NET 技术，例子使用 VB.NET 的语法。对 VB 的语法或对 ASP 很熟悉的读者将很快就得上手，精通其他编程语言得读者也会及时跟上编程技术发展的前沿。思考周到，光盘中包含全书应用到的所有例子的源代码，这些例子在书中也有相应的说明，读者在学习有事半功倍。适合对象，本书面向初级和中级层面的读者，对高级用户、高等院校相关专业师生也有重要的参考作用，同时也可作为科技图书馆的馆藏书。本光盘内容包括本版电子书。

系 列 盘 书：“九五”国家重点电子出版五规划项目·计算机知识普及系列 体验办公新工具丛书(2)

盘 书 名：Microsoft ASP.NET 深入编程

总 策 划：北京希望电子出版社

文 本 著 者：陈英学

C D 制 作 者：希望多媒体开发中心 曾冬梅

C D 测 试 者：希望多媒体测试部

责 任 编 辑：文华

出 版、发 行 者：北京希望电子出版社

地 址：北京中关村大街 26 号，100080

网址：www.bhp.com.cn E-mail：lwm@hope.com.cn

电话：010-62562329, 62541992, 62637101, 62637102, 62633308, 62633309 (图书发行)

经 销：010-62613322-215 (门市) 010-62613322-334 (编辑部)

各地新华书店、软件连锁店

排 版：希望图书输出中心 杜海燕

C D 生 产 者：北京中新联光盘有限责任公司

文 本 印 刷 者：北京广益印刷厂

开 本 / 规 格：787 毫米 × 1092 毫米 1/16 开本 23.75 印张 563 千字

版 次 / 印 次：2001 年 7 月第 1 版 2001 年 7 月第 1 次印刷

印 数：0001—8000 册

本 版 号：ISBN 7-900071-74-1/TP·73

定 价：38.00 元（1CD，含配套书）

说明：凡我社光盘配套图书若有自然破损、缺页、倒页、脱页，本社负责调换。

声 明

本电子版不包括第 8 章内容，请参看配套图书相关章节。

前 言

感谢你选购《ASP.NET 深入编程》，本书包括了.NET 技术的很多内容，它会引导你在 ASP.NET 的技术中遨游。

.NET 是微软公司推出的全新的概念，而 ASP.NET 就是把.NET 变为现实的一种手段之一。

阅读本书之后，你将会对 ASP.NET 有比较深入的了解，并且完全能胜任用 ASP.NET 技术来开发项目。

本书面向初级和中级层面的读者，对高级用户也有参考作用。它全面系统地介绍了 ASP.NET 的特点、基础知识和具体的应用技术。如果你对.NET 技术很熟练，本书将给你更多的启示。

本书用丰富的范例，由浅入深，层层深入的讲解了 ASP.NET 技术，在本书中写作中，例子使用了 VB.NET 的语法。所以，如果对 VB 的语法或对 ASP 很熟悉的话，你将很快就可以上手。如果你精通别的编程语言，通过对本书的阅读，也会让你跟上编程技术发展的前沿。

本书结构

全书共分为 15 章，从 ASP.NET 的基础知识到高级应用。

第 1 章概述：介绍了.NET 战略、ASP.NET 的发展；

第 2 章环境与 Visual Studio.NET：介绍.NET 的安装和运行环境、Visual Studio.NET 的使用；

第 3 章 ASP.NET 的语言基础：介绍 C# 的语法；

第 4 章 Web Form 页面：介绍 Web Form 的概念、应用，相关的例子讲解；

第 5 章 ASP.NET 的验证：介绍在 ASP.NET 技术中应用广泛的验证技术 特别在 Web Form 是非常有用的。

第 6 章服务器端控件：介绍服务器控件技术；

第 7 章自定义与 HTML 控件：介绍自定义控件和 HTML 控件技术，是深入了解 ASP.NET 的基础；

第 8 章 ASP.NET 的数据库编程：介绍数据库编程的基础、ADO.NET 数据库编程的基础、ADO.NET 数据库基本操作、Dataset 的用法和数据绑定技术；

第 9 章应用程序：先介绍如何配置 config.Web，如何编写 global.asax，再结合一个实例“会员系统”来对应用程序进行深入的讲解。

第 10 章 Web Service：对 Web Service 进行了详细的讲解。通过例子学习 Web Service。同时，深入地讲解了数据交换和存取站点对象，本章是微软.NET 的一个重点之一。

第 11 章性能优化：ASP.NET 有两种用于 Web 应用的缓冲技术：输出缓冲和数据缓冲，

在本章中将围绕的这两种缓冲技术，进行详细的讲解。

第 12 章高级应用：在高级应用一章中，我们将介绍三个方面的内容：XML 在 ASP.NET 中的应用、三层结构及其应用、以及微软消息队列（MSMQ）。

第 13 章 ASP.NET 实战：大量的例子，各种情况的综述，相关的技巧；

第 14 章延伸：向 Window 编程发展，举一个例子进行讲述；

第 15 章网上资源：网上的相关的 ASP.NET 的资源。

学习本书需要使用的工具

本书附有光盘，光盘中包含全书应用到的所有例子的源代码，这些例子在书中也有相应的说明。

你需要配置一个运行环境来运行这些代码，具体的要求看书中的详述。

阅读本书不要求安装 Visual Studio.NET 开发工具，安装好 FrameWork SDK 调试环境即可。

本书由陈英学、贺英、刘林、赵庆勇、刘杰珍、吴永朝、刘靖非、周勇等编写。由于时间仓促，书中难免有不足之处，敬请广大读者、专家提出宝贵意见。

编者

2001 年 6 月

目 录

- 第 1 章 概述
- 第 2 章 环境与 Visual Studio.NET
- 第 3 章 ASP.NET 的语法基础
- 第 4 章 Web 页面
- 第 5 章 深入讲解 ASP+验证
- 第 6 章 服务器端控件
- 第 7 章 自定义与 HTML 控件
- 第 8 章 .NET 的数据库编程技术
- 第 9 章 应用程序
- 第 10 章 Web Service
- 第 11 章 性能优化
- 第 12 章 高级应用
- 第 13 章 ASP.NET 实战篇
- 第 14 章 C#的 Windows 编程
- 第 15 章 附录

第 1 章 概 述

1.1 微软的 .NET 战略

未来的 IT 世界将是一个以网络为中心的世界。面对这个即将来临的网络世界，各大软件公司都制订了自己的计划，而最近全球最大的软件公司微软公布了它的网络战略计划，这就是 .NET 计划。微软的计划将会为公众提供更加丰富、有用的网络资源与服务。

微软公司称 Microsoft .NET 的基本思想是：

侧重点从连接到互联网的单一网站或设备上，转移到计算机、设备和服务群组上，使其通力合作，提供更广泛更丰富的解决方案。用户将能够控制信息的传送方式、时间和内容。计算机、设备和服务将能够相辅相成，从而提供丰富的服务，而不是像孤岛那样，由用户提供唯一的集成。企业可以提供一种方式，允许用户将它们的产品和服务无缝地嵌入自己的电子构架中。这种思路扩展了二十世纪八十年代首先由 PC 赋予的个人权限。

Microsoft .NET 将开创互联网的新局面，基于 HTML 的显示信息将通过可编程的基于 XML 的信息得到增强。XML 是经“万维网联盟”定义的受到广泛支持的行业标准，Web 浏览器标准也是由该组织创建的。微软公司为开发它投入了大量精力，但它并不是 Microsoft 的专有技术。XML 提供了一种从数据的演示视图分离出实际数据的方式。这是新一代互联网的关键，提供了开启信息的方式，以便对信息进行组织、编程和编辑；可以更有效地将数据分布到不同的数字设备；允许各站点进行合作，提供一组可以相互作用的“Web 服务”。

Microsoft .NET 构成

Microsoft .NET 平台：包括用于创建和操作新一代服务的 .NET 基础结构和工具；可以启用大量客户机的 .NET User Experience；用于建立新一代高度分布式的数以百万计的 .NET 积木式组件服务；以及用于启用新一代智能互联网设备的 .NET 设备软件。

Microsoft .NET 产品和服务：包括 Windows.NET，连同建立积木式服务的核心集成套件；MSNTM .NET；个人订购服务；Office.NET；Visual Studio .NET；以及用于 .NET 的 bCentralTM。

在 .Net 环境中的突破性改进包括下面一些内容：

- ☞ 使用统一的 Internet 标准（如 XML）将不同的系统对接；
- ☞ 这是 Internet 上首个大规模的高度分布式应用服务架构；
- ☞ 使用了一个名为“联盟”的管理程序，这个程序能全面管理平台中运行的服务程序，并且为它们提供强大的安全保护后台。

.NET 平台包括如下组件：

- ☞ 用户数据访问技术。其中包括一个新的基于 XML 的、以浏览器为组件的混合信息架构，叫做“通用画板”；
- ☞ 基于 Windows DNA 2000 的构建和开发工具；

一系列模块化的服务，其中包括认证、信息传递、存储、搜索和软件传递功能；

最后还包括一系列驱动客户设备的软件；

微软终端用户产品以及服务将会按照提供的架构服务标准化，微软的架构服务将会以新一代 Windows 用户平台——Windows.Net 为主导。

对于开发者来讲，Visual Studio.NET 将提供一个基于 XML 的编程模型以及快速开发环境。

基于 .Net 平台，微软的 bCentral 小企业门户可以得到更多的扩展，并且能提供更广泛的服务。比如在线的分类目录服务以及相关的电子商务服务，还有网络化的客户关系管理工具，以及 Web 版本 Outlook 对电子邮件和个人信息进行管理的服务。

最后，MSN 成员的互联网门户服务也将转变为 MSN.Net，其中将会为 MSN 成员提供多种访问途径和访问选择。现在根据微软公司透露的情况，MSN.Net 已经开始进行他自己的集成客户端的 BETA 测试。

1.2 ASP.NET 的发展

ASP 的第一个版本是 0.9 测试版。它给 Web 开发带来一阵暴风，它能够将代码直接嵌入 HTML，使得设计 Web 页面变得更简单，更强大，并且通过内置的组件能够实现强大功能，最明显的就是 ActiveX Data Objects (ADO)，它使得建立一个动态页面如小孩子玩游戏一样简单。

最终出场的是 Active Server Page 1.0，它作为 IIS 的附属产品免费发送。并且不久就在 Windows 平台上广泛使用。ASP 与 ADO 的结合使开发者很容易地在一个数据库中建立和打开一个记录集。这不无疑是它如此快就被大众接受的因素，因为你现在能使用这些脚本建立和打开一个记录集，处理和输出任何数据，以任何顺序，几乎只要你能想到的，它就能完成。

1998 年，微软公司又发布了 ASP 2.0。ASP 1.0 和 ASP 2.0 主要区别是外部的组件需要实例化。有了 ASP 2.0 和 IIS 4.0，我们就有可能建立 ASP 应用了，而且每个组件就有了自己单独的内存空间。内置的 Microsoft Transaction Server(MTS)也使用制做组件而变得简单。

微软公司接着开发了 Windows 2000 操作系统，Windows 版本带上了 IIS 5.0 以及 ASP 3.0。此次并不是简单对 ASP 进行补充，核心的不同实际上是把很多的事情交给了 COM 来做。在 Windows 2000 中，微软结合了 MTS 与 COM 核心环境做出了 COM+，这就让主机有了一种新的方法来使用组件，同样给主机带来了更多的稳定性，成了一个可以升级的效率高的工作平台。IIS 5.0 在表面上似乎没有改什么，但是在接口上动的手术比较大。在内部，它使用 COM+ 组件服务来对组件提供一个更好的执行的环境。

1.3 ASP.NET 特点

ASP.NET 不是 ASP 的简单升级，而是 Microsoft 推出的新一代 Active Server Pages。ASP.net 是微软发展的新的体系结构 .NET 的一部分，其中全新的技术架构会让每个人的编程生活变得更为简单。

首先，需要特别指出的是，ASP.NET 不仅仅只是一个有新界面并且修复了一些缺憾的 ASP3.0 升级版本（就像 ASP 3.0 于 2.0 版做比较一样）。更为重要的是 ASP.NET 是在抓住 ASP 的最大优点并全力使其扩大化的基础上开发出来的，并且同时也修复了许多 ASP 运行时发生的错误。同时，ASP.NET 提供稳定的性能，优秀的升级性，更快速更简便的开发，更简便的管理，全新的语言以及网络服务。

新的 ASP.NET 运行环境 | (NGWS runtime)

新的 ASP.NET 运行环境不只是 ASP 的一个简单变化。在此引入受控代码 (managed code) 这样一个全新概念，它横贯整个视窗开发平台。受控代码运行在 NGWS Runtime 下面。NGWS Runtime 是一个时间运行环境，它管理代码的执行，使程序设计更为简便。

ASP.NET 的新性能

一个程序，速度是一件非常令人渴望的东西。一旦代码开始工作，接下来就得尽可能的让它运行得快些，再快些，在 ASP 中只有尽可能拧干代码，以至于不得不将他们移植到一个仅有很少一点性能的部件中。而现在，微软推出的 ASP.NET 会妥善的解决这一问题。

Web Controls

Web Controls 使创建 forms 和 HTML Controls 的工作将会变得简单易行。例如，在 ASP 中典型的选择框 / select box 里，你须创建一个循环以便让控制系统装入数据。但在 ASP.NET 里，你将会拥有一个 "data-bound"，这意味着它会与数据源连接，并会自动装入数据。

语言支持

ASP.NET 支持多种语言，它的缺省语言将是：Visual Basic.NET 而不是 VBScript，这意味着我们可以摆脱 VBScript 的语言限制，代码将是编译后运行的（而不是原来的解释执行）。

更好的代码控制

运用 ASP 技术的时候，比较麻烦的一件事情就是 COM 对象需要再在服务器上注册，在 ASP.NET 中，这个问题得到了彻底的解决。

更好的升级能力

系统建成后本身有着一定的特性，它还可以改进多处理器和运行环境中的性能。例如，session state 能够通过单独的处理器来维持。在一个单独的机器上，甚至在数据库中允许交叉的服务器会话。

1.4 ASP.NET 与 ASP 的比较

Web 源于静态文本，现在很多站点还是采用这种方式。静态文本的缺点就是维护难(文件数目多)、查询难、修改难。所以现在很多站点都采用 ASP 的动态页面。ASP(Active Server Page)是一种类似于 Visual Basic.NET 的面向对象的程序语言。ASP.NET 也有创作动态页面的能力，而对于 ASP 来说，ASP.NET 有下面的一些突破：

(1) 运行机制不同

ASP 属于一种解释型的编程框架，它的核心是 VBScript 和 JavaScript，受这两种脚本语言的限制，决定了 ASP 先天不足，它无法进行象传统编程语言那样的底层操作，所以如果你需要进行一些诸如 socket、文件等的操作时不得不借助于用其他传统编程语言如 C++、Visual Basic.NET 等编写的组件，并且由于它是解释执行的，所以在运行效率上大打折扣。而 ASP.NET 是一种编译型的编程框架，它的核心是 NGWS runtime，除了和 ASP 一样可以采用 VBScript 和 JavaScript 作为编程语言外，还可以用 Visual Basic.NET 和 C Sharp 来编写，这就决定了它功能的强大，可以进行很多低层操作而不必借助于其他编程语言。

(2) 速度

ASP.NET 是编译后执行的，也就是说当 aspx 文件(ASP.NET 的 Web FORM 文件)第一次被请求时被编译，以后的请求就不需要重新编译了。而 ASP 是解释性脚本语言，每次都需要重新编译，由于这种原因，其速度就无法和 ASP.NET 来比了。不过 ASP.NET 的编译速度也够慢的，在本地机上调试，第一次执行的速度是很慢的，不过以后就很快了。

(3) 功能

ASP.NET 的功能是无比强大的，几乎能做我们在网络上能想到的事情。举个简单的例子，如文件的上传，在 ASP 中，这个问题只能通过组件才行，但是在 ASP.NET 中只需要简单的代码就可以了。

ASP.NET 能做的事远不止如此。上面只是举个小例子，它的更多功能后面将会介绍。ASP.NET 还有一大优点就是结构化编程，程序语言可以自己任意加，目前支持 C#，Visual Basic.NET，JavaScript，对于 C#，大家可能比较陌生，不过还是有很多地方是我们容易上手的。

写 ASP.NET 程序中很多的思想全都来自于 Visual Basic，VC++，这也是 ASP.NET 的一大优点所在。所以我们写程序的思维需要改变，要让你自己感觉这是在写软件，不是在写传统的 ASP 程序。

另外一个问题就是，装有 NGWS 的 Windows 2000 是否还支持 ASP？担心有了 aspx，就没有了 ASP，以前的程序就不能够使用了。这个问题大家根本不用担心，NGWS 设计时微软公司就考虑过，NGWS 同样可以解析 ASP，文件的扩展名是 ASP，那就使用 ASP 的方法解析，如果是 aspx 就使用 ASP.NET 来解析，所以以前的 ASP 是可以继续使用的。

1.5 ASP.NET 与 Java 的比较

Java 语言发展到现在,已经成了当前 Web 编程的主要语言,获得了广泛的业界支持。Java 的最大的特性就是与平台的无关,良好的 OO 属性,是互连网编程的强大的粘合剂。下面是 ASP.NET 与 Java 的一些区别。

(1) 业界支持

如果单纯从业界的支持者来讲,Java 获得了更多的业界的支持。IBM 的“e-business application framework”架构在 Java XML ,CORBA ,IBM 有 VisuageAge For Java ,WebSphere 作为 Web 开发的工具。IBM 认为选择 Application Framework 有几个原则:不依赖于特定的平台,建立在工业标准之上,大部分计算放在 server 端 ,scalable 等 ,同时后面还有 Oracle、Sybase,当然还有 Java 的祖先 Sun 公司。

但是目前微软推出的 .NET 除了平台无关性不能实现之外,功能上完全达到了这个要求,如果真的 .NET 以后成为业界的标准,则微软又一次引领 IT 潮流。

(2) 开发模型

在 .NET 技术还没有出来的时候,国内的很多软件厂商在向 Web 编程迁移的时候,大多选择了 Java。核心软件和南北财务系统,用 Applet 开发客户端界面,使用起来还是跟传统的软件一样。

由于目前采用 HTML 开发界面复杂的应用尚不是很合适,这是选择 java applet 的理由。而 ASP.NET 出来之后,开发复杂的基于 server 的应用要方便的多。

目前采用 applet 的应用,把很多编程逻辑方在客户端,这种方式尚不是未来的编程集中在 server 端的方式。也许,Java 也会推出新的 package 支持这种应用。

在 server 端,Java 采用 Servlet、Java Bean,ASP 采用 Script、COM 组件,目前的编程模式相类似。

ASP.NET 中提到的 Web service 集成的方式,目前微软力推 SOAP、UDDI。但是,这些是建议成为 w3c 标准,不依赖于特定的平台。

从以上关于业界支持、开发模型观察,Java 是一个成熟的产品。而 ASP.NET 的到来,将会是 Web 开发方式的一个更大的发展。

第 2 章 环境与 Visual Studio.NET

2.1 ASP.NET 的运行环境

只有在你的机器安装了 Windows 2000 Server，并且升级到 SP1 上；安装了 IE5.5；安装 Framework SDK，微软的 Visual Studio.NET 才能在上面运行。

2.2 软件的安装

硬件要求

- /// CPU: Intel Pentium II-class 300 MHz (最好 Intel Pentium III-class 600 MHz)
- /// 内存: 96 MB (最好 128 MB)
- /// 磁盘空间: 250 MB(完全安装) 155 MB(快速安装)
- /// 显示: 800x600, 256 colors
- /// CD-ROM: required

软件要求

- /// Microsoft Windows 2000 + SP1
- /// Microsoft Internet Explorer 5.5
- /// IIS5.0
- /// 其它: MDAC 2.6 Beta 2

ASP.NET 的安装过程很简单，只需按照简单提示安装即可。如果你的机器安装了 Office 2000，建议安装 ASP.NET 之前先备份\Microsoft Office\Office\mso9.dll 这个文件，因为安装完 ASP.NET 后，Office 会提示你注册，否则的话 Office2000 就会出现限制使用 50 次。此时将备份的 mso9.dll 文件覆盖掉原来的文件即可。

ASP.NET(NGWS SDK)的下载地址：

<http://download.microsoft.com/download/platformsdk/Trial/1812.10full/NT5/EN-US/Setup.exe>

安装微软的 VisualStudio.NET Beta1.0 和安装 Framework SDK 有很多地方是相似的，所以在此简单地提示一下。

安装 beta1.0 版本的记得必须先安装以下内容：

- /// windows2000 sp1
- /// 安装 IE5.5
- /// 必须要装有 iis,而且 iis 要带 front page 扩展
- /// front page 服务扩展的补丁 QFE

相同地，安装完 VisualStudio.NET 后同样会出现 Office2000 的 50 次限制，所以可以用同样的方法，先备份 mso9.dll 文件，然后安装完后覆盖掉原来的文件。

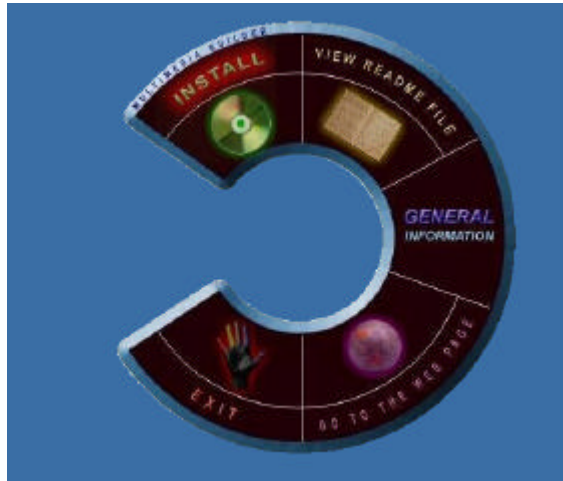


图 2-1

完成图 2-1 的环境要求后，开始安装 Visual Studio.NET Beta1.0，在开始菜单中的表示如图 2-2 所示。

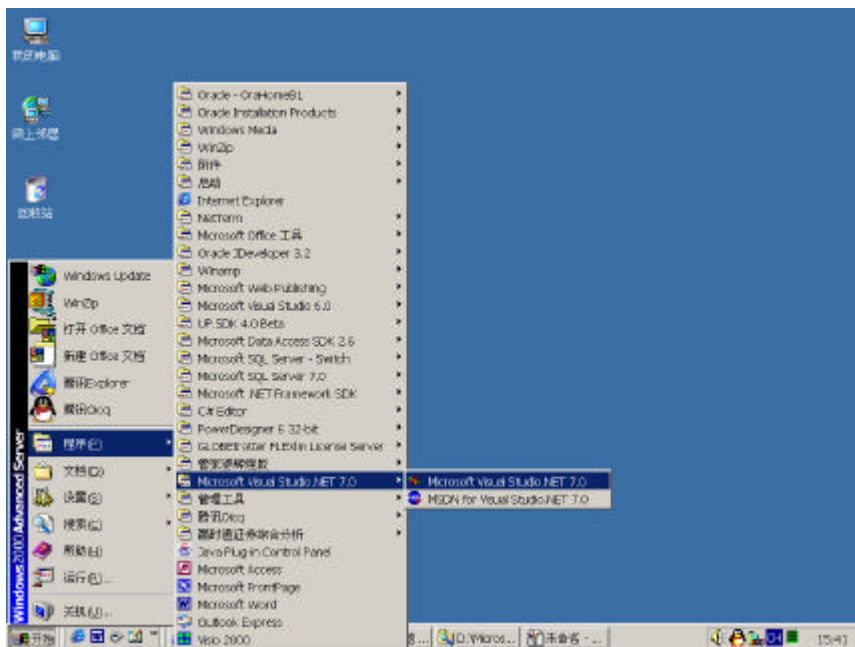


图 2-2

打开后进入的界面，这样就安装成功了。

2.3 Visual Studio.NET 的使用

下面介绍微软的编程环境 Visual Studio.NET，我们创建一个项目来说明 Visual Studio.NET 的应用。

按照上面步骤，打开 Visual Studio.NET 运行环境，打开“File”菜单项，把鼠标放到“New”选项上面，如图 2-3 所示。

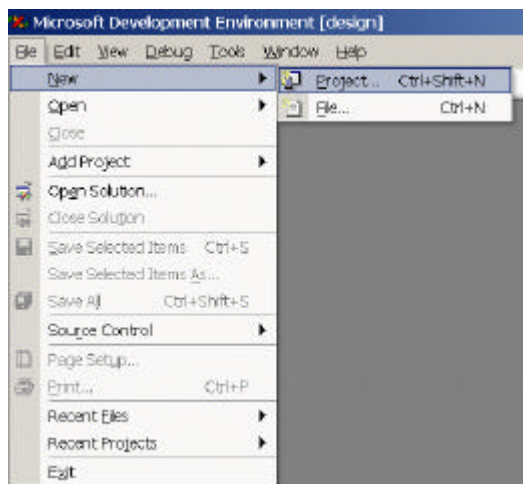


图 2-3

点击“Project...”选项，也可以按照它的提示，按快捷键 Ctrl+Shift+N 来打开它，打开后如图 2-4 所示。

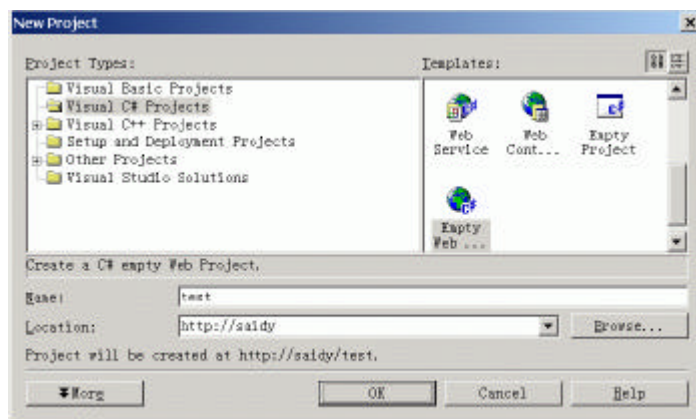


图 2-4

建立一个空的 Web 站点，放在 saidy 这个 Web Server 上的根目录下面，点击“Ok”按钮，结果如图 2-5 所示。

在页面的中间会出现一个对话框，提示正在创建一个 Web 项目“test”，创建完毕后自动消失，出现如图 2-6 所示。

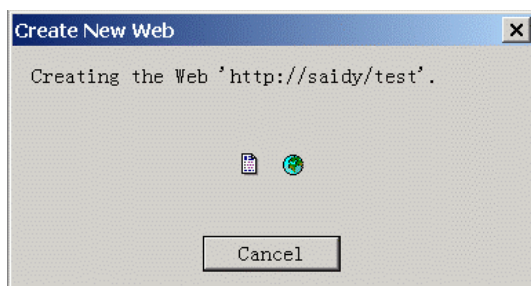


图 2-5

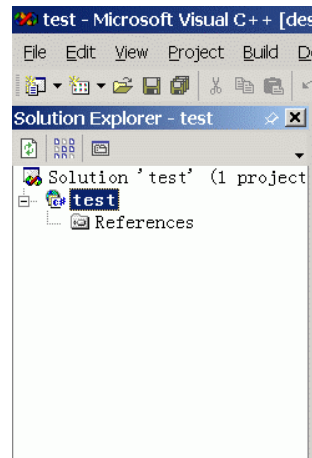


图 2-6

项目创建完毕，出现创建的项目“test”，其中 References 包含的是工作过程中用到的名字空间。

添加一个.cs 文件：

在“test”项目下加上一个.cs 文件,在“test”上点击鼠标的右键，如图 2-7 所示。

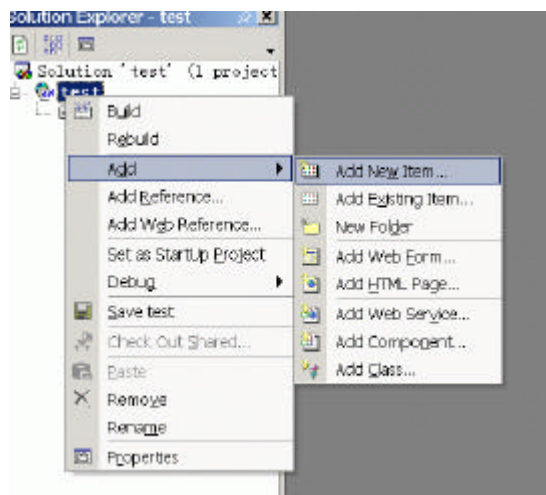


图 2-7

弹出相关的选项，把鼠标移到“Add”选项上面，弹出选项，点击“Add New Item...”项，点击它，如下对话框图 2-8 所示。

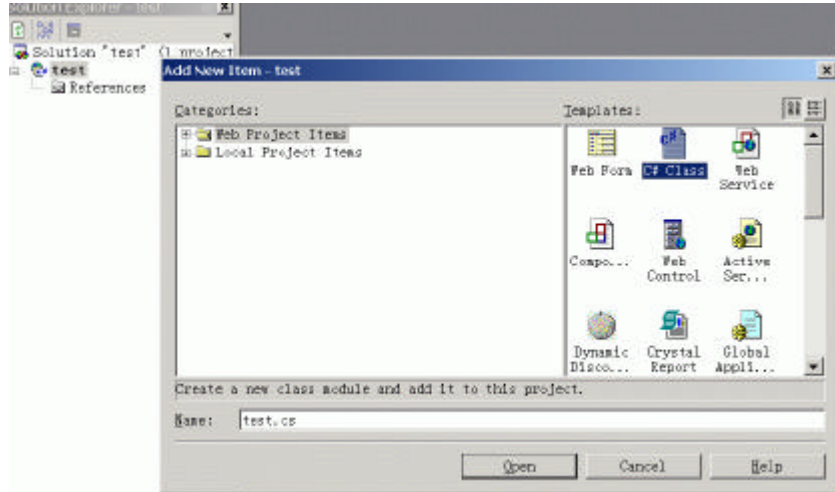


图 2-8

在“Categories”对话框里面有两个选项，选择第一个选项“Web Project Items”，在右边的 Templates 里面点击“C# Class”，定义文件的名字，点击“Open”按钮，见图 2-9 所示。

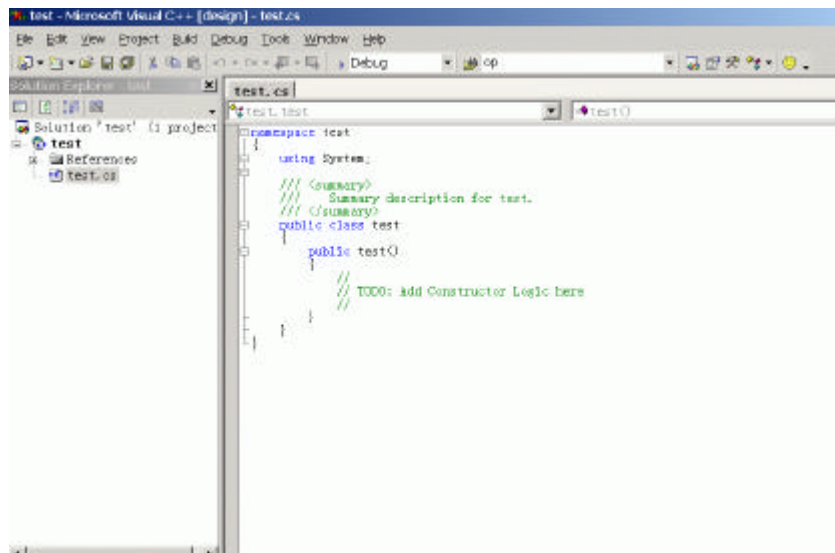


图 2-9

一个.cs 文件创建完毕。

添加一个 Web Form 文件：

右键单击 test 项目，选择“Add”选项并单击，在选择“Add New Item...”，选择“Web Form”，添加一个 aspx 文件，如图 2-10。

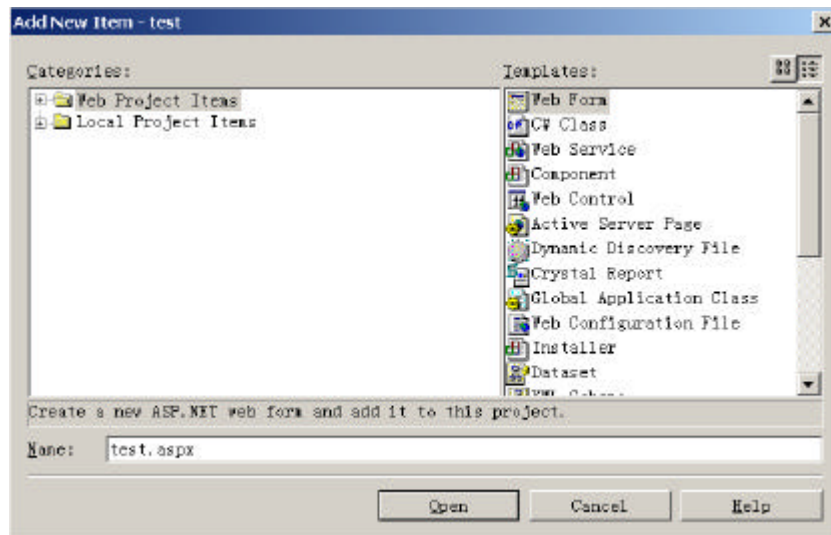


图 2-10

创建一个 Web Form 文件之后，系统会自动生成一个同样名字的.cs 文件，以达到代码和模板的分离，双击创建的文件，在“Design”条件下双击页面，看到同名字的文件代码，见图 2-11。

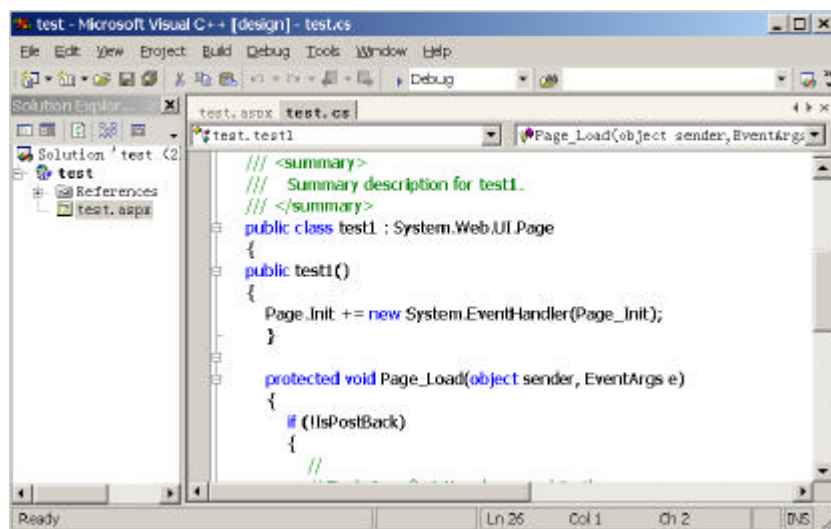


图 2-11

右键单击项目“test”，弹出下面的对话框，可以设置相关属性来创建.dll 文件或者.exe 文件，见图 2-12。

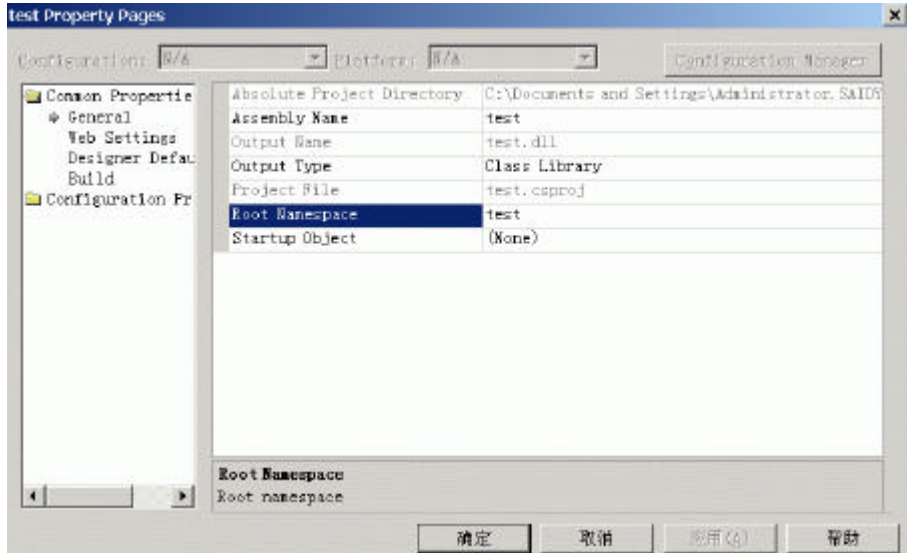


图 2-12

在 Visual Studio.NET 下也可以创建于数据库的连接，点击“Tools”菜单里面的“Connection To Database...”选项，弹出如图 2-13 的对话框。

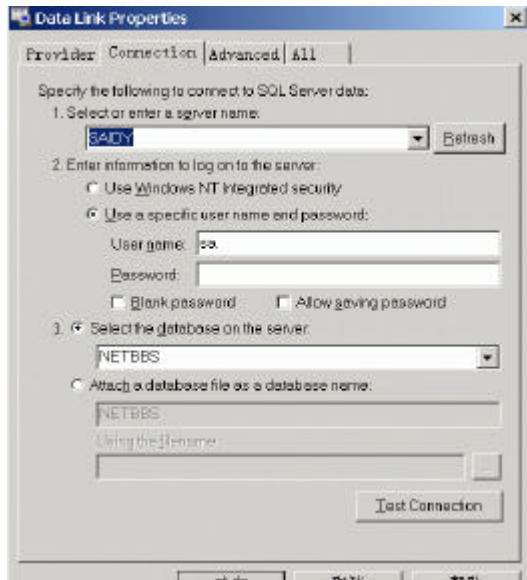


图 2-13

填写相关的内容后就可以创建于数据库的连接，在编程环境下面直接浏览数据库的内容，见图 2-14。

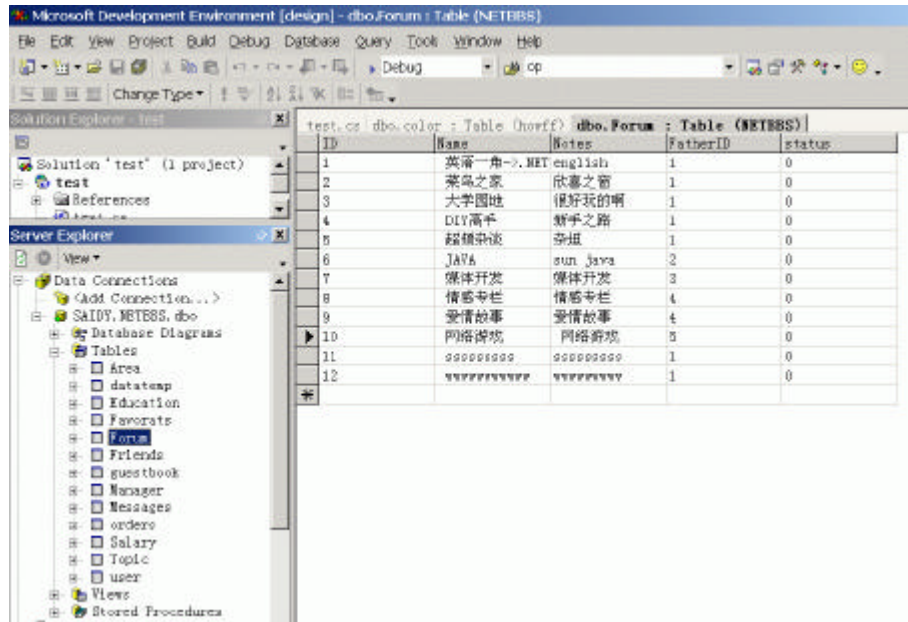


图 2-14

上面的介绍对 VS.NET 来讲只是一个简单的介绍,如果要讲详细,可以写一本书。Visual Studio.NET 集成了 VB.NET、VC.NET、C#等开发环境,是微软.NET 计划的重要组成部分。不过有应用过 Jdeveloper、Jbuilder、VC++、VB 等的用户不会对这样的环境感到陌生的。

第 3 章 ASP.NET 的语法基础

ASP.NET 程序可以用 C# (C Sharp)、VB.NET、Jscript 来实现，在本书中，绝大多数的程序例子是使用 C# (C Sharp) 语言来实现的，所以在此主要介绍 C# 语法。

3.1 数据类型和表达式

3.1.1 声明变量

在 C# 中，对变量的声明如下：

```
int x;
String s;
String s1, s2;
Object o;
Object obj = new Object();
public String name;
```

3.1.2 语句

3.1.2.1 表达式

```
Response.Write("foo");
```

3.1.3 注释

在 C# 中的注释与在 C++ 一样，可以如下注释：

```
// 我的注释
/*
```

```
/*
我
的
注
释
```

尤其有意思的是，微软在 C# 中对注释页下了一点功夫，我们在一个 .cs 文件中用

```
///<summary>
///你的注释，，
///</summary>
```

这些注释最后会形成一个以 xml 文件形式的文档结构。

3.1.4 获得数值

获得一个数值的方法在 Web 编程中经常用到，比如通过一个连接来传递参数，或者通过表单的提交来获得表单参数，如下所示。

```
String s = Request.QueryString["Name"];
String value = Request.Cookies["key"];
```

3.1.5 声明属性

属性的声明，可以通过 get 和 set 方法来进行，如下所示。

```
public String name {
    get {
        ...
        return ...;
    }
    set {
        ... = value;
    }
}
```

3.1.6 数组 Arrays

一个数组包含通过索引来访问的变量，所有的变量都包含在一个数组中，它们的类型必须一样。

一个一维数组：

```
String[] a = new String[3];
a[0] = "1";
a[1] = "2";
a[2] = "3";
```

一个二维数组：

```
String[][] a = new String[3][3];
a[0][0] = "1";
a[1][0] = "2";
a[2][0] = "3";
```

3.1.7 初始化

用下面的方法可以实现初始化：

```
String s = "Hello World";
int i = 1;
double[] a = { 3.00, 4.00, 5.00 };
```

3.1.8 处理

3.1.8.1 字符串相加

字符串在每个编程语言中都要涉及到的，下面简单介绍 C# 的字符串的处理问题，接下来是一个字符串相加的例子，其中包含字符串的定义：

```
String s1;  
String s2 = "hello";  
s2 += " world";  
s1 = s2 + " !!!";
```

3.1.8.2 字符转换

字符的转化在很多场合都要应用到的，下面的例子把一个整数字符转化为一个字符串和一个 Double 类型：

```
int i = 3;  
String s = i.ToString();  
double d = Double.Parse(s);
```

3.2 控制语句

3.2.1 选择语句

3.2.1.1 if 表达式

```
if (Request.QueryString != null) {  
.....  
}
```

3.2.1.2 Case 表达式

```
switch (FirstName) {  
    case "John" :  
        ...  
        break;  
    case "Paul" :  
        ...  
        break;  
    case "Ringo" :  
        ...  
        break;  
    default:  
        ...  
        break;
```

```
}
```

3.2.2 循环语句

3.2.2.1 for 循环

一个应用 for 语句的例子如下：

```
for (int i=0; i<3; i++){  
    a(i) = "test";  
    //  
}
```

标准的 for 语句如下：

```
for(initialize;condition;iterator){  
    //语句  
}
```

其中 initialize;condition;iterator 都是可选的。

3.2.2.2 While 循环

一个 while 语句的例子如下：

```
int i = 0;  
while (i<3) {  
    Console.WriteLine(i.ToString());  
    i += 1;  
}
```

while 的标准的语句如下：

```
while(condition){  
    //语句  
}
```

其中的条件为布尔型表达式，可以用 break 和 continue 来控制 while 语句的执行。

另外循环语句还有 do 和 foreach 语句，其中 foreach 是一个很有用的语句，用来获得枚举数据；do 语句应用于循环当中，跟 while 非常相似，只是 do 语句是在在执行完第一次循环语句之后才开始验证的。

3.3 类

一个类可以包含数据、函数等，数据可以是常数、字符串或者事件；函数包含方法、属性、索引器、操作符、构造器和析构器。大家应该区分 struct 和 class 的区别，正确的是 struct 是一个值类型而 class 是一个引用类型。

3.3.1 事件处理方法

```
void MyButton_Click(Object sender,
                    EventArgs E) {
}
}
```

3.3.2 派生

```
MyObject obj = (MyObject)Session["Some Value"];
IMyObject iObj = obj;
```

3.3.3 类的定义和继承

C#是一种面向对象的编程语言，C#中提供了一系列的面向对象的特性，其中包括应用的继承性，一个类可以由继承其他的类生成。与其他的面向对象语言一样，在C#中可以覆盖基础类中的方法和属性，还可以利用多态性创建功能更强大、可扩充的组件。

```
using System;
namespace MySpace {
    public class Foo : Bar {
        int x;
        public Foo() { x = 4; }
        public void Add(int x) { this.x += x; }
        public int GetNum() { return x; }
    }
}
// csc /out:librarycs.dll /t:library
// library.cs
```

C#不仅仅可以覆盖方法或属性，而且可以实现方法的重载。利用重载，用户可以使用相同的名字定义具有不同数据类型的参数的方法或属性。例如，如果需要一个组件对具有不同数据类型的一组数据进行排序，利用重载就无需三个具有不同名字的方法（每种数据类型对应一个方法），而只需重载同一个方法即可。

3.3.4 包含一个 Main 方法的类

```
using System;
public class ConsoleCS {
    public ConsoleCS() {
        Console.WriteLine("Object Created");
    }
    public static void Main (String[] args) {
        Console.WriteLine("Hello World");
        ConsoleCS ccs = new ConsoleCS();
    }
}
```

```
}  
// csc /out:consolecs.exe /t:exe console.cs
```

3.3.5 标准模块类

```
using System;  
public class Module {  
public static void Main (String[] args) {  
    Console.WriteLine("Hello World");  
}  
}  
// csc /out:consolecs.exe /t:exe console.cs
```

3.3.6 类其中的两个应用

3.3.6.1 继承

继承是两个父子类之间的关系，父类也叫基类、超类或者祖先类，当一个子类继承一个基类，这种关系就存在了。比方说有一个超类 Cat，一个子类 Lion。Lion 与 Cat 就存在这种关系，这种关系允许 Lion 继承 Cat 的所有属性，并且可以扩展和增加 Cat 的属性。Cat 类中有两个方法 eat()和 sleep()，但是 Lion 类可以扩展、增加和改变这些方法，比如增加一个方法 Play()。

C#是一个面向对象的编程语言，所以它有面向对象语言的一切特性，继承就是它的特性之一，下面的例子说明 C#的这一特性。

在组件文件 profile.cs，其中用到了继承。我们创建两个类 Profile、ExtendedProfile，其中 ExtendedProfile 为子类。在 Profile 类中有三个属性：_phoneNumber, _firstName, _lastName，用 get 和 set 方法为它们赋值。最后 ExtendedProfile 继承了这三个属性并且加上了自己的属性和方法。下面是完整的代码：

页面显示文件 (inheritance/TestProfile.aspx):

```
<%@ Import Namespace="inheritance" %>  
<html>  
<style>  
div  
{  
    font: 8pt verdana;  
background-color:cccccc;  
border-color:black;  
border-width:1;  
border-style:solid;  
padding:10,10,10,10;  
}
```

```
</style>
<script language="C#" runat="server">

    public void Page_Load(Object sender, EventArgs E) {

        Profile profile = new Profile();
        Message.InnerHtml += "<u>Profile Class</u><br>";
        Message.InnerHtml += "First: " + profile.getFirstName() + "<br>";
        Message.InnerHtml += "Last: " + profile.getLastName() + "<br>";
        Message.InnerHtml += "Phone: " + profile.getPhoneNumber() + "<br><br>";

        ExtendedProfile extendedProfile = new ExtendedProfile();
        Message.InnerHtml += "<u>ExtendedProfile Class</u><br>";
        Message.InnerHtml += "First: " + profile.getFirstName() + "<br>";
        Message.InnerHtml += "Last: " + profile.getLastName() + "<br>";
        Message.InnerHtml += "Phone: " + extendedProfile.getPhoneNumber() +
"<br>";
        Message.InnerHtml += "Adress1: " + extendedProfile.getAdress1() +
"<br>";
        Message.InnerHtml += "Adress2: " + extendedProfile.getAdress2() +
"<br>";
        Message.InnerHtml += "City: " + extendedProfile.getCity() + "<br>";
        Message.InnerHtml += "State: " + extendedProfile.getState() + "<br>";
        Message.InnerHtml += "Postal: " + extendedProfile.getPostal() + "<br>";
        Message.InnerHtml += "Description: " + extendedProfile.getDescription()
+ "<br>";

    }

</script>
<body style="font: 10pt verdana">
    <b><h3>Simple Inheritance Example</h3></b><br><br>
    Object Output:<br>
    <br>
    <div id="Message" runat="server"/>
</body>
</html>
组件文件 inheritance/profile.cs , 其中用到了继承 ,
```

```
(inheritance/profile.cs)
namespace inheritance {
using System;
using System.Text;
    public class Profile
    {
        private String _firstName;
        private String _lastName;
        private String _phoneNumber;
        public Profile()
        {
            _firstName = "Saidy";
            _lastName = "Chan";
            _phoneNumber = "(010)-000-00000";
        }

        public void setPhoneNumber(String phoneNumber)
        {
            _phoneNumber = phoneNumber;
        }
        public String getPhoneNumber()
        {
            return _phoneNumber;
        }

        public void setFirstName(String firstName)
        {
            _firstName = firstName;
        }
        public String getFirstName()
        {
            return _firstName;
        }

        public void setLastName(String lastName)
        {
            _lastName = lastName;
        }
        public String getLastName()
```

```
    {  
        return _lastName;  
    }  
}  
  
public class ExtendedProfile: Profile  
{  
    private String _address1;  
    private String _address2;  
    private String _city;  
    private String _state;  
    private String _postal;  
    private String _description;  
    public ExtendedProfile()  
    {  
        _address1 = "01,HuangFu Street";  
        _address2 = "XuanWu";  
        _city = "BeiJing";  
        _state = "BeiJing";  
        _postal = "100000";  
        _description = "软件工程师";  
    }  
  
    public void setAddress(String address1, String address2)  
    {  
        _address1 = address1;  
        _address2 = address2;  
    }  
    public String getAddress1()  
    {  
        return _address1;  
    }  
    public String getAddress2()  
    {  
        return _address2;  
    }  
  
    public void setCity(String city)  
    {
```

```
        _city = city;
    }
    public String getCity() {
        return _city;
    }

    public void setState(String state)
    {
        _state = state;
    }
    public String getState() {
        return _state;
    }

    public void setPostal(String postal)
    {
        _postal = postal;
    }
    public String getPostal()
    {
        return _postal;
    }

    public void setDescription(String description)
    {
        _description = description;
    }
    public String getDescription()
    {
        return _description;
    }
}

}
```

(编译文件) inhe.bat :

```
csc /t:library /r:System.Xml.dll /out:..\bin\IProfile.dll Profile.cs
```

运行结果如图 3-1。

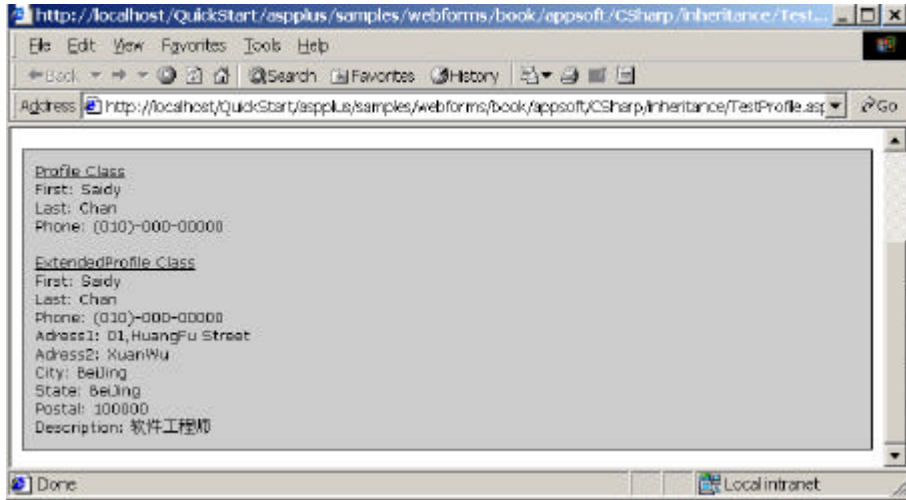


图 3-1

3.3.6.2 多态性

多态也是面向对象的编程语言的特性之一，在 C#中，多态性被描述为可以共享其他的方法而在执行可以进行自己的修改。

在这个例子中创建两个类：父类 Profile、子类 ExtendedProfile，父类 Profile 用来保存用户的信息，子类 ExtendedProfile 继承父类 Profile 的属性并扩展它们、增加属性和方法。还创建一个方法来保存来自外部的信息：

```
<SOURCE>
    interface ISaveData
    {
        void save();
    }
</SOURCE>
```

一个 interface 允许用户创建类和方法，在你的类中可以通过执行它来继承这些方法，如下：

```
<SOURCE>
    public class Profile : ISaveData
</SOURCE>
```

下面就是代码：

页面文件 (multi/TestProfile.aspx)：

```
<%@ Import Namespace="Shai" %>
<html>
```

```
<style>
  div
  {
    font: 8pt verdana;
    background-color:cccccc;
    border-color:black;
    border-width:1;
    border-style:solid;
    padding:10,10,10,10;
  }

</style>
<script language="C#" runat="server">

  public void Page_Load(Object sender, EventArgs E) {

    Profile profile = new Profile();
    Message.InnerHtml += "<u>Profile Class</u><br>";
    Message.InnerHtml += "First: " + profile.getFirstName() + "<br>";
    Message.InnerHtml += "Last: " + profile.getLastName() + "<br>";
    Message.InnerHtml += "Phone: " + profile.getPhoneNumber() + "<br><br>";

    profile.save();

    ExtendedProfile extendedProfile = new ExtendedProfile();
    Message.InnerHtml += "<u>ExtendedProfile Class</u><br>";
    Message.InnerHtml += "First: " + profile.getFirstName() + "<br>";
    Message.InnerHtml += "Last: " + profile.getLastName() + "<br>";
    Message.InnerHtml += "Phone: " + extendedProfile.getPhoneNumber() +
"<br>";
    Message.InnerHtml += "Adress1: " + extendedProfile.getAdress1() +
"<br>";
    Message.InnerHtml += "Adress2: " + extendedProfile.getAdress2() +
"<br>";
    Message.InnerHtml += "City: " + extendedProfile.getCity() + "<br>";
    Message.InnerHtml += "State: " + extendedProfile.getState() + "<br>";
    Message.InnerHtml += "Postal: " + extendedProfile.getPostal() + "<br>";
    Message.InnerHtml += "Description: " + extendedProfile.getDescription()
+ "<br>";

    extendedProfile.save();
  }
}
```



```
</script>
<body style="font: 10pt verdana">
  <b><h3>简单的继承/多态例子</h3></b>
  Object Output:<br>
  <br>
  <div id="Message" runat="server"/>
</body>
</html>
```

组件文件 (multi/profile.cs):

```
namespace Shai {
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Xml;

interface ISaveData
{
    void save();
}

public class Profile : ISaveData
{
    protected String _firstName;
    protected String _lastName;
    protected String _phoneNumber;

    public Profile()
    {
        _firstName = "Saidy";
        _lastName = "Chan";
        _phoneNumber = "(010)-000-00000";
    }

    public virtual void setPhoneNumber(String phoneNumber)
    {
        _phoneNumber = phoneNumber;
    }
    public String getPhoneNumber()
    {
        return _phoneNumber;
    }
}
```

```
    }

    public void setFirstName(String firstName)
    {
        _firstName = firstName;
    }
    public String getFirstName()
    {
        return _firstName;
    }

    public void setLastName(String lastName)
    {
        _lastName = lastName;
    }
    public String getLastName()
    {
        return _lastName;
    }

    public virtual void save() {

        //save data in text format:
        FileStream s = new FileStream("D:\\profile.bin",
FileMode.Create, FileAccess.Write);
        BinaryFormatter b = new BinaryFormatter();
        b.Serialize(s, this);
        s.Close();
    }

}

public class ExtendedProfile: Profile
{
    protected String _address1;
    protected String _address2;
    protected String _city;
    protected String _state;
    protected String _postal;
    protected String _description;
    public ExtendedProfile()
    {
```

```
        _address1 = "清华大学";
        _address2 = "海淀区";
        _city = "北京";
        _state = "北京";
        _postal = "100000";
        _description = "教授";
    }

    public override void setPhoneNumber(String phoneNumber)
    {
        _phoneNumber = phoneNumber;
    }

    public void setAddress(String address1, String address2)
    {
        _address1 = address1;
        _address2 = address2;
    }

    public String getAddress1()
    {
        return _address1;
    }

    public String getAddress2()
    {
        return _address2;
    }

    public void setCity(String city)
    {
        _city = city;
    }

    public String getCity() {
        return _city;
    }

    public void setState(String state)
    {
        _state = state;
    }

    public String getState() {
        return _state;
    }
}
```

```
public void setPostal(String postal)
{
    _postal = postal;
}
public String getPostal()
{
    return _postal;
}

public void setDescription(String description)
{
    _description = description;
}
public String getDescription()
{
    return _description;
}

public override void save()
{
    String _document = "D:\\sai dy.xml" ;
    XmlTextWriter writer = null;

    try
    {

        writer = new XmlTextWriter (_document, null);
        writer.Formatting = Formatting.Indented;
        writer.WriteStartDocument(false);
        writer.WriteDocType("Profile", null, null, null);
        writer.WriteStartElement("Profile");
            writer.WriteElementString("firstName", _firstName);
            writer.WriteElementString("lastName", _lastName);
            writer.WriteElementString("phoneNumber", _phoneNumber);
            writer.WriteElementString("address1", _address1);
            writer.WriteElementString("address2", _address2);
            writer.WriteElementString("city", _city);
            writer.WriteElementString("state", _state);
            writer.WriteElementString("postal", _postal);
        writer.WriteEndElement();
    }
}
```


3.5 中间语言概述

无论是用 VB.NET 或者 C#编写的软件都会被编译成一种“中间语言”(IL),只有在软件运行时,一个运行时编译器(JITter)才将 IL 代码编译成机器语言,这意味着创建非 Windows 平台的.NET 运行库是可能的。

在 IL 一级对代码有影响的 CLR 的变化能使所有的使用 CLR 的开发人员受益。对特定语言的优化主要与如何将这种语言编译为 IL 的质量有关,因此从技术上说,在不同的.NET 语言之间还是有着细微的差别。尽管如此,总体情况还是很好的,比如,VB.NET 与 C#具有相同水平的调试和分析工具,因为它们使用的就是同一个工具。

CLR 提供了空前的跨语言集成能力,其中包括跨语言的代码继承。所有的使用 CLR 的语言都共享一个相同的类型系统,这就使得利用多种编程语言开发软件变得更为简单。

在 CLR 中运行的代码被称作管理代码,它使用的内存是完全由 CLR 控制的。管理代码带来的好处是显而易见的,包括跨语言的集成性、跨语言的异常处理和组件交互的单一模型。Visual Basic 只能使用管理代码,而 C#则还可以不使用管理代码(不使用运行库),使用指针管理等功能,这是 VB.NET 与 C#的一个不同之处,这一点的重要性取决于你需要完成的任务。

由 CLR 带来的结构上的差异性远不止跨语言的继承、共享的特性和管理代码。Visual Studio.NET 的基础架构不是 COM,包括字符串在内的 VB.NET 中的所有元素都是对象。基于这些原因和其他的一些原因,微软改变了基础架构处理对象的方式,每当引用一个对象时,COM 都把对象引用计数器加 1。

第 4 章 Web 页面

4.1 Web Form

表单，英文是 Form，学习过 VB 的朋友一定不会陌生。在 MS.NET 架构里，Form 是一个经常使用的词汇。比如：编写 Windows 应用时要提到 Windows Form，编写 Web 应用时要提到 Web Form。Windows Form 可以看作一个 Windows 窗体，这和 VB 里面一样。而 Web Form 则代表了一个一个的 Web 页面。这里译作“Web 表单”似乎有些不妥。“表单”这个词，在 Web 程序员看来，总是和 HTML 里面的“Form”混淆。这里的“Web 表单”似乎翻译成“Web 页面”更加妥当一些。请读者留意这一点。

大家还记得 VB 里面的 Form 实际上就是一个对象，它可以有自己的属性、方法、事件等。Web 表单，或者说 Web 页面，实际上是一个“对象” (Object)。MS.NET 架构里面最重要的东西就是“对象”：所有的东西都是对象，甚至数据类型都成了对象；每种数据类型都有自己特有的属性和方法。我们在后面的编程中可以体会到。

Web Form 的后缀名是 aspx。当一个浏览器请求一个 aspx 文件时，Web Form 页面被 CLR 编译器编译。当再有用户访问此页面的时候，由于 aspx 页面已经被编译，所以，CLR 会直接执行编译过的代码。这和 ASP 完全不同。ASP 只支持 VBScript 和 JavaScript 这样的解释性的脚本语言。所以 ASP 页面是解释执行的。而 ASP.NET 支持可编译的语言，包括 VB.NET、C#、Jscript.NET 等。所以，ASP.NET 是一次编译多次执行的。

为了简化程序员的工作，ASPX 页面不需要手工编译，而是在页面被调用的时候，由 CLR 自动决定是否编译。一般来说，下面两种情况下，ASPX 会被重新编译：

1. aspx 页面第一次被浏览器请求；
2. aspx 被改写；

由于 aspx 页面被编译，所以 aspx 页面具有组件一样的性能。这就使得 aspx 页面至少比同样功能的 asp 页面快 250%！

下面我们来看一下简单的 Web 页面。

4.2 我的第一个 Page

图 4-1 展示的就是我们要完成的第一个 aspx 页面。

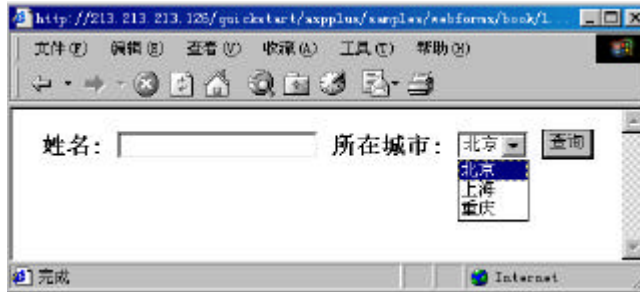


图 4-1

把下面的代码拷贝到 1-1.aspx 文件中，然后从浏览器访问这个文件：

```
<form action="form1.aspx" method="post">
  <h3> 姓名: <input id="name" type="text">

  所在城市: <select id="city" size=1>
    <option>北京</option>
    <option>上海</option>
    <option>重庆</option>
  </select>

  <input type="submit" value="查询">

</form>
```

你可能觉得这个页面太简单了，用 HTML 就可以完成。是的！微软建议你所有的文件哪怕是纯 HTML 文件都保存为 ASPX 文件后缀，这样可以加快页面的访问效率！不仅仅是 aspx，在 IIS5.0 以后的 ASP3.0 就已经支持这个特性了。

由于我们没有对表单提交做任何响应，所以，当你按下“查询”按钮，页面的内容没有什么改变。

下面我们将逐步用 ASP.NET 的思考方法，来完善这个例子。

4.3 使用 Server Control

上面说过，在 ASP.NET 里面，一切都是对象。我们也谈到：Web 页面本身就是一个对象。或者说，Web 页面就是一个对象的容器。那么，这个容器可以装些什么东西呢？下面学习服务器端控件，英文是 Server Control。这是 Web 页面能够容纳的对象之一。

什么是 Control？熟悉 VB 的读者肯定再清楚不过了。简单地说，Control 就是一个可重用的组件或者对象，这个组件不但有自己的外观，还有自己的数据和方法，大部分组件还可以响应事件。通过微软的集成开发环境(Visual Studio.NET 7.0)，你可以简单地把一个 Control 拖放到一个 Form 中。

那为什么叫“Server Control”？这是因为这些 Control 是在服务器端存在的。服务器端控件也有自己的外观，在客户端浏览器中，Server Control 的外观由 HTML 代码来表现。Server Control 会在初始化时，根据客户的浏览器版本，自动生成适合浏览器的 HTML 代码。以前我们做网页或者开发 ASP 程序时候，必须考虑到浏览器的不同版本对 HTML 的支持有所不同，比如 Netscape 和 IE 对 DHTML 的支持就有所不同。当时，解决浏览器版本兼容问题的最有效方法，就是不同版本的浏览器中作测试。现在，由于 Server Control 自动适应不同的浏览器版本，也就是自动兼容不同版本的浏览器，程序员的工作量减轻了许多。

下面，我们来看看如何在 Web Form 中嵌入 Server Control。该例子是从上一节继承来的：

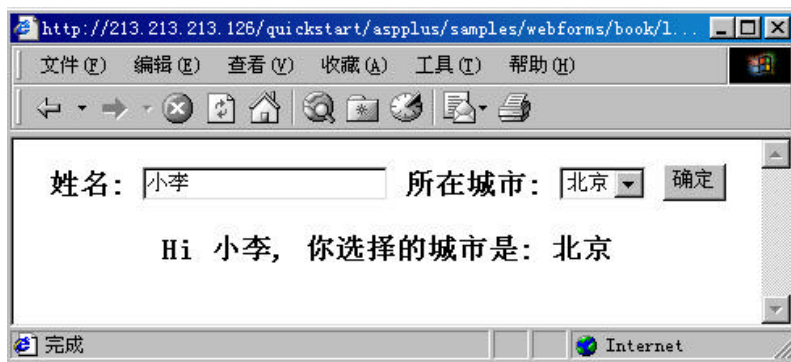


图 4-2

下面是实现图 4-2 结果的代码 (1-2.aspx)：

```
<html>
  <script language="VB" runat=server>
    Sub SubmitBtn_Click(Sender As Object, E As EventArgs)
      Message.Text = "Hi " & Name.Text & ", 你选择的城市是: " &
city.SelectedItem.Text
    End Sub
  </script>
  <body>
    <center>
      <form action="form2.aspx" method="post" runat="server">
        <h3>姓名: <asp:textbox id="Name" runat="server"/>

        所在城市: <asp:dropdownlist id="city" runat=server>
          <asp:listitem>北京</asp:listitem>
          <asp:listitem>上海</asp:listitem>
          <asp:listitem>重庆</asp:listitem>
        </asp:dropdownlist>
```

```
        <asp:button type=submit text="确定" OnClick="SubmitBtn_Click"
runat="server"/>
        <p>
            <asp:label id="Message" runat="server"/>
        </form>
    </center>
</body>
</html>
```

请注意，上面的代码中使用了三种 Server Control，分别是：

1. asp:textbox
2. asp:dropdownlist
3. asp:label

我们注意到三个控件都有相同的 RunAt 属性：**RunAt="Server"**。所有的服务器端控件都有这样的属性。这个属性标志了一个控件是在 Server 端进行处理的。

看下面的代码：

```
<script language="VB" runat=server>
    Sub SubmitBtn_Click(Sender As Object, E As EventArgs)
        Message.Text = "Hi " & Name.Text & ", 你选择的城市是: " &
city.SelectedItem.Text
    End Sub
</script>
```

用过 VB 的朋友是不是觉得很熟悉？没错，这是用 VB 写的一个事件处理函数，void SubmitBtn_Click(Object sender, EventArgs e)，你可能一看就明白了，void 代表该函数没有返回值，该函数带有两个参数，可是这里的 Sender 的意义是什么意思呢？他的用处又到底是什么呢？其实很简单，这个 Sender 就是这个事件的触发者。这里，Sender 就是被 Click 的 button。其中代码只有一行，你可能注意到这行代码中的 Message、Name、city 并没有定义，那么它们从哪里来的呢？

看下面的代码：

```
<form action="form2.aspx" method="post" runat="server">
    <h3> Name: <asp:textbox id="Name" runat="server"/>
        Category: <asp:dropdownlist id="city" runat=server>
            <asp:listitem>北京</asp:listitem>
            <asp:listitem>上海</asp:listitem>
            <asp:listitem>重庆</asp:listitem>
        </asp:dropdownlist>
    <asp:button type=submit text="确定" OnClick="SubmitBtn_Click"
runat="server"/>
    <p>
        <asp:label id="Message" runat="server"/>
```

```
</form>
```

我们发现每个服务端的控件都带有一个 ID 号。而我们在 VB.NET 代码中所引用的就是这些 ID。可以认为 ID 就是控件的名称。在 ASP 中也使用过 ID 吧。那时候, ID 属性和 Name 属性并没有什么不同:

```
<input id=email name=email >
```

在客户端, 通过 VBScript 代码或者 Jscript 代码, 可以这样访问 Form 表单的 Input 域:

```
<SCRIPT LANGUAGE=javascript>
```

```
<!--
```

```
document.all("email")="darkman@yesky.com";
```

```
//-->
```

```
</SCRIPT>
```

从上面的代码可以看出, 在 DHTML 中, 我们也是通过 ID 来访问 Form 表单的输入域的。在 ASPX 中, 情况有些类似之处。差别在于: 一个在客户端, 一个在服务器端。

如果你和第一节例子代码对比一下, 你会发现: 这个表单的写法和 html 表单完全不同了吧? 首先, 所有的表单项包括表单本身后面都加上了 `runat=server`, 这句话的意思就是说这个是服务器端控制项, 另外象传统表单的什么 `<input type=text>` 等的写法都变了, 你仔细观察一下可以看出, 原来的文本框变为 `<asp:textbox>`, 选择框变为 `<asp:dropdownlist>`, 选择框选项变为 `<asp:listitem>`, 而 submit 按钮变为 `<asp:button>`, 这个按钮对应的控制函数就是刚才我提到的那个 `SubmitBtn_Click` 函数, 它是运行在服务器端的。另外还有一个服务器端控制 `<asp:label id="Message" runat="server" />`, 这个 `asp:label` 是传统表单所没有的, 它是一个服务器端文本控制, 那么就存在一个问题, 如果传统的 HTML 里没有这个元素, 那么 ASP+ 是怎么接收的呢? 你运行一下这个程序, 然后看一下 HTML 源码, 你会发现这么一行:

```
<input type="hidden" name="__VIEWSTATE" value="..." />
```

对, ASP+ 就是通过这个隐藏表单的形式传递过去的。

所以, 一个客户端控件, 加上 `runat=Server` 就变成服务器端控件, 服务器端控件能在服务器端脚本中被自由运用。在以后的章节中, 我们还要对常用的服务器端控件进行详细介绍。

4.4 Web 页面处理过程

接下来深入到 ASP.NET 内部, 看看页面是怎样被处理的。

和所有的服务器端进程一样, 当 aspx 页面被客户端请求时, 页面的服务器端代码被执行, 执行结果被送回到浏览器端。这一点和 asp 并没有太大的不同。

但是, ASP.NET 的架构为我们做了许多。比如, 它自动处理浏览器的表单提交, 把各个表单域的输入值变成对象的属性, 使得我们可以像访问对象属性那样来访问客户的输入。它还把客户的点击映射到不同的服务器端事件。

了解 Web 页面的处理过程很重要。这样可以仔细地优化你的代码，提高代码的效率。

4.4.1 页面的一次往返处理

用户对 Server Control 的一次操作，就可能引起页面的一次往返处理：页面被提交到服务器端，执行响应的事件处理代码，重建页面，然后返回到客户端。

正因为每个 Control 都可能引发一次页面的服务器端事件，所以，ASP.NET 尽量减少了控件的事件类型。很多组件只有 OnClick 事件。我们注意到 ASP.NET 不支持服务器端的 OnMouseOver 事件。我们知道，OnMouseOver 事件是非常频繁发生的。所以，支持服务器端的 OnMouseOver 事件是非常不现实的。

4.4.2 页面重建

每一次页面被请求，或者页面事件被提交到服务器，ASP.NET 运行环境将执行必要的代码，重建整个页面，把结果页面送到浏览器，然后抛弃页面的变量、控件的状态和属性等页面信息。当下一次页面被处理时，ASP.NET 运行环境是不知道它的上一次执行情况的。在这个意义上，ASPX 页面是没有状态的。这也是 HTTP 协议的特点。

说明：为了加速页面的访问，在 ASP.NET 页面里面可以使用缓存机制，也就是保存页面的执行结果，下一次页面被请求时，直接送回上一次的执行结果。

在 ASP 中，当页面被提交到服务器端时，只有那些用户输入的值被传递到服务器。其他的比如组件的属性、变量的值，是不会传递的。所以服务器无法了解组件的进一步的信息。

在 ASP.NET 中，页面对象的属性、页面控件的属性被称为“view state”(页面状态)，页面状态在 ASP.NET 中被受到特别关照。请看服务器端的代码：

```
<HTML>
<BODY>
<SCRIPT language="VB" runat="server">
  Sub ShowValues(Sender As Object, Args As EventArgs)
    divResult.innerText = "You selected '" _& selOpSys.value & "'
for machine '" _& txtName.value & "'."
  End Sub
</SCRIPT>
<DIV id="divResult" runat="server">
</DIV>
<FORM runat="server">
  机器名：
  <INPUT type="text" id="txtName" runat="server">
  <P />
  操作系统：
```

```

<select id="selOpSys" size="1" runat="server">
  <OPTION>Windows 95</OPTION>
  <OPTION>Windows 98</OPTION>
  <OPTION>Windows NT4</OPTION>
  <OPTION>Windows 2000</OPTION>
</SELECT>
<P />
<INPUT type="submit" value="Submit" runat="server"
onserverclick="ShowValues">
</FORM>
</BODY>
</HTML>

```

运行后将自动被解释成客户端代码，如下：

```

<HTML>
<BODY>
You selected 'Windows 98' for machine 'iceberg'.
<FORM name="ctrl0" method="post" action="pageone.aspx" id="ctrl0">
<INPUT type="hidden" name="__VIEWSTATE" value="a0z1741688109__x">
机器名:
<INPUT type="text" id="txtName" name="txtName" value="tizzy">
<P />

```

操作系统:

```

<SELECT id="selOpSys" size="1" name="selOpSys">
  <OPTION value="Windows 95">Windows 95</OPTION>
  <OPTION selected value="Windows 98">Windows 98</OPTION>
  <OPTION value="Windows NT4">Windows NT4</OPTION>
  <OPTION value="Windows 2000">Windows 2000</OPTION>
</SELECT>
<P />
<INPUT type="submit" value="Submit">
</FORM>
</BODY>
</HTML>

```

对于上面的代码，服务器端控件能在服务器端脚本中被自由运用。如果我们用传统的 ASP 代码实现上述的功能的话：

```

If Len(Request.Form("selOpSys")) > 0 Then
  StrOpSys = Request.Form("selOpSys")
  StrName = Request.Form("txtName")
  Response.Write("You selected '" & strOpSys & "' for machine
'" & strName & "'.")
End If

```

如果我们用 asp+的话，程序代码如下：

```
If Len(selOpSys.value) > 0 Then
    Response.Write("You selected '" & selOpSys.value _& "' for
machine '" & txtName.value & "'.")
End If
```

通过上面不难看出：

asp+页面具有组件方式的功能。

注意：ASP.NET 通过把页面的状态封装到一个隐藏的输入域，从而在不同的页面之间传递页面的状态。

另外，ASP.NET 也支持应用程序一级的状态管理。这个特性在 ASP 中就已经实现。

4.4.3 页面处理内部过程

我们来看看页面处理的内部过程。下面的过程是依次进行的：

4.4.3.1 Page_load

首先，页面的状态被恢复，然后触发 Page_OnLoad 事件。在这个过程中，你可以读取或者重置页面的属性和控件的属性，根据 IsPostBack 属性判定页面是否为第一次被请求，执行数据绑定，等等。

现在通过一个具体的例子，来详细讲述 Page_load 事件：

我们所做的这个例子有关用户登录的。

我们先来看 page.aspx 的代码：

```
<%@ Register TagPrefix="Acme" TagName="Login" Src="page.ascx" %>
<html>
<title>登录演示</title>
<script language="VB" runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)
        If (Page.IsPostBack)
            MyLabel.Text &= "用户名：" & MyLogin.UserId & "<br>"
            MyLabel.Text &= "密码：" & MyLogin.Password & "<br>"
        End If
    End Sub
</script>
<body style="font: 10pt verdana">
<center> <h3>登录</h3></center>
<form runat="server">
    <Acme:Login id="MyLogin" UserId="" Password="" BackColor="beige"
runat="server"/>
```

```

    </form>
    <asp:Label id="MyLabel" runat="server" />
</body>
</html>

```

在这个文件中，使用了 Page_OnLoad 事件的 IsPostBack 属性，用来显示用户登录时的用户名和密码。

在来看一下 page.ascx 文件：

```

<script language="VB" runat="server">
    Public BackColor As String = "white"
    Public Property UserId As String
        Get
            Return UserName.Text
        End Get
        Set
            UserName.Text = Value
        End Set
    End Property
    Public Property Password As String
        Get
            Return Pass.Text
        End Get
        Set
            Pass.Text = Value
        End Set
    End Property
</script>
<center>
    <table style="background-color:<%=BackColor%>;font: 10pt
verdana;border-width:1;border-style:solid;border-color:black;"
cellspacing=15>
        <tr>
            <td><b>用户名: </b></td>
            <td><ASP:TextBox id="UserName" runat="server" /></td>
        </tr>
        <tr>
            <td><b>密码: </b></td>
            <td><ASP:TextBox id="Pass" TextMode="Password" runat="server" /></td>
        </tr>
        <tr>
            <td></td>
            <td><ASP:Button Text="提交" runat="server" /></td>
        </tr>
    </table>

```

```
</tr>  
</table>  
</center>
```

在这个文件中，我们设置了控件的属性。使之能在 page.aspx 中调用程序的运行如图 4-3。

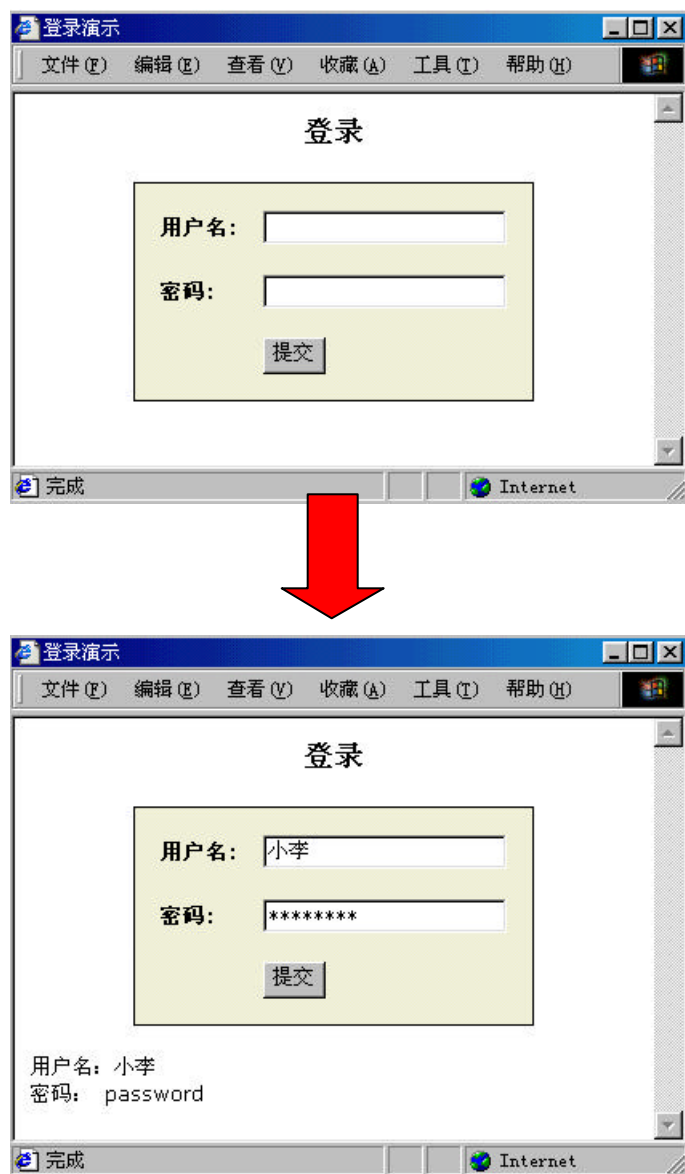


图 4-3

在下一个例子中，我们将使用 Page_OnLoad 事件，来执行数据绑定，代码如下：

```
<html>
<head>
<title>数据绑定演示</title>
  <script language="VB" runat="server">
    Sub Page_Load(sender As Object, e As EventArgs)
      If Not IsPostBack Then

        Dim values as ArrayList= new ArrayList()
        values.Add ("北京")
        values.Add ("上海")
        values.Add ("杭州")
        values.Add ("成都")
        values.Add ("重庆")
        values.Add ("西安")
        DropDownList.DataSource = values
        DropDownList.DataBind
      End If
    End Sub
    Sub SubmitBtn_Click(sender As Object, e As EventArgs)
      Label1.Text = "你选择的城市是： " + DropDownList.SelectedItem.Text
    End Sub
  </script>
</head>
<body>
<center><h3><font face="Verdana">数据绑定演示</font></h3></center>
  <form runat=server>
    <center><asp:DropDownList id="DropDown1" runat="server" /></center>
    <center><asp:button Text="提交" OnClick="SubmitBtn_Click"
runat=server/></center>
  <p>
    <center><asp:Label id=Label1 font-name="Verdana" font-size="10pt"
runat="server" /></center>
  </form>
</body>
</html>
```

程序运行结果如图 4-4。



图 4-4

当我们点击“提交”按钮时，如图 4-5 所示。

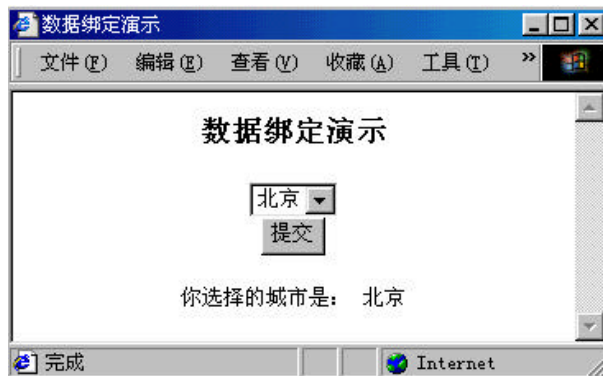


图 4-5

在下面的例子中将用 `page_load` 事件来对数据库进行连接。

还要说明的是，如果用 SQL 语句对数据库进行操作的时候，需要在页面中导入 `System.Data` 和 `System.Data.SQL` 名字控件，语句如下：

```
<%@ Import Namespace="System.Data" %>  
<%@ Import Namespace="System.Data.SQL" %>
```

程序代码如下 (`pagedata.aspx`):

```
<%@ Import Namespace="System.Data" %>  
<%@ Import Namespace="System.Data.SQL" %>  
<html>  
<script language="VB" runat="server">  
    Sub Page_Load(Src As Object, E As EventArgs)  
        Dim DS As DataSet  
        Dim MyConnection As SqlConnection
```

```
        Dim MyCommand As SQLDataSetCommand
        MyConnection = New
SQLConnection("server='iceberg';uid=sa;pwd=;database=info")
        MyCommand = New SQLDataSetCommand("select * from
infor",MyConnection)
        DS = New DataSet()
        MyCommand.FillDataSet(ds, "infor")
        MyDataGrid.DataSource=ds.Tables("infor").DefaultView
        MyDataGrid.DataBind()
    End Sub
</script>
<center>
<body>
<h3><font face="Verdana">Page_load事件演示</font></h3>
<ASP:DataGrid id="MyDataGrid" runat="server"
    Width="600"
    BackColor="white"
    BorderColor="black"
    ShowFooter="false"
    CellPadding=3

    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="8pt"
    HeaderStyle-BackColor="#aaaadd"
    MaintainState="false"
/>
/body>
</center>
</html>
```

在这个程序中的 page_load 事件中，我们做了哪些事呢？

与数据库连接。在这个例子中，我们使用 SQL Server 作为后台数据库。在这个库中，我们建立了 info 数据库，在数据库中有一张 infor 表。

执行 SQL 操作。

将筛选后的数据显示出来。

我们再来看看程序运行的结果，如图 4-6。



图 4-6

上面就是对 Page_load 事件的介绍，相信大家通过例子能对该事件有个理解。

4.4.3.2 事件处理

下面介绍处理表单的事件。你可以处理特定的事件，也可以在表单需要校验的情况下，根据 IsValid 属性判定页面的输入是否有效。

Web Form 提供了一些具有验证功能的服务器控件。这些控件提供了一套简单易用并且很强大的功能能检查输入时是否有错误。而且，还能显示提示信息给用户。

对于每个控件来说，都有一特定的属性，来验证输入的值是否有效。我们来看一下对输入控件需要验证的属性，如表 4-1 所示。

表 4-1

控件	需要验证的属性
HtmlInputText	Value
HtmlTextAreaHtm	Value
HtmlSelect	Value
HtmlInputFile	Value
TextBox	Text
ListBox	SelectedItem
DropDownList	SelectedItem
RadioButtonList	SelectedItem

好了，有了上面的介绍，我们以例子来讲解表单的有效性验证。

下面一个简单的例子是对用户的输入验证。

如 Validate.aspx 的内容如下：

```
<html>
<head>
  <script language="VB" runat="server">
    Sub ValidateBtn_Click(sender As Object, e As EventArgs)
      If (Page.IsValid) Then
        lblOutput.Text = "页面有效!"
      Else
        lblOutput.Text = "在页面中不能出现空项!"
      End If
      if not isnumeric(TextBox1.text) then
        lblOutput.Text="请输入数值!"
      End if
    End Sub
  </script>
</head>
<body>
<center><h3><font face="Verdana">验证表单的例子</font></h3></center>
<p>
<form runat="server">
<title>表单验证</title>
<center>
  <table bgcolor="white" cellpadding=10>
    <tr valign="top">
      <td colspan=3>
        <asp:Label ID="lblOutput" Text="请填写下面的内容" ForeColor="red"
Font-Name="Verdana" Font-Size="10" runat=server /><br>
      </td>
    </tr>

    <tr>
      <td align=right>
        <font face=Verdana size=2>储蓄卡类型:</font>
      </td>
      <td>
        <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow"
```

```
runat=server>
    <asp:ListItem>绿卡</asp:ListItem>
    <asp:ListItem>牡丹卡</asp:ListItem>
    </ASP:RadioButtonList>
</td>
<td align=middle rowspan=1>
    <asp:RequiredFieldValidator id="RequiredFieldValidator1"
        ControlToValidate="RadioButtonList1"
        Display="Static"
        InitialValue="" Width="100%" runat=server>
        *
    </asp:RequiredFieldValidator>
</td>
</tr>
<tr>
    <td align=right>
        <font face=Verdana size=2>卡号:</font>
    </td>
    <td>
        <ASP:TextBox id=TextBox1 runat=server />
    </td>
    <td>
        <asp:RequiredFieldValidator id="RequiredFieldValidator2"
            ControlToValidate="TextBox1"
            Display="Static"
            Width="100%" runat=server>
            *
        </asp:RequiredFieldValidator>
    </td>
</tr>
    <td>
</tr>
<tr>
    <td></td>
    <td>
        <ASP:Button id=Button1 text="验证" OnClick="ValidateBtn_Click"
runat=server />
    </td>
<td></td>
```

```

        </tr>
    </table>
</center>
</form>
</body>
</html>

```

我们对验证按钮的 clicked 事件进行编程，其中用到了 IsNumeric()函数，来判断变量是否为数值型的。我们还可以用 IsData()函数对输入的日期进行判断。我们需要说明的是合法的日期为 100 年 1 月 1 日到 9999 年 12 月 31 日。

运行结果如图 4-7：



图 4-7

当我们在卡号一栏中输入一些字母，而不是数值时，页面上将会提示你输入数值。

下面再举一个很有用的例子：

当用户在填写个人信息的时候，往往需要输入身份证号，如何进行身份证号的验证呢？

在讲解下面的程序之前，先看看我国的身份证号是如何编码的。

1 2 3 4 5

XX XXXX XXXXXX XX X （这个是没有升位以前的一个身份证号码的组成方式）

1 省 2 地市 3 生日 4 顺序码 5 性别

在这个例子中，我们只对省份进行判断。

身份编码一览表（表 4-2）。

表 4-2

北京	11	吉林	22	福建	35	广东	44	云南	53
天津	12	黑龙江	23	江西	36	广西	45	西藏	54
河北	13	上海	31	山东	37	海南	46	陕西	61
山西	14	江苏	32	河南	41	重庆	50	甘肃	62
内蒙古	15	浙江	33	湖北	42	四川	51	青海	63
辽宁	21	安徽	34	湖南	43	贵州	52	宁夏	64
新疆	65	台湾	71	香港	81	澳门	82	国外	91

在这个程序中，仅仅作了一个简单的判断。

Validate1.aspx 的文件内容如下：

```
<html>
<head>
  <script language="VB" runat="server">
    Sub ValidateBtn_Click(sender As Object, e As EventArgs)
      If (Page.IsValid) Then
        lblOutput.Text = "页面有效!"
      Else
        lblOutput.Text = "在页面中不能出现空项!"
      End If
      If not isnumeric(TextBox1.text) then
        bloutput.text="请输入数值!"
      End if
      if left$(textbox1.text,2)<>"11" then
        lbloutput.text="请验证你的身份证输入"
      End if
    End Sub
  </script>
</head>
<body>
<center><h3><font face="Verdana">验证表单的例子</font></h3></center>
<p>
<form runat="server">
<title>表单验证</title>
<center>
  <table bgcolor="white" cellpadding=10>
    <tr valign="top">
      <td colspan=3>
```



```
        <asp:Label ID="lblOutput" Text="请填写下面的内容" ForeColor="red"
Font-Name="Verdana" Font-Size="10" runat=server /><br>
    </td>
</tr>
<tr>
    <td align=right>
        <font face=Verdana size=2>身份证号:</font>
    </td>
    <td>
        <ASP:TextBox id=TextBox1 runat=server />
    </td>
    <td>
        <asp:RequiredFieldValidator id="RequiredFieldValidator2"
            ControlToValidate="TextBox1"
            Display="Static"
            Width="100%" runat=server>
            *
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td>
</tr>
<tr>
    <td></td>
    <td>
        <ASP:Button id=Button1 text="验证" OnClick="ValidateBtn_Click"
runat=server />
    </td>
<td></td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

在这个程序中，我们仅对北京地区的身份证号进行了验证，使用 `Left()` 函数把字符串的前两个字符取出进行比较。如果大家感兴趣的话，可以把这个程序补充完整。

程序的运行结果如图 4-8。



图 4-8

这是输入正确的情况，如输入不正确，则显示如图 4-9。



图 4-9

我们在验证的时候，有时需要进行特殊的验证。在下面的表 4-3 中，列出了需要进行特殊验证时要使用的特殊控件。

表 4-3

控件	描述
RequiredFieldValidator	使用户在输入时，不是使这一项为空
CompareValidator	对两个控件的值进行比较
RangeValidator	对输入的值进行控制，使其值界定在一定范围内
RegularExpressionValidator	把用户输入的字符和自定义的表达式进行比较
CustomValidator	自定义验证方式
ValidationSummary	在一个页面中显示总的验证错误

下面对各个验证控件介绍：

1. RequiredFieldValidator

在下面的这个例子中演示了 RequiredFieldValidator 控件的使用方法。

validate3.aspx 文件：

```
<html>
<body>
<center>
<title>验证控件演示 (1)</title>
<h3><font face="Verdana">验证控件演示 (1)</font></h3>
<form runat=server>
  姓名: <asp:TextBox id=Text1 runat="server"/>
  <asp:RequiredFieldValidator id="RequiredFieldValidator1"
ControlToValidate="Text1" Font-Name="Arial" Font-Size="11" runat="server">
    此项不能为空!
  </asp:RequiredFieldValidator>
  <p>
    <asp:Button id="Button1" runat="server" Text="验证" />
  </p>
</form>
</center>
</body>
</html>
```

当我们不在文本框中输入内容的时候，页面上将会出现不能为空的提示。

程序运行如图 4-10。



图 4-10

2. CompareValidator 控件

为了比较两个控件的值，此时我们需要使用 **CompareValidator** 控件。在下面的这个例子中将讲解 **CompareValidator** 控件的用法。

先看文件 `validata4.aspx`：

```
<%@ Page clienttarget=downlevel %>
<html>
<title>CompareValidator控件示例</title>
<head>
  <script language="VB" runat="server">
    Sub Button1_OnSubmit(sender As Object, e As EventArgs)
      If Page.IsValid Then
        lblOutput.Text = "比较正确!"
      Else
        lblOutput.Text = "比较不正确!"
      End If
    End Sub
    Sub lstOperator_SelectedIndexChanged(sender As Object, e As
EventArgs)
      comp1.Operator = lstOperator.SelectedIndex
      comp1.Validate
    End Sub
  </script>
</head>
<body>
<center>
  <h3><font face="Verdana">CompareValidator控件示例</font></h3>
```

```

<form runat=server>
  <table bgcolor="#e0e0e0" cellpadding=10>
    <tr valign="top">
      <td>
        <h5><font face="Verdana">字符串 1:</font></h5>
        <asp:TextBox Selected id="txtComp"
runat="server"></asp:TextBox>
      </td>
      <td>
        <h5><font face="Verdana">比较运算符:</font></h5>
        <asp:ListBox id="lstOperator"
OnSelectedIndexChanged="lstOperator_SelectedIndexChanged" runat="server">
          <asp:ListItem Selected Value="Equal" ></asp:ListItem>
          <asp:ListItem Value="NotEqual" ><</asp:ListItem>
          <asp:ListItem Value="GreaterThan" >></asp:ListItem>
          <asp:ListItem Value="GreaterThanEqual"
>></asp:ListItem>
          <asp:ListItem Value="LessThan" ><</asp:ListItem>
          <asp:ListItem Value="LessThanEqual" >=<</asp:ListItem>
        </asp:ListBox>
      </td>
      <td>
        <h5><font face="Verdana">字符串 2:</font></h5>
        <asp:TextBox id="txtCompTo" runat="server"></asp:TextBox><p>
        <asp:Button runat=server Text="验证" ID="Button1"
onclick="Button1_OnSubmit" />
      </td>
    </tr>
  </table>
  <asp:CompareValidator id="comp1" ControlToValidate="txtComp"
ControlToCompare = "txtCompTo" Type="String" runat="server"/>
  <br>
  <asp:Label ID="lblOutput" Font-Name="verdana" Font-Size="10pt"
runat="server"/>
</form>
</center>
</body>
</html>

```

在上面的代码中，我们实现了对两个控件的值进行比较。
程序运行如图 4-11。

当我们在两个文本框中输入值，然后选定运算符后，点验证按钮后，在页面上将显示比较结果。

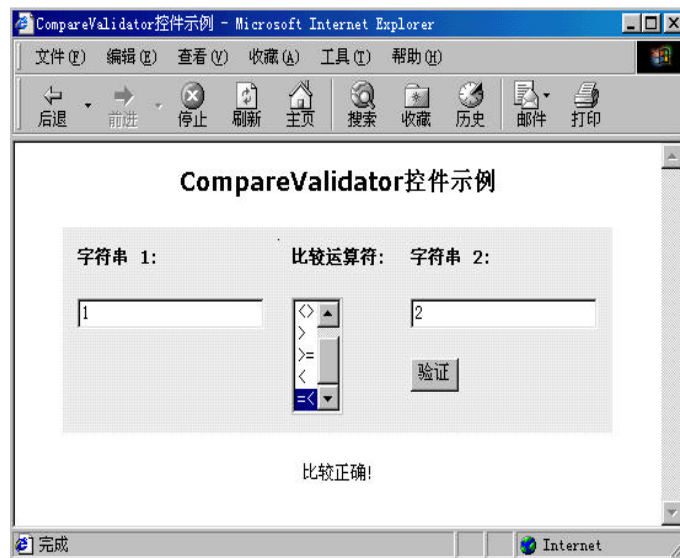


图 4-11

3. RangeValidator 控件

RangeValidator 控件主要界定输入的值的范围。因为有时我们要求输入的值是要有一定范围的，所以我们要使用 RangeValidator 来判断。

在下面的这个例子中介绍 RangeValidator 控件。

请看 validata5.aspx 的程序内容：

```
<%@ Page clienttarget=downlevel %>
<html>
<center>
<title>RangeValidator控件演示</title>
<head>
  <script language="VB" runat="server">
    Sub Button1_Click(sender As Object, e As EventArgs)
      If (Page.IsValid) Then
        lblOutput.Text = "结果正确!"
      Else
        lblOutput.Text = "结果不正确!"
      End If
    End Sub
  Sub lstOperator_SelectedIndexChanged(sender As Object, e As
  EventArgs)
```

```
        rangeVal.Type = lstType.SelectedIndex
        rangeVal.Validate
    End Sub
</script>
</head>
<body>

<h3><font face="Verdana">RangeValidator控件演示</font></h3>
<p>
<form runat="server">
    <table bgcolor="#eeeeee" cellpadding=10>
        <tr valign="top">
            <td>
                <h5><font face="Verdana">输入要验证的值:</font></h5>
                <asp:TextBox Selected id="txtComp" runat="server" />
            </td>
            <td>
                <h5><font face="Verdana">数据类型:</font></h5>
                <asp:DropDownList id="lstType"
OnSelectedIndexChanged="lstOperator_SelectedIndexChanged" runat="server">
                    <asp:ListItem Selected Value="String" >String</asp:ListItem>
                    <asp:ListItem Value="Integer" >Integer</asp:ListItem>
                </asp:DropDownList>
            </td>
            <td>
                <h5><font face="Verdana">最小值:</font></h5>
                <asp:TextBox id="txtMin" runat="server" />
            </td>
            <td>
                <h5><font face="Verdana">最大值:</font></h5>
                <asp:TextBox id="txtMax" runat="server" /><p>
                <asp:Button Text="验证" ID="Button1" onclick="Button1_Click"
runat="server" />
            </td>
        </tr>
    </table>
    <asp:RangeValidator id="rangeVal" Type="String"
ControlToValidate="txtComp" MaximumControl="txtMax" MinimumControl="txtMin"
runat="server" />
    <br>
    <asp:Label id="lblOutput" Font-Name="verdana" Font-Size="10pt"
runat="server" />
```

```
</form>
</body>
</center>
</html>
```

当我们在三个文本框中分别输入要验证的值，最大值和最小值，然后按下验证按钮，页面上将显示判断的结果。

在本例中我们只能比较 integer 和 string 的值，当然，我们也可以增加数据类型，如 double 型，float 型，date 型，currency 型等。

结果运行如图 4-12。



图 4-12

4. RegularExpressionValidator 控件

我们在制作网站的时候，尤其是各种电子商务网站，首先都会让用户填写一些表格来获取注册用户的各种信息，因为用户有可能输入各式各样的信息，而有些不符合要求的数据会给我们的后端 ASP 处理程序带来不必要的麻烦，甚至导致网站出现一些安全问题。因此我们在将这些信息保存到网站的数据库之前，要对这些用户所输入的信息进行数据的合法性校验，以便后面的程序可以安全顺利地执行。

使用 RegularExpressionValidator 服务器控件，可以用来检查我们输入的信息是否和我们的自定义的表达式一致。比方说用它可以检查 e-mail 地址，电话号码等合法性。

在 RegularExpressionValidator 服务器控件使用之前，先来了解一下正则表达式 (RegularExpression) 的来源。

正则表达式的“祖先”可以一直追溯到对人类神经系统如何工作的早期研究。Warren McCulloch 和 Walter Pitts 这两位神经生理学家研究出一种数学方式来描述这些神经网络。1956 年,一位叫 Stephen Kleene 的美国数学家在 McCulloch 和 Pitts 早期工作的基础上,发表了一篇标题为“神经网络事件的表示法”的论文,引入了正则表达式的概念。正则表达式就是用来描述他称为“正则集的代数”的表达式,因此采用“正则表达式”这个术语。随后,发现可以将这一工作应用于使用 Ken Thompson 的计算搜索算法的一些早期研究, Ken Thompson 是 Unix 的主要发明人。正则表达式的第一个实用应用程序就是 Unix 中的 qed 编辑器。如他们所说,剩下的就是众所周知的历史了。从那时起直至现在正则表达式都是基于文本的编辑器和搜索工具中的一个重要部分。

其实,正则表达式 (RegularExpression) 是一个正则表达式就是由普通字符 (例如字符 a 到 z) 以及特殊字符 (称为元字符) 组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板,将某个字符模式与所搜索的字符串进行匹配。

使用正则表达式,就可以:

1. 测试字符串的某个模式。例如,可以对一个输入字符串进行测试,看在该字符串是否存在一个电话号码模式或一个信用卡号码模式。这称为数据有效性验证。
2. 替换文本。可以在文档中使用一个正则表达式来标识特定文字,然后可以全部将其删除,或者替换为别的文字。
3. 根据模式匹配从字符串中提取一个子字符串。可以用来在文本或输入字段中查找特定文字。

例如,如果需要搜索整个 Web 站点来删除某些过时的信息并替换某些 HTML 格式化标记,则可以使用正则表达式对每个文件进行测试,看在该文件中是否存在所要查找的材料或 HTML 格式化标记。用这个方法,就可以将受影响的文件范围缩小到包含要删除或更改的信息的那些文件。然后可以使用正则表达式来删除过时的信息,最后,可以再次使用正则表达式来查找并替换那些需要替换的标记。

另一个说明正则表达式非常有用的示例是一种其字符串处理能力还不为人所知的语言。VBScript 是 Visual Basic 的一个子集,具有丰富的字符串处理功能。与 C 类似的 VB Scripting Edition 则没有这一能力。正则表达式给 VB Scripting Edition 的字符串处理能力带来了明显改善。不过,可能还是在 VBScript 中使用正则表达式的效率更高,它允许在单个表达式中执行多个字符串操作。

正是由于“正则表达式”的强大功能,才使得微软慢慢将正则表达式对象移植到了视窗系统上面。在书写正则表达式的模式时使用了特殊的字符和序列。下表描述了可以使用的字符和序列,并给出了实例。

字符描述: \: 将下一个字符标记为特殊字符或字面值。例如"n"与字符"n"匹配。"\n"与换行符匹配。序列"\"与"\"匹配, \"("与"("匹配。

`^` : 匹配输入的开始位置。

`$` : 匹配输入的结尾。

`*` : 匹配前一个字符零次或几次。例如, "`zo*`"可以匹配"`z`"、"`zoo`"。

`+` : 匹配前一个字符一次或多次。例如, "`zo+`"可以匹配"`zoo`",但不匹配"`z`"。

`?` : 匹配前一个字符零次或一次。例如, "`a?ve?`"可以匹配"`never`"中的"`ve`"。

`.` : 匹配换行符以外的任何字符。

`(pattern)` 与模式匹配并记住匹配。匹配的子字符串可以从作为结果的 `Matches` 集合中使用 `Item [0]...[n]`取得。如果要匹配括号字符(和 `)`, 可使用"`\("` 或 "`\)`"。

`x|y` : 匹配 `x` 或 `y`。例如 "`z|food`" 可匹配 "`z`" 或 "`food`"。"`(z|f)ood`" 匹配 "`zoo`" 或 "`food`"。

`{n}` : `n` 为非负的整数。匹配恰好 `n` 次。例如, "`o{2}`" 不能与 "`Bob`" 中的 "`o`" 匹配, 但是可以与 "`foooooo`" 中的前两个 `o` 匹配。

`{n,}` : `n` 为非负的整数。匹配至少 `n` 次。例如, "`o{2,}`"不匹配"`Bob`"中的"`o`", 但是匹配 "`foooooo`" 中所有的 `o`。"`o{1,}`"等价于 "`o+`"。"`o{0,}`"等价于 "`o*`"。

`{n,m}` : `m` 和 `n` 为非负的整数。匹配至少 `n` 次, 至多 `m` 次。例如, "`o{1,3}`" 匹配 "`foooooo`" 中前三个 `o`。"`o{0,1}`"等价于 "`o?`"。

`[xyz]` : 一个字符集。与括号中字符的其中之一匹配。例如, "`[abc]`" 匹配 "`plain`" 中的 "`a`"。

`^[xyz]` : 一个否定的字符集。匹配不在此括号中的任何字符。例如, "`^[abc]`" 可以匹配 "`plain`" 中的 "`p`"。

`[a-z]` : 表示某个范围内的字符。与指定区间内的任何字符匹配。例如, "`[a-z]`" 匹配 "`a`" 与 "`z`" 之间的任何一个小写字母字符。

`^[m-z]` : 否定的字符区间。与不在指定区间内的字符匹配。例如, "`[m-z]`" 与不在 "`m`" 到 "`z`" 之间的任何字符匹配。

`\b` : 与单词的边界匹配, 即单词与空格之间的位置。例如, "`er\b`" 与 "`never`" 中的 "`er`" 匹配, 但是不匹配 "`verb`" 中的 "`er`"。

`\B` : 与非单词边界匹配。"`ea*\B`" 与 "`never early`" 中的 "`ear`" 匹配。

`\d` : 与一个数字字符匹配。等价于 `[0-9]`。

`\D` : 与非数字的字符匹配。等价于 `^[^0-9]`。

`\f` : 与分页符匹配。

`\n` : 与换行符字符匹配。

`\r` : 与回车字符匹配。

`\s` : 与任何空白字符匹配, 包括空格、制表符、分页符等。等价于 "`[\f\n\r\t\v]`"。

`\S` : 与任何非空白的字符匹配。等价于 "`^[^ \f\n\r\t\v]`"。

`\t` : 与制表符匹配。

\v : 与垂直制表符匹配。

\w : 与任何单词字符匹配, 包括下划线。等价于 "[A-Za-z0-9_]"。

\W : 与任何非单词字符匹配。等价于 "[^A-Za-z0-9_]"。

\num : 匹配 num 个, 其中 num 为一个正整数。引用回到记住的匹配。例如, "(.)\1" 匹配两个连续的相同的字符。

\n : 匹配 n, 其中 n 是一个八进制换码值。八进制换码值必须是 1, 2 或 3 个数字长。

例如, "\11" 和 "\011" 都与一个制表符匹配。"\0011" 等价于 "\001" 与 "1"。八进制换码值不得超过 256。否则, 只有前两个字符被视为表达式的一部分。允许在正则表达式中使用 ASCII 码。

\xn : 匹配 n, 其中 n 是一个十六进制的换码值。十六进制换码值必须恰好为两个数字长。例如, "\x41" 匹配 "A"。"\x041" 等价于 "\x04" 和 "1"。允许在正则表达式中使用 ASCII 码。

RegularExpressionValidator 有两种主要的属性来进行有效性验证。ControlToValidate 包含了一个值进行验证。如取出文本框中的值。如 ControlToValidate="TextBox1" ValidationExpression 包含了一个正则表达式进行验证。

好了, 有了上面的叙述, 我们就举个例子来说明正则表达式。比如, 想要对用户输入的电子邮件进行校验, 那么, 什么样的数据才算是一个合法的电子邮件呢? 可以这样输入: test@yesky.com, 当然也会这样输入: xxx@yyy.com.cn, 但是这样的输入就是非法的: xxx@@com.cn 或者 @xxx.com.cn, 等等, 所以得出一个合法的电子邮件地址至少应当满足以下几个条件:

1. 必须包含一个并且只有一个符号 "@"
2. 第一个字符不得是 "@" 或者 "."
3. 不允许出现 "@." 或者 ".@"
4. 结尾不得是字符 "@" 或者 "."

所以根据以上的原则和上面表中的语法, 我们很容易的就可以得到需要的模板如下:

```
"= "^\w+((-w+)(\.\w+))*@[A-Za-z0-9]+((\|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$"
```

请看 validate5.aspx 的内容:

```
</head>
<body>
<center><h3><font face="Verdana">使用正则表达式验证</font></h3></center>
<p>
<form runat="server">
<center>
<title>使用正则表达式验证</title>
<table bgcolor="#e0e0e0" cellpadding=10>
<tr valign="top">
<td colspan=3>
```

```
<asp:Label ID="lblOutput" Text="输入E-mail地址" Font-Name="Verdana"
Font-Size="10pt" runat="server"/>
</td>
</tr>
<tr>
<td align=right>
<font face=Verdana size=2>E-mail:</font>
</td>
<td>
<ASP:TextBox id=TextBox1 runat=server />
</td>
<td>
<asp:RegularExpressionValidator id="RegularExpressionValidator1"
runat="server"
ControlToValidate="TextBox1"
ValidationExpression="^\w+((-w+)|(\.w+))*@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+
)*\.[A-Za-z0-9]+$"
Display="Static"
Font-Name="verdana"
Font-Size="10pt">
请输入有效的E-mail地址!
</asp:RegularExpressionValidator>
</td>
</tr>
<tr>
<td></td>
<td>
<ASP:Button text="验证" OnClick="ValidateBtn_Click" runat=server />
</td>
<td></td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

这样，我们只要定制不同的模板，就可以实现对不同数据的合法性校验了。所以，正则表达式对象中最重要的属性就是：“Pattern”属性，只要真正掌握了这个属性，才可以自由的运用正则表达式对象来为我们的数据校验进行服务。

程序的运行结果如图 4-13。

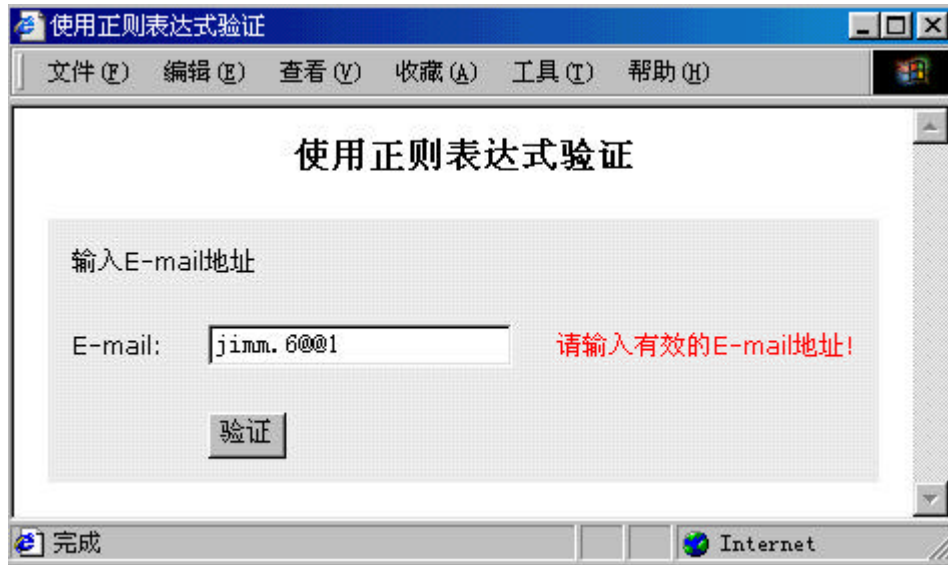


图 4-13

通过上面的介绍，我们对数据验证的方法有了一定的认识。在下面的介绍中还将通过更具体的实例，来对数据的有效性验证进行讲解。

4.4.3.3 Page_Unload

页面已经处理完毕，需要做些清理工作。一般地，可以在这个阶段关闭打开的文件和数据库链路，或者释放对象。

1. 断开数据库连接

请看如下脚本：

```
<script language="VB" runat="server">
'定义一个共有变量
    public Dim MyConnection As SqlConnection
'定义Page_Load事件
    Sub Page_Load(Src As Object, E As EventArgs)
        Dim DS As DataSet

        Dim MyCommand As SQLDataSetCommand
        MyConnection = New
        SqlConnection("server='iceberg';uid=sa;pwd=;database=info")
        MyCommand = New SQLDataSetCommand("select * from infor",MyConnection)
        Myconnection.open()
        DS = New DataSet()
        MyCommand.FillDataSet(ds, "infor")
    End Sub
End Class
```

```
MyDataGrid.DataSource=ds.Tables("infor").DefaultView
MyDataGrid.DataBind()
End Sub
'定义Page_UnLoad事件
Sub Page_UnLoad(Src As Object, E As EventArgs)
'与数据库断开连接
MyConnection.Close()
End Sub
```

现在再来看一个对文件操作的例子。

在这个例子中，我们使用的了两个事件，Page_Load 事件和 Page_Unload 事件。在 Page_Load 事件先创建一个文件，然后向这个文件中写入内容。在 Page_Unload 事件中我们将此文件关闭。

代码如下：

```
<%@ import namespace="system.io" %>
<html>
<head>
<title>ASP+ 测试 写 文本文件</title>
</head>
<body>
<script language="vb" runat="server">
public Dim writeFile As StreamWriter
Sub Page_Load(Sender As Object,E as EventArgs)
writeFile = File.CreateText( "c:\test.txt" )
writeFile.WriteLine( "这是一个测试文件!" )
writeFile.WriteLine( "使用了Page_Load事件和Page_Unload事件!" )
Response.Write( "test.txt 创建 并 写入 成功!" )
End Sub
Sub Page_UnLoad(Sender AS Object, E as EventArgs)
writeFile.Close
End Sub
</script>
</body>
</html>
```

这样，就使用了 Page_Load 事件和 Page_Unload 事件。很明显，定义 Page_Load 事件，是因为这个阶段页面已经处理完毕，需要做些清理工作。

上面分析了页面处理最重要的几个阶段。应该说明的是：页面的处理过程远比上面的复杂，因为每个控件都需要初始化。在后面的章节中，还将了解到更加详细的页面处理过程。

4.5 Web Form 事件模型

在 ASP.NET 中，事件是一个非常重要的概念。下面举两个例子来说明在 Web Form 中的应用。

4.5.1 例子 1：多按钮事件

在一个<form></form>里面有几个按钮，多个事件的响应该怎么处理呢？在 ASP.NET 中有很好的处理机制，可以在一个页面中用几个方法来分别响应不同的事件。

根据五个按钮的功能，定义五种方法：AddBtn_Click(Sender As Object, E As EventArgs)、AddAllBtn_Click(Sender As Object, E As EventArgs)、RemoveBtn_Click(Sender As Object, E As EventArgs)、RemoveAllBtn_Click(Sender As Object, E As EventArgs)、result(Sender As Object, E As EventArgs)，分别用来处理全部加进、单个加进、单个取消、全部取消和提交事件。form 提交的时候，还是提交给本页面，由本页面进行处理：

```
<form action="menent.aspx" runat=server>
menent.aspx 就是本页面。
<html>

<script language="VB" runat="server">

    Sub AddBtn_Click(Sender As Object, E As EventArgs)

        If Not (AvailableFonts.SelectedIndex = -1)
            InstalledFonts.Items.Add(New ListItem(AvailableFonts
.SelectedItem.Value))
            AvailableFonts.Items.Remove(AvailableFonts.SelectedItem.Value)
        End If
    End Sub

    Sub AddAllBtn_Click(Sender As Object, E As EventArgs)

        Do While Not (AvailableFonts.Items.Count = 0)
            InstalledFonts.Items.Add(New
ListItem(AvailableFonts.Items(0).Value))
            AvailableFonts.Items.Remove(AvailableFonts.Items(0).Value)
        Loop
    End Sub

    Sub RemoveBtn_Click(Sender As Object, E As EventArgs)
```

```
        If Not (InstalledFonts.SelectedIndex = -1)
            AvailableFonts.Items.Add(New
ListItem(InstalledFonts.SelectedItem.Value))

InstalledFonts.Items.Remove(InstalledFonts.SelectedItem.Value)
        End If
    End Sub

    Sub RemoveAllBtn_Click(Sender As Object, E As EventArgs)

        Do While Not (InstalledFonts.Items.Count = 0)
            AvailableFonts.Items.Add(New
ListItem(InstalledFonts.Items(0).Value))
            InstalledFonts.Items.Remove(InstalledFonts.Items(0).Value)
        Loop
    End Sub

    Sub result(Sender As Object,E As EventArgs)

        dim tmpStr as String

        tmpStr="<br>"
        Do While Not (InstalledFonts.Items.Count = 0)
            tmpStr=tmpStr & InstalledFonts.items(0).value & "<br>"
            InstalledFonts.items.remove(InstalledFonts.items(0).value)
        Loop
        tmpStr=System.Web.HttpUtility.UrlEncodeToString(tmpStr,System.Text.Encoding.UTF
8)
        Page.Navigate("result.aspx?InstalledFonts=" & tmpStr)

    End Sub

</script>

<body bgcolor="#ccccff">
<center>
    <h3><font face="Verdana">.NET->不同事件的处理方法!</font></h3>
</center>
<center>
    <form action="menent.aspx" runat=server>
```



```
<table>
  <tr>
    <td>
      现有字体
    </td>
    <td>
      <!-- Filler -->
    </td>
    <td>
      选择的字体
    </td>
  </tr>
  <tr>
    <td>
      <asp:listbox id="AvailableFonts" width="100px"
runat=server>
        <asp:listitem>Roman</asp:listitem>
        <asp:listitem>Arial Black</asp:listitem>
        <asp:listitem>Garamond</asp:listitem>
        <asp:listitem>Somona</asp:listitem>
        <asp:listitem>Symbol</asp:listitem>
      </asp:listbox>
    </td>
    <td>
      <!-- Filler -->
    </td>
    <td>
      <asp:listbox id="InstalledFonts" width="100px"
runat=server>
        <asp:listitem>Times</asp:listitem>
        <asp:listitem>Helvetica</asp:listitem>
        <asp:listitem>Arial</asp:listitem>
      </asp:listbox>
    </td>
  </tr>
  <tr>
    <td>
      <!-- Filler -->
    </td>
    <td>
      <asp:button text="<==< OnClick="RemoveAllBtn_Click"
runat=server/>
```

```
                <asp:button text="<--" OnClick="RemoveBtn_Click"
runat=server/>
                <asp:button text="-->" OnClick="AddBtn_Click"
runat=server/>
                <asp:button text="==>>" OnClick="AddAllBtn_Click"
runat=server/>
                <asp:label id="Message" forecolor="red" font-bold="true"
runat=server/>
                </td>
            </tr>
        <tr align=center>
        <td align=center>
            <asp:button text="提交" Onclick="result" runat=server/>
        <!-- Filler -->
        </td>
        </tr>
    </table>

    </form>
</center>
</body>

</html>
```

写一个页面，在提交时候接收相关信息。在页面进入的时候取得传送过来的数值，用：

`Request.Params("InstalledFonts")`

来获得，具体来看文件 `result.aspx` 的代码：

```
<html>
    <script language="VB" runat="server">
        Sub Page_Load(Sender As Object, E As EventArgs)
            If Not (Page.IsPostBack)
                NameLabel.Text = Request.Params("InstalledFonts")
            End If
        End Sub
    </script>

    <BODY BGCOLOR="#CCCCFF">
        <h3><font face="Verdana">.NET->多事件处理!</font></h3>
        <p>
        <p>
        <hr>
        <form action="controls_NavigationTarget.aspx" runat=server>
            <font face="Verdana">
```

```
Hi,你的选择是: <asp:label id="NameLabel" runat=server/>!  
</font>  
</form>  
</body>  
</html>
```

运行结果如图 4-14。

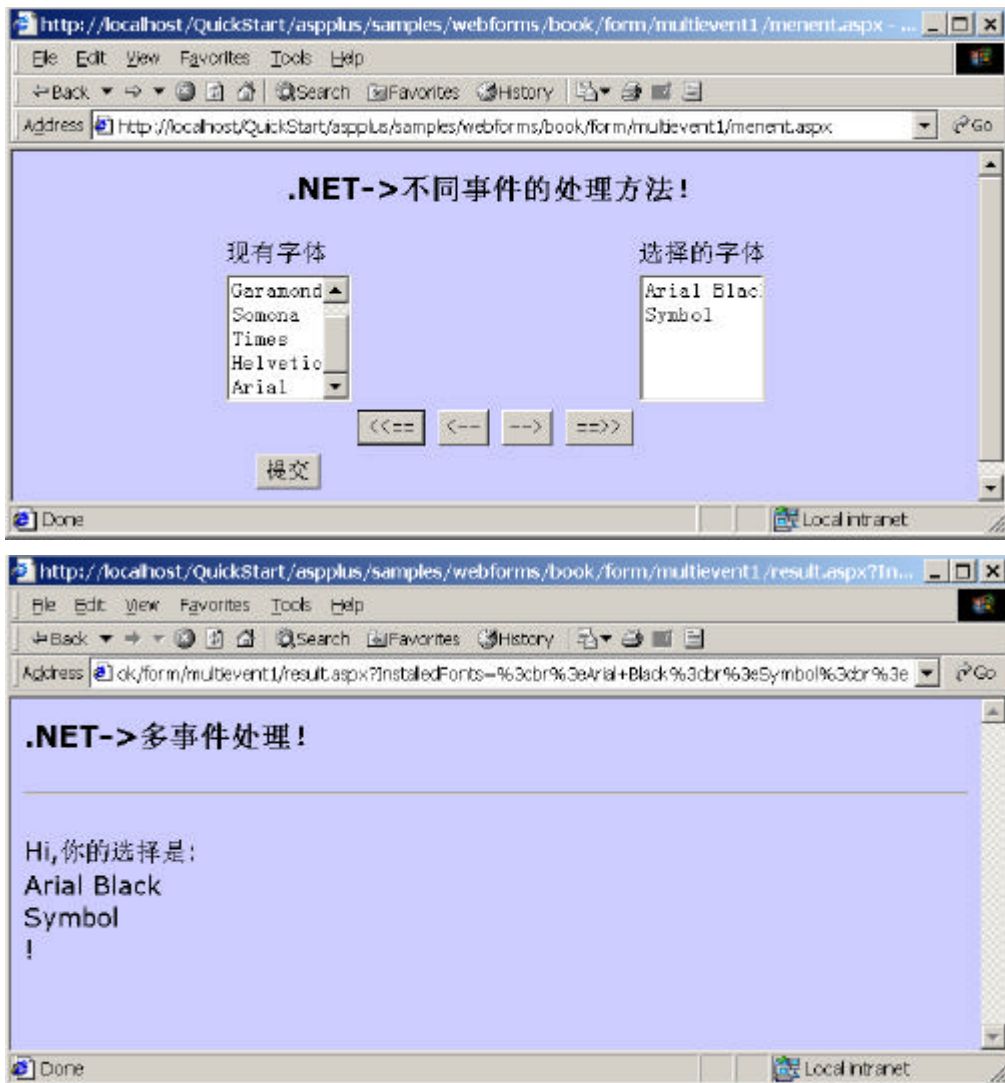


图 4-14

4.5.2 例子 2 : AutoPostBack

PostBack 属性在 Page_Load 事件中出现的, 在一个用户请求结束后, 如果页面重新 Load, 则返回一个 true。这对初始化一个页面来讲是一件非常容易的事情, 下面看代码:

```
void Page_Load(Object Sender, EventArgs e) {  
    if ((IsPostBack) & (TextBox2.Text == "")) {  
        TextBox2.Text="Hello " + TextBox1.Text + "!! 你好啊!";  
    }  
}
```

如果 IsPostBack 返回一个真值并且 TextBox2.Text 为空, 程序执行它下面的语句。在另外一个方面, 设置一个标示:

```
<asp:TextBox id="TextBox1" Text="请在这里输入你的名字! 并按下<Tab>"  
    AutoPostBack="True" Columns=50 runat="server"/>
```

设定 AutoPostBack="True", 自动 PostBack, 下面是完整的代码:

```
<html>  
<head>  
<script language="C#" runat="server">  
  
    void Page_Load(Object Sender, EventArgs e) {  
        if ((IsPostBack) & (TextBox2.Text == "")) {  
            TextBox2.Text="Hello " + TextBox1.Text + "!! 你好啊!";  
        }  
    }  
  
</script>  
</head>  
<body bgcolor="#ccccff">  
<center>  
    <br><br><br>  
    <h3><font face="Verdana">.NET->AutoPostBack技术</font></h3>  
    <br><br>  
</center>  
<center>  
<form runat="server">  
    <p>  
        <asp:TextBox id="TextBox1" Text="请在这里输入你的名字! 并按下<Tab>"  
            AutoPostBack="True" Columns=50 runat="server"/>  
    <p>  
        <asp:TextBox id="TextBox2" Columns=50 runat="server"/>  
    <p>  
        <asp:Button Text="提交" Runat="server"/>
```

```
<p>  
</form>  
</center>  
</body>  
</html>
```

访问结果如图 4-15 所示。



图 4-15

输入完成后，按下<Tab>键，得到如图 4-16 结果：

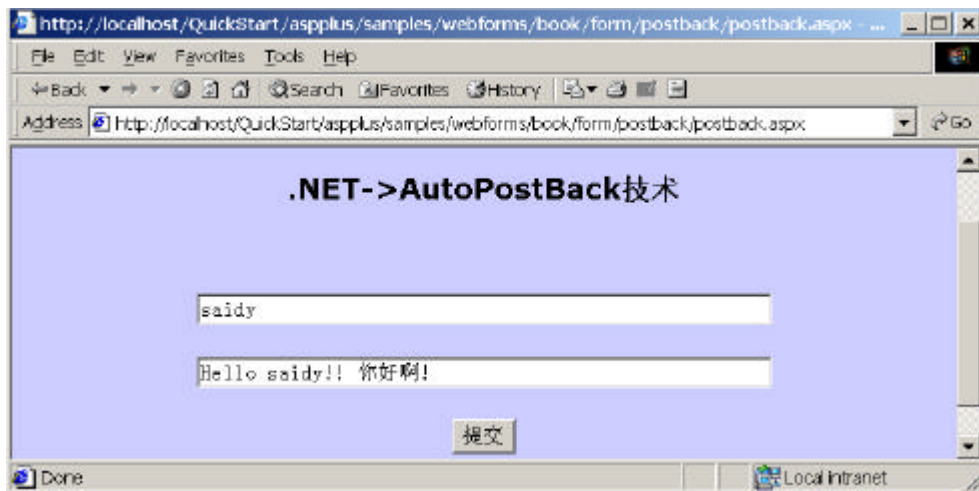


图 4-16

第 5 章 深入讲解 ASP+验证

5.1 简介

本章将详细的讲解 ASP.NET 验证控件的工作方式，验证控件可以生成包含验证控件的复杂页面，或是要扩展验证框架。

5.2 入门

在整个 ASP+ 开发过程中，了解验证是非常重要的事情。如今的大多数商业 Web 站点，这些站点中有许多表单，表单明显是通过执行大量手写的代码来执行验证。编写验证代码并不是一件有趣的工作。

如果要通过编写代码来显示数据表或动态生成图表，可能会很吸引人，但是这种方法不能够禁止在姓名字段中输入空值。

因为其它一些原因，Web 应用程序的验证也是非常麻烦的。HTML 3.2 可以控制的内容或可以从用户处得到的反馈的限制很多，因此无法应用在功能更全的客户机上可以使用的技巧，例如禁止用户输入某些字符，或发出响声。使用浏览器脚本可能会产生更强大的验证。但是这种方法很难得以证实，因为客户浏览器中并非一定有脚本，并且恶意的用户可以绕过。因此，为了保证站点安全，有必要对服务器进行同样的检查。

在开发 ASP.NET 时，我们的初衷是只使用一个控件来处理验证，可能本该是一个能够显示错误的 TextBox 控件。可是到了设计该控件时，却发现无法实现这种愿望。但是下面的数据输入表单具有许多有用的特性：

- ☞ 尽管错误信息或图标经常与输入元素相邻，但是它们几乎总是位于表的不同单元格中；
- ☞ 页面中经常会有一个区域来汇总所有错误。
- ☞ 许多站点包含客户端脚本，以便提供更快捷的反馈，同时防止白白地在与服务器之间往返。
- ☞ 许多包含客户端脚本的站点在出现错误时会显示信息框。
- ☞ 不仅会验证文本输入，还会验证下拉列表和单选按钮。
- ☞ 如果某个字段为空，站点通常会显示与该条目无效时不同的信息或图标。
- ☞ 许多有效性检查可以很好地代替常用的表达式。
- ☞ 验证通常是基于两个输入之间的比较结果。
- ☞ 90% 或 90% 以上的验证任务是一些常见的操作，例如检查姓名或邮政编码。大多数站点似乎仍在重复进行这些工作。

因为站点之间的差别通常太大，无法获得一种完美的解决方案来处理每个站点的所有验证任务。

考虑了上述所有情况，最终获得的解决方案包括五个验证器控件、ValidationSummary 控件以及与 Page 对象的集成。同时很明显，该解决方案需要扩展，在客户机和服务器上均需要有一个 API 来配合。

在大多数组件环境中，例如 Microsoft ActiveX，可能本来试图将所有验证控件的功能集成到一个控件中，处理不同模式下的不同属性。不过 Microsoft .NET 框架中有神奇的继承性，可以提供一套控件来对特定的属性进行特定的验证，因为派生每个新控件所需的额外工作量非常小。

这些控件所完成的大多数工作均在其公用的父级 BaseValidator 中实现。也可以从 BaseValidator 或其它控件派生来完成各项工作。实际上，即使 BaseValidator 都懒得实现其自己的 Text 属性，而是从 Label 属性继承。

5.3 何时发生何事

在处理包含验证 Web 控件的页面时，了解事件序列非常有效。如果某个验证条件是可选的，需要准确了解客户机和服务器上何时进行验证。如果要个人编写验证例程，可能会非常耗时，或者有副作用。同时，了解调用验证例程的时机也很重要。

5.4 服务器端的验证序列

了解页面的有效期非常重要，如果习惯于在 Visual Basic 或类似功能齐全的客户机工具中处理表单，则需要花一定的时间来了解。页面和页面上的所有对象并非在与用户交互时一直有效，尽管有时表面上是这样。

以下是在第一次访问某个页面时一个简化的事件序列：

1. 基于 ASPX 文件创建页面及其控件。
2. 触发 Page_Load 事件。
3. 页面和控件属性保存在一个隐藏字段中。
4. 页面和控件转换到 HTML。
5. 丢弃所有内容。

现在，当用户单击某个按钮或类似控件时，将返回服务器，然后执行一个类似的事件序列。该序列称为返回序列：

1. 基于 ASPX 文件创建页面及其控件。
2. 从隐藏字段恢复页面和控件属性。
3. 根据用户输入更新页面控件。
4. 触发 Page_Load 事件。

5. 触发更改通知事件。
6. 页面和控件属性保存在一个隐藏字段中。
7. 页面和控件转换到 HTML。
8. 再次丢弃所有内容。

为什么不将所有对象保留在内存中呢？因为使用 ASP.NET 建立的 Web 站点无法处理数量非常大的用户。因此，服务器的内存中只保留马上要处理的内容。

何时进行服务器端验证？在第一次获取页面信息时，根本不会进行服务器端验证。大多数最终用户都非常认真，允许用户自己确认在表单中填写的信息是否正确，然后再使用红色的文字通知用户填错的信息。

在返回事件序列中，第 3 步和第 4 步之间会进行验证。也就是说，进行验证是在来自用户的数据装回控件属性后，但在大多数代码执行之前。这意味着在编写用户事件代码时，通常可以利用已经进行的验证。一般情况下，你都会希望这样做。

在该时刻进行验证的缺点是：如果要通过编程来修改某些影响该验证的属性，该时刻就太迟了。例如，如果通过编写代码来启用或禁用验证控件或更改验证控件的属性，在下次处理该页之前，不会看到任何影响。通过以下两种方法可以避免这个问题：

☞ 在进行验证之前修改属性。

☞ 在属性更改之后重新验证控件。

这两种方法均需要使用在 Page 对象上有效的验证属性和方法。

5.5 页面 API

Page 对象包含一些与服务器端验证有关的重要属性和方法。表 5-1 中总结了这些属性和方法。

表 5-1 Page 对象的属性和方法

属性或方法	说明
IsValid 属性	这是最有用的属性。该属性可以检查整个表单是否有效。通常在更新数据库之前进行该检查。只有 Validators 集中的所有对象全部有效，该属性才为真，并且不将该值存入缓存。
Validators 属性	该页所有验证对象的集合。这是实现 IValidator 界面的对象的集合。
Validate 方法	在验证时调用的一种方法。在 Page 对象上默认的执行方式是转至每个验证器，并要求各验证器自行评估。

Validators 集合对于许多任务都非常有用。该集合是实现 IValidator 界面的对象的集合。之所以使用对象这个词，而不是使用控件，是因为 Page 对象只关注 IValidator 界面。既然所有的验证器通常都是用来实现 IValidator 的一些可视化控件，那么任何人都应能够

使用任意的验证对象，并将验证对象加入页面中。

IValidator 界面包含以下属性和方法，如表 5-2 所示。

表 5-2 IValidator 界面的属性和方法

属性或方法	说明
IsValid 属性	指出单独的验证对象进行的有效性检查是否已经通过。可以在验证后手工更改该值。
ErrorMessage 属性	介绍验证对象要验证的错误以及可能会向用户显示的错误。
Validate 方法	对验证对象执行有效性检查，以更新其 IsValid 值。

可以使用该界面执行一些有趣的任务。例如，要将页面重置为有效的状态，请使用以下代码（如 C# 中的示例所示）：

```
IValidator val;
foreach(val in Validators) {
    Val.IsValid = true;
}
```

要重新执行整个验证序列，请使用以下代码：

```
IValidator val;
foreach(val in Validators) {
    Val.Validate();
}
```

如果有 Beta 1 版或更高版本，也可以只对 Page 对象调用 Validate 方法，这样可以完成相同的任务。要在验证前进行某些更改，可以覆盖 Validate 方法。本例显示一个包含验证器的页面，其中的验证器根据复选框的值开或关：

```
public class Conditional : Page {
    public HtmlInputCheckBox chkSameAs;
    public RequiredFieldValidator rfvalShipAddress;

    protected override void Validate() {
        //只检查到货地址（如果与付款地址不同）
        bool enableShip = !chkSameAs.Checked;
        rfvalShipAddress.Enabled = enableShip;
        //现在执行验证
        base.Validate();
    }
}
```

5.6 客户端的验证

如果页面启用了客户端验证，则在往返过程中会发生完全不同的事件序列。客户端的验证使用客户端 JScript® 实现。实现该验证不需要任何二进制组件。

尽管 JScript 语言的标准化做得很好，但是用于与浏览器中的 HTML 文档交互的文档对象模型 (Document Object Model, DOM) 没有广泛采用的标准。因此，客户端的验证只在 Internet Explorer 4.0 和更高版本中进行，因为该验证的对象是 Internet Explorer DOM。

从服务器的角度来说，客户端的验证只意味着验证控件将不同的内容发送到 HTML 中。除此之外，其事件序列完全相同。服务器端的检查仍然执行。尽管看起来似乎多余，但是却十分重要，因为：

❖ 某些验证控件可能不支持客户端脚本。有一个很好的例子：如果要同时使用 CustomValidator 和服务端验证函数，但是没有客户端验证函数。

❖ 安全性注意事项。某些人很容易得到一个包含脚本的页面，然后禁用或更改该页面。不应利用脚本来阻止坏数据进入系统，而只应是为了用户得到更快的反馈。因此，如果要使用 CustomValidator，则不应提供没有相应服务端验证函数的客户端验证函数。

每个验证控件都可以确保将一个标准的客户端脚本块发送到页面中。实际上，这只是一小部分代码，其中包含对脚本库 WebUIValidation.js 中的代码的引用。这个脚本库文件包含客户端验证的所有逻辑，该文件需单独下载，并且可以存储在浏览器的缓存中。

关于脚本库

因为验证 Web 控件脚本在脚本库中，所以不必将所有客户端验证的代码直接发送到页面中，尽管表面上似乎是这样做的，主要的脚本文件引用类似如下所示：

```
<script language="javascript"
src="/_aspx/1.0.9999/script/WebUIValidation.js"></script>
```

默认情况下，脚本文件将安装在 "_aspx" 目录中默认的根目录下，并使用相对于根脚本 include 指令调用，该指令以正斜线开头。该引用表明每个单独的对象不必包含脚本库，同一台计算机上的所有页面可以引用同一个文件。你会注意到，该路径中还有一个公用的语言运行时版本号，以便不同的运行时版本可以在同一台计算机上运行。

如果查看一下默认的虚拟根目录，会找到该文件并查看其中的内容。这些文件的位置在 config.Web 文件中指定。config.Web 文件是一个用于大多数 ASP.NET 设置的 XML 文件。以下是该文件中位置的定义：

```
<Webcontrols
  clientscriptslocation="/_aspx/{0}/script/"
/>
```

在运行时版本更新时，这些脚本可能也需要相应的更新，你将或者放弃更改，或者面临脚本不工作的问题。如果特定项目必须更改这些脚本，先备份这些脚本，然后将项目指

向备份文件，方法是使用私有的 config.Web 文件替代这些文件的位置。如果字符串中包含格式指令 "{0}"，运行时版本号将替换该指令。最好将该位置更改为一个相对引用或绝对引用。

禁用客户端的验证

有时可能不希望进行客户端验证，如果输入字段的数目很少，客户端验证可能用处不大。毕竟每次都要有一个需要往返服务器一次的逻辑，会发现客户机上动态出现的信息对布局会有负面影响。

要禁用客户端验证，应使用 Page 指令 "clienttarget=downlevel"。该指令类似以下 ASPX 文件的开头：

```
<%@ Page Language="c#" clienttarget=downlevel %>
```

该指令的默认值为 "auto"，表示只对 Microsoft Internet Explorer 4.0 或更高版本进行客户端验证。

注意：在 Beta 1 中，该指令并非仅仅是禁用验证，同时还会使所有 Web 控件使用 HTML 3.2 标记来处理，这可能会产生意想不到的结果。最终版本提供了更好的方法来控制这个问题。

客户端事件序列

该序列是在运行包含客户端验证的页面时发生的事件序列：

1. 在页面载入浏览器时，需要对每个验证控件进行一些初始化。这些控件作为 标记发送，其 HTML 特性与服务器的特性最接近。最重要的是，此时会将验证器引用的所有输入元素“挂接”。被引用的输入元素将修改其客户端事件，以便在每次输入更改时调用验证例程。
2. 脚本库中的代码将在用户使用 tab 键在各字段之间切换时执行。某个独立的字段更改时，将重新评估验证条件，根据需要使验证器可见或不可见。
3. 当用户尝试提交表单时，将重新评估所有验证器。如果这些验证器全部有效，表单将提交给服务器。如果存在一处或多处错误，则会出现下述情况：
 - ☞ 提交被取消。表单并不提交给服务器。
 - ☞ 所有无效的验证器均可见。
 - ☞ 如果某个验证摘要包含 ShowSummary=true，则将收集来自验证控件的所有错误，并使用这些错误更新其内容。
 - ☞ 如果某个验证摘要包含 ShowMessageBox=true，则将收集错误，并在客户机的信息框中显示这些错误。

因为在每次输入更改时或提交时会执行客户端验证控件，所以在客户机上通常会评估这些验证控件两次或两次以上。请注意，提交后，仍将会在服务器上对这些验证控件进行重新评估。

客户端 API

有一个可以在客户机上使用的小型 API，以便在客户端代码中实现各种结果。因为某些例程不可能隐藏，所以理论上讲，可以利用客户端验证脚本所定义的所有变量、特性和函数。不过，其中许多都是可以更改的实施细节。表 5-3 总结了使用的客户端对象。

表 5-3 客户端对象

名称	类型	说明
Page_IsValid	Boolean 变量	指出页面当前是否有效。验证脚本总是保持该变量为最新。
Page_Validators	元素数组	这是包含页面上所有验证器的数组。
Page_ValidationActive	Boolean 变量	指出是否应进行验证。将此变量设置为 False 可以通过编程关闭验证。
isvalid	Boolean 属性	每个客户端验证器均具有该属性，指出验证器当前是否有效。请注意，在 PDC 版本中，该属性混用大小写 ("IsValid")。

绕过客户端验证

经常需要执行的一项任务是在页面上添加“取消”按钮或导航按钮。在这种情况下，即使页面上有错误，可能也希望使用该按钮提交页面。因为客户端按钮 "onclick" 事件在表单的 "onsubmit" 事件之前发生，因此可能会避免提交检查，并绕过验证。以下说明如何使用 HTML Image 控件作为“取消”按钮完成该任务：

```
<input type=image runat=server
  value="取消"
  onclick="Page_ValidationActive=false;"
  OnServerClick=cmdCancel_Click >
```

使用 Button 或 ImageButton 控件执行该任务会出现一些混淆，因为 "onclick" 事件假定为同名的服务器端事件。应在客户端脚本中设置该事件：

```
<asp:ImageButton runat=server id=cmdImgCancel
  AlternateText="取消"
  OnClick=cmdCancel_Click/>

<script language="javascript">
  document.all["cmdImgCancel"].onclick =
    new Function("Page_ValidationActive=false;");
</script>
```

解决该问题的另一种方法是：对“取消”按钮进行一定的设置，使其在返回时不会触发客户端脚本中的提交事件。HtmlInputButton 和 LinkButton 控件就是这样的例子。

特殊结果

另一种常见的要求是：在出错时，除了由验证器自身显示的错误信息外，还需要其它

一些结果。在这种情况下，任何修改均需在服务器或客户机上同时进行。假设需要加入一个 Label，根据输入是否有效来更改颜色。以下是如何在服务器上实现该任务：

```
public class ChangeColorPage : Page {
    public Label lblZip;
    public RegularExpressionValidator valZip;

    protected override void OnLoad(EventArgs e) {
        lblZip.ForeColor = valZip.IsValid? Color.Black : Color.Red;
    }
}
```

上述方法一切都很完美，但是，只要如上所述修改验证，就会发现除非在客户机上进行相同的操作，否则看起来会非常不一致。验证框架会避免许多这种双重结果，但是无法避免必须在客户机和服务器上同时实现的其它结果，以下是在客户机上执行同一任务的片段：

```
<asp:Label id=lblZip runat=server
    Text="Zip Code:" />
<asp:TextBox id=txtZip runat=server
    OnChange="txtZipOnChange();" /></asp:TextBox><br>
<asp:RegularExpressionValidator id=valZip runat=server
    ControlToValidate=txtZip
    ErrorMessage="无效的邮政编码"
    ValidationExpression="[0-9]{5}" /><br>

<script language=javascript>
function txtZipOnChange() {
    //如果客户端验证未处于活动状态，则不执行任何操作
    if (typeof(Page_Validators) == "undefined") return;
    //更改标签的颜色
    lblZip.style.color = valZip.isvalid ? "Black" : "Red";
}
</script>
```

Beta 1 客户端 API

对于 Beta 1 版，一些可以从客户端脚本调用的函数会造成其它一些情况，如表 5-4 所示。

表 5-4 从客户端脚本调用的函数

名称	说明
ValidatorValidate(val)	将某个客户端验证器作为输入。使验证器检查其输入并更新其显示。

(续表)

名称	说明
ValidatorEnable(val, enable)	获取一个客户端验证器和一个 Boolean 值。启用或禁用客户端验证器。如果禁用, 将不会评估客户端验证器, 客户端验证器将总是显示为有效。
ValidatorHookupControl(control, val)	获取一个输入 HTML 元素和一个客户端验证器。修改或创建该元素的 change 事件, 以便在更改时更新验证器。该函数适合于基于多个输入值的自定义验证器。

其特殊用途是启用或禁用验证器, 如果希望验证只是在特定的情况下生效, 可能需要在服务器和客户机上同时更改激活状态, 否则, 会发现用户无法提交该页面。

以下是上面的示例加上一个字段, 该字段只在取消选中某个复选框时才会进行验证。

```
public class Conditional : Page {
    public HtmlInputCheckBox chkSameAs;
    public RequiredFieldValidator rfvalShipAddress;
    protected override void Validate() {
        bool enableShip = !chkSameAs.Checked;
        rfvalShipAddress.Enabled = enableShip;
        base.Validate();
    }
}
```

以下是客户端等效的代码:

```
<input type=checkbox runat=server id=chkSameAs
    onclick="OnChangeSameAs();" >与付款地址相同<br>
<script language=javascript>
function OnChangeSameAs() {
    var enableShip = !event.srcElement.status;
    ValidatorEnable(rfvalShipAddress, enableShip);
}
</script>
```

5.7 有效性规则和有用的错误信息

每个验证器会显示有关特定控件特定情况的特定错误信息, 其中有一些确认是否有效的规则, 开始, 作为一个开发人员可能会有些混淆, 但是如果生成对用户有实际帮助的错误信息, 这些规则是必要的。

所有空的验证器 (除了 RequiredFieldValidator) 均会被认为有效。如果某个空值无效, 通常需要一个 RequiredFieldValidator 和一个其它验证器, 需要这样做, 因为一般情况下, 总是希望对空验证器和有效性显示不同的错误信息。也可以使用不明确的信息, 例如“你必

须输入一个值，并且该值必须在 1 和 10 之间”。

在输入字段无法转换为指定数据类型时使用的另一个特殊规则与 CompareValidator 和 RangeValidator 有关。对指定了 ControlToCompare 的 CompareValidator 进行的有效性评估过程类似如下所述：

1. 如果 ControlToValidate 引用的输入字段为空，则有效。
2. 如果 ControlToValidate 引用的输入字段无法转换成所需数据类型，则无效。
3. 如果 ControlToCompare 引用的输入字段无法转换成所需数据类型，则有效。
4. 输入字段转换成所需数据类型并进行比较。

第三步看起来有些不符合直觉。之所以这样评估，是因为如果验证器同时检查多个字段的有效性，很难为该验证器写出有意义的错误信息。应使用一个独立的验证器来报告 ControlToCompare 输入字段中的错误情况。RangeValidator 的工作方式类似，具有 maximum 和 minimum 属性。

5.8 Enabled、Visible 和 Display 属性的作用

验证器的 Enabled、Visible 和 Display 属性之间的区别可能不是非常明显。

Display=None 可以用来指定验证器不直接显示任何内容，但是仍然进行评估，仍然影响总体的有效性，并且仍可以将错误放在客户机和服务器上的摘要中。对于客户端验证，这些值确定使用可见性样式特性还是使用显示样式特性来打开或关闭验证器。对于服务器端验证，Display=Dynamic 表示输入有效时不显示任何内容，而 Display=Static 表示显示一个不换行的空格 (" ")。使用最后一个设置是为了表中只包含验证器的单元格在有效时，不会折叠成不显示任何内容。

为什么不只使用 Visible=false 使验证器不可见呢？在 ASP.NET 中，控件的 Visible 属性有许多含义：Visible=false 的控件根本不会被处理来预显示或显示。正是因为这种含义，验证器的 Visible=false 意味着不仅不会显示任何内容，而且无法使用。不会对这样的验证器进行评估，不会影响页面的有效性，也不会将错误放在摘要中。

Enabled 则为中性。对于大多数情况，Enabled=false 与 Visible=false 的结果完全相同。在 Beta 1 版或更高版本中，存在一个重要的区别：在客户端验证中，禁用的验证器仍会发送到浏览器中，但是处于禁用状态。可以使用客户端脚本中的 ValidatorEnable 函数激活该验证器。

使用 Visible 或 Enabled 控制是否进行验证时，应注意上述服务器上的事件顺序，或者在验证之前进行更改，或者在更改之后重新验证。否则，它们的 IsValid 值不会将更改反映到属性上。

5.9 CustomValidator 控件

扩展验证框架最简单的方法是使用 CustomValidator 控件，该控件既可以用来执行其它验证控件无法进行的验证，也可以执行需要访问服务器上信息（例如数据库或 Web 服务）的验证。

如果添加了只定义一个服务器验证函数的 CustomValidator，该验证器并不参与客户端验证。当用户使用 tab 键在各字段之间切换时，CustomValidator 不会更新，并且需要往返服务器一次以执行其验证。如果要使用 CustomValidator 执行不需要任何服务器上信息的检查，也可以使用 ClientValidationFunction 属性让验证器完全参与客户端验证。假设提供了一个 ClientValidationFunction，理想情况下，应与服务器验证处理程序执行完全相同的检查。但实际上，只是执行该验证的一部分。客户端验证函数进行的验证不要超过在服务器上执行的验证，因为黑客很容易绕过该验证函数。

以下是在客户机和服务器上使用 CustomValidator 的一个简单示例，只检查输入是否是偶数。以下先介绍服务器函数（在 C# 中）：

```
public bool ServerValidation(object source, string value) {
    try {
        int i = int.FromString(value);
        return ((i % 2) == 0);
    } catch {
        return false;
    }
}
```

以下是该函数在客户机上的声明方法，以及一个执行相同检查的客户端验证函数。这通常是 JScript 形式，不过如果你的目标是 Microsoft Internet Explorer，也可以使用 VBScript 形式。

```
<asp:CustomValidator id="customVal2" runat=server
    ErrorMessage="数字不可以被 2 除！"
    ControlToValidate="txtCustomData"
    OnServerValidationFunction=ServerValidation
    ClientValidationFunction="CheckEven" /><br>
Data Field : <asp:TextBox id="txtCustomData" runat="server" />
<script language=javascript>
<!--
function CheckEven(source, value) {
    var val = parseInt(value, 10);
    if (isNaN(val))
        return false;
    return ((val % 2) == 0);
}
```



```
// -->
</script>
```

以下是使用 CustomValidator 的一些注意事项：

- ☞ 与所有其它验证控件类似（RequiredFieldValidator 除外），如果输入字段为空，则认为 CustomValidator 有效。
- ☞ 如果使用较旧的浏览器，或者关闭了客户端验证，将无法调用客户端验证函数。在定义该函数之前，不必检查所用浏览器的功能，但是需要确保浏览器不会因为定义而造成脚本错误。
- ☞ 两个参数传递到客户端函数中，与传递给服务器函数的参数对应。第一个是客户端验证器元素，第二个是 ControlToValidate 指定的控件值。不过，在客户机上，可以选择不为函数定义参数，这样也会正常工作。
- ☞ 如果使用 Beta1 版或更高版本，可以保留 ControlToValidate 为空。在该模式中，服务器函数每次往返总会触发一次，客户端函数每次尝试提交时总会触发一次。可以使用该特性来验证其它方法无法验证的控件，例如 CheckBoxList 或单独的单选按钮。如果条件是基于多个控件，并且不希望用户使用 tab 键在页面上各字段之间切换时评估该条件，可以使用该方法。
- ☞ Beta 1 版或更高版本中的另一个选项是挂接多个控件的 change 事件。方法是加入一些调用客户端函数 ValidatorHookupControl 的内嵌脚本，如上所述。

5.10 哪些控件可以被验证

要使控件可以被验证控件引用，该控件必须具有验证属性，所有可以验证的控件均具有 ValidationPropertyAttribute 属性，该属性指明验证时应读取的属性。如果编写自己的控件，可以通过提供其中一个特性来指定要使用的属性，从而使该控件参与验证。

要使验证可以在客户端正常进行，该属性必须与客户端显示的 HTML 元素的 value 特性对应。许多复杂的控件（例如 DataGrid 和 Calendar）在客户端没有值，只能在服务器上进行验证。因此，只有最接近 HTML 元素的控件才可以参与验证。此外，控件必须在客户端具有单个逻辑值。因此，RadioButtonList 可以被验证，但是 CheckBoxList 不可以。

5.11 详细的用户输入验证

5.11.1 简介

对用户输入进行验证是基于 Web 的应用程序中常见的情况。为了开发应用程序，开发人员经常在这项工作中花费大量的时间和编辑大量的代码，比希望的多得多。在 ASP.NET 页面架构的构建过程中，努力使输入验证这一任务比以前更容易是很重要的。

5.11.2 什么是验证器

为了有效地使用验证器，给出其严格的定义是很有帮助的，提炼一下以前给出的定义：

“验证是一个控件，它在一个输入控件中检查某种特定类型的错误条件，并显示该问题的说明。”

这是一个重要的定义，因为它意味着经常需要为每个输入控件使用多个验证器控件。

例如，如果要检查文本框中的用户输入是否为 (a) 非空，(b) 在特定范围中的合法日期，或 (c) 小于在另一文本输入控件中的日期，那么可能要使用三个验证器。这也许显得有些繁琐，但请记住这对用户是有帮助的，对所有这些情况可能要使用三种不同的文本说明。

下面列出几种不同的验证器（表 5-5）。

表 5-5

RequiredFieldValidator	检查用户是否输入或选择了任何内容
RegularExpressionValidator	根据规则表达式检查用户输入。该过程允许进行许多种类的检查，可以用于邮政编码和电话号码等的检查。
CompareValidator	将输入控件与一个固定值或另一个输入控件进行比较。例如，它可以用在口令验证字段中。也可以用来比较输入的日期和数字。
RangeValidator	与 CompareValidator 非常相似，只是它用来检查输入是否在两个值或其它输入控件的值之间。
CustomValidator	允许用户编写自己的代码以加入到验证框架中。

5.11.3 验证器演示

为了示范验证任务，下面概要介绍向现有页面添加验证的过程，以下是一些必需条件的示例：

编写一个 Web 页面来收集新的用户 id 和口令。用户 id 必须包含 6-10 个字母字符，且必须是未使用的。口令必须包括 4-12 个字母，其中至少有一个数字和一个“@#%&^&*’”中的字符。用户必须再次输入口令以确保输入无误。

从一个 HTML 页面开始，该页面已做了最小程度的转换以适用于 ASP+ 页面框架。转换页面的过程包括下述步骤：

1. 将扩展后缀从“.html”或“.asp”转换为“.aspx”。
2. 将窗体模板和所有输入标签改为“runat=server”。
3. 使用“ID”而不是“name”。

起始代码：

```
<html>
<head>
```

```
<title>验证示例;/title>
</head>
<body>

<p>请输入新的用户 ID 和口令</p>
<form runat=server>
<table>
  <tr>
    <td>用户 ID </td>
    <td><input type=text runat=server id=txtName></td>
  </tr>
  <tr>
    <td>口令 </td>
    <td><input type=password runat=server id=txtPWord></td>
  </tr>
  <tr>
    <td>请重新输入口令 </td>
    <td><input type=password runat=server id=txtRePWord></td>
  </tr>
</table><br>
<input type=submit runat=server id=cmdSubmit value=Submit>
</form>

</body>
</html>
```

起始页如图 5-1 所示。



请输入新的用户 ID 和口令	
用户 ID	<input type="text" value="dumpling"/>
口令	<input type="password" value="*****"/>
请重新输入口令	<input type="password"/>
<input type="submit" value="提交"/>	

图 5-1

5.11.4 并非自愿

首先要强制使所有字段都已填充。

在每一个字段前添加 RequiredFieldValidator。若输入字段为空，要在字段前显示一个星号(*)，并在摘要区域报告文本错误。以下是向 User ID 字段添加 RequiredFieldValidator 控件的方法：

```
<tr>
  <td>
    <asp:RequiredFieldValidator runat=server
      ControlToValidate=txtName
      ErrorMessage="需要用户 ID。"> *
    </asp:RequiredFieldValidator>
  </td>
  <td>User ID:</td>
  <td><input type=text runat=server id=txtName></td>
</tr>
```

若输入为空，则在标签旁边显示 *。出错消息在摘要中报告。“ControlToValidate”属性指定了需要验证的控件 ID。最后一步是向页面顶部添加验证总结，如下所示：

```
<asp:ValidationSummary runat=server
  headerText=此页有错误： />
```

该页面如图 5-2 所示。

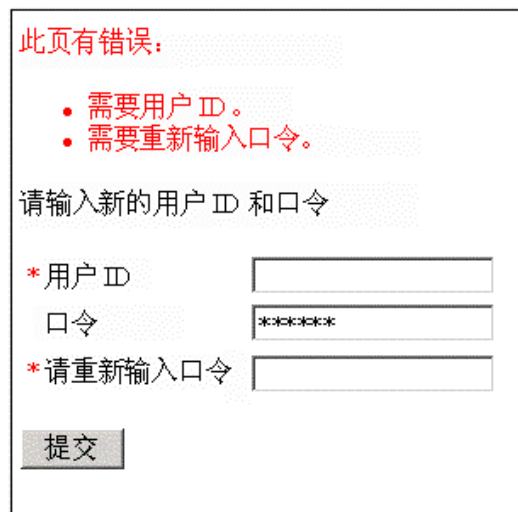


图 5-2

5.11.5 规则化

下一步我们需要强调用户 ID 和口令字段的要求。对这些字段，要使用 RegularExpressionValidator 控件。规则表达式在此类信息的简明表达检查中非常有效，这些信息还包括邮政编码、电话号码和电子邮件地址。

下面显示了如何指定对用户 ID 的限制：

```
<td>
  <input type=text runat=server id=txtName>
  <asp:RegularExpressionValidator runat=server
    ControlToValidate="txtName"
    ErrorMessage="口令必须由 6-10 个字母组成。"
    ValidationExpression="[a-zA-Z]{6,10}" />
</td>
```

注意在本示例中，并没有指定标记内的任何内部内容。内部文本等同于控件的“文本属性”。如果它为空白，出错信息将显示在控件所在的位置，同样它也出现在摘要中。

可以用下面两个验证器来检查口令。如果你使用了多个验证器，则只有它们全都匹配时，才可以认为输入是有效的。

```
<asp:RegularExpressionValidator runat=server display=dynamic
  ControlToValidate="txtPWord"
  ErrorMessage="口令必须包含 @#$%^&* / 中的一个。"
  ValidationExpression=".*[@#$%^&* /].*" />
<asp:RegularExpressionValidator runat=server display=dynamic
  ControlToValidate="txtPWord"
  ErrorMessage="口令必须是 4-12 个非空白字母。"
  ValidationExpression="[\\S]{4,12}" />
```

这是操作中带有表达式的窗体，如图 5-3 所示。

此页有错误：

ID 必须由 6-10 个字母组成。
口令必须包含 @#\$%^&* / 中的一个。
需要重新输入口令。

请输入用户 ID 及口令：

用户 ID	<input type="text" value="this that"/>	ID 必须由 6-10 个字母组成。
口令	<input type="password" value="****"/>	口令必须包含 @#\$%^&* / 中的一个。
*请重新输入口令	<input type="password"/>	

图 5-3

5.11.6 逐一比较

需要确认口令的重新输入字段与口令匹配。需要使用 CompareValidator 控件完成此操

作。因为它能够同时处理两个输入控件。

```
<asp:CompareValidator runat=server
    ControlToValidate=txtRePWord
    ControlToCompare=txtPWord
    ErrorMessage="口令不匹配。" />
```

默认情况下，CompareValidator 只做简单的字符串匹配比较。如果需要，它可进行涉及日期和数字的更复杂的比较。

5.11.7 符合习惯

需要检查的最后一件事是名称尚未在假想的站点中使用，需继续在服务器上查找一些数据。为了模拟此操作，在服务器端的代码段中创建一个哑函数来检查第一个字符是否是“a”。下面的 C Sharp 函数在页面上定义：

```
public bool CheckID(Object source, string value){
    return value.Substring(0, 1).ToLower() != "a";
}
```

要调用这一函数，添加一个 CustomValidator，它用于调用开发者的代码来执行检查。以下是它的声明：

```
<asp:CustomValidator runat=server
    controltovalidate="txtName"
    errormessage="ID 已经在使用。"
    OnServerValidationFunction="CheckID" />
```

它也可以调用客户机脚本中的定制函数，尽管如果它必须在数据库中查找值时这会不太实用。在有脚本的客户机上，在其它所有验证器先在客户机上执行检查后才允许提交。如果只在服务器上进行的检查检测到这样的错误，则会向用户返回页面指出这些错误。

5.11.8 尾声

现在，唯一剩下的事就是使用经过精细验证后的数据了，需要做的只是检查页面的 IsValid 属性，判定是否可以进行数据库的更新。提交的处理程序可能类似下面的形式：

```
public OnSubmit(Object source, EventArgs e){
    if (Page.IsValid){
        //现在我们可以执行事务;
    }
}
```

正如所看到的那样，该处理程序允许数据输入页包含特定于应用程序的代码，而不是通篇代码都来处理一般的输入检查。

图 5-4 是关于最终活动页面的其它一些信息。

此页有错误:

ID 已经在使用。
口令不匹配。

请输入用户 ID 及口令:

用户 ID ID 已经在使用。

口令

请重新输入口令 口令不匹配。

图 5-4

第 6 章 服务器端控件

6.1 文本输入控件

文本输入控件目的是让用户输入文本，文本模式是一个单行的输入框，但是用户可以根据自己的需要把它改成密码输入模式或者多行输入模式。

在此我们用单行文本输入模式和密码模式来说明，在 ASP.NET 中，输入一个文本，可用下面的语句来表示：

```
<!--输入邮件地址-->
```

```
<asp:TextBox id=email width=200px maxlength=60 runat=server />
```

第一句为注释，我们可以设定输入框的宽度和文本的长度，runat=server 为表示运行于服务器端。下面我们来看看输入控件的校验，一个简单的语句就可以实现在普通的 html 页面上的复杂的 JavaScript、VBScript 或者其他代码的验证。首先用户必须输入邮件地址：

```
<!--验证邮件的有效性！->不能为空-->
```

```
<asp:RequiredFieldValidator id="emailReqVal"
    ControlToValidate="email"
    ErrorMessage="Email. "
    Display="Dynamic"
    Font-Name="Verdana" Font-Size="12"
    runat=server>
    *
</asp:RequiredFieldValidator>
```

ControlToValidate="email"属性为针对 TextBox id=email 的文本框，Display 属性定义为“Dynamic”，即为当鼠标光标所在位置发生变化时属性根据条件变化。如果为空，则打印出“ * ”字符出来。

在通常情况下，邮件地址总会包含一些特定的字符，在验证的时候，就可以要求用户输入的内容中包含规定的字符。

```
<!--验证邮件的有效性！->必须包含有效字符-->
```

```
<asp:RegularExpressionValidator id="emailRegexVal"
    ControlToValidate="email"
    Display="Static"
    ValidationExpression="^[\\w-]+@[\\w-]+\\. (com|net|org|edu|mil)$"
    Font-Name="Arial" Font-Size="11"
    runat=server>
    不是有效邮件地址
</asp:RegularExpressionValidator>
```

ControlToValidate="email"语句跟上面一样，ValidationExpression="^[\\w-]+@[\\w-]+\\.

(com|net|org|edu|mil)\$"表示我们在邮件里要包含的内容，如果没有包含，则打印出“不是有效邮件地址”这个提示。

对密码也是同样的道理的，主要的差别是，对于密码，通常我们要确认一次，校验两次输入的密码是否相等。下面是我们的代码：

```
<!--输入确认密码-->两个密码是否相等-->
    <asp:CompareValidator id="CompareValidator1"
        ControlToValidate="passwd2" ControlToCompare="passwd"
        Display="Static"
        Font-Name="Arial" Font-Size="11"
        runat=server>
        密码不相等
    </asp:CompareValidator>
```

ControlToValidate="passwd2" ControlToCompare="passwd"此语句即为两个密码之间的比较，不相等，打印出“密码不相等”的提示。

下面是我们的完整的代码（csbook\form\all.aspx）：

```
<html>
<body>
<br><br><br>
    <center>
        <h3><font face="Verdana">.NET->文本控件</font></h3>
    </center>
    <form method=post runat=server>
    <hr width=600 size=1 noshade>
<br><br>
    <center>
<!--标题-->
    <asp:ValidationSummary ID="valSum" runat="server"
        HeaderText="按照下面的要求填写："
        DisplayMode="SingleParagraph"
        Font-Name="verdana"
        Font-Size="12"
    />
    <p>
    <table border=0 width=600>
    <tr>
        <td align=right>
            <font face=Arial size=2>电子邮件:</font>
        </td>
        <td>
<!--输入邮件地址-->
            <asp:TextBox id=email width=200px maxlength=60 runat=server />
        </td>
```

```
<td>
<!--验证邮件的有效性!-->不能为空-->
    <asp:RequiredFieldValidator id="emailReqVal"
        ControlToValidate="email"
        ErrorMessage="Email. "
        Display="Dynamic"
        Font-Name="Verdana" Font-Size="12"
        runat=server>
        *
    </asp:RequiredFieldValidator>
<!--验证邮件的有效性!-->必须包含有效字符-->
    <asp:RegularExpressionValidator id="emailRegexVal"
        ControlToValidate="email"
        Display="Static"
        ValidationExpression="^[\\w-]+@[\\w-]+\\. (com|net|org|edu|mil)$"
        Font-Name="Arial" Font-Size="11"
        runat=server>
        不是有效邮件地址
    </asp:RegularExpressionValidator>
</td>
</tr>

<tr>
    <td align=right>
        <font face=Arial size=2>密码:</font>
    </td>
    <td>
<!--输入密码-->
        <asp:TextBox id=passwd TextMode="Password" maxlength=20
runat=server/>
    </td>
    <td>
<!--输入密码-->密码不能为空-->
        <asp:RequiredFieldValidator id="passwdReqVal"
            ControlToValidate="passwd"
            ErrorMessage="Password. "
            Display="Dynamic"
            Font-Name="Verdana" Font-Size="12"
            runat=server>
            *
        </asp:RequiredFieldValidator>
<!--输入密码-->包含其中有效字符-->
        <asp:RegularExpressionValidator id="passwdRegexBal"
            ControlToValidate="passwd"
```

```

        ValidationExpression=".*[!@#%&*+;:].*"
        Display="Static"
        Font-Name="Arial" Font-Size="11"
        Width="100%" runat=server>
        密码必须包含 (!@#%&*+;:)
    </asp:RegularExpressionValidator>
</td>
</tr>
<tr>
    <td align=right>
        <font face=Arial size=2>再次输入密码:</font>
    </td>
    <td>
        <!--输入确认密码-->
        <asp:TextBox id=passwd2 TextMode="Password" maxlength=20
runat=server/>
    </td>
    <td>
        <!--输入确认密码-->不能为空-->
        <asp:RequiredFieldValidator id="passwd2ReqVal"
            ControlToValidate="passwd2"
            ErrorMessage="Re-enter Password. "
            Display="Dynamic"
            Font-Name="Verdana" Font-Size="12"
            runat=server>
            *
        </asp:RequiredFieldValidator>
        <!--输入确认密码-->两个密码是否相等-->
        <asp:CompareValidator id="CompareValidator1"
            ControlToValidate="passwd2" ControlToCompare="passwd"
            Display="Static"
            Font-Name="Arial" Font-Size="11"
            runat=server>
            密码不相等
        </asp:CompareValidator>
    </td>
</tr>
</table>
<p>
    <input runat="server" type=submit value="提交">
<p>
    <hr width=600 size=1 noshade>
</form>

```

```
</center>  
</body>  
</html>
```

运行结果如图 6-1 所示。

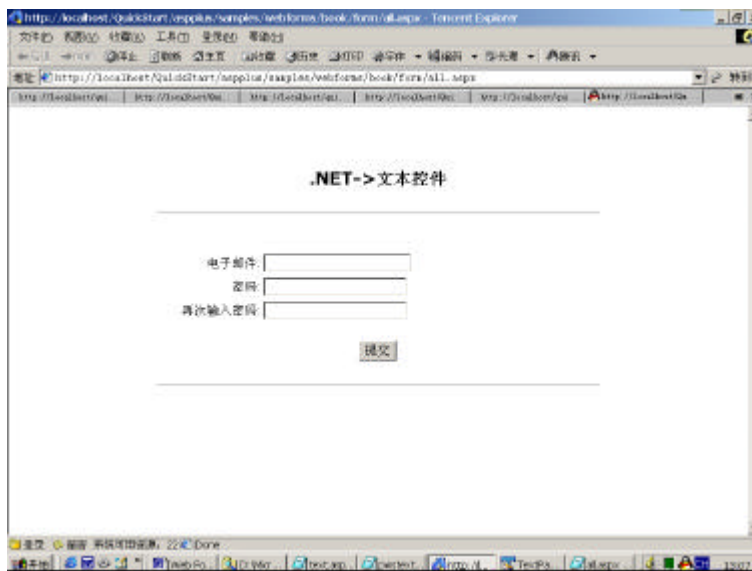


图 6-1

我们不按照要求的输入，得到了图 6-2 的提示。

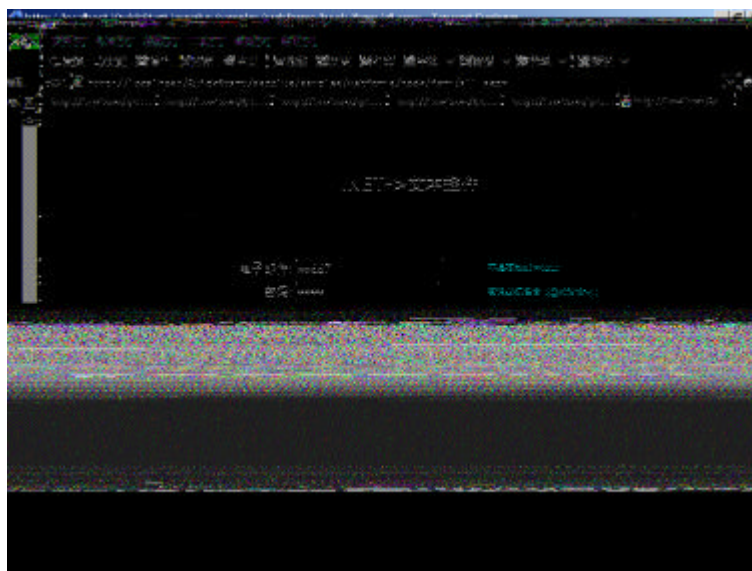


图 6-2

多行文本输入控件一般用来输入相关的内容，比如用户简短介绍、相关信息的补充等，一般情况下可以不用限制用户的输入。当然有些时候像留言板，我们不希望用户的输入内容中包含 HTML 的相关标记，这个时候就可以用上面的同样的方法来限制用户的输入，用法都是一样的，在此就不打算举例说明了。

6.2 选择控件

选择的方式有两种：单选、多选。下面用具体的例子来说明这两种选择控件在 ASP.NET 上的实现。

对单选控件，ASP.NET 里面有一个专用的表示：RadioButtonList，看下面的代码：

```
<!--列出选择内容-->
<ASP:RadioButtonList id=ccType Font-Name="Arial" RepeatLayout="Flow"
runat=server>
<asp:ListItem>招行一卡通</asp:ListItem>
    <asp:ListItem>牡丹卡</asp:ListItem>
</ASP:RadioButtonList>
```

在取值的时候，就可以用一个语句：

```
Request.QueryString("ccType")
```

来取得这个值。下面是说明的完整代码：

```
( csbook\form\select.aspx )

<html>
<body bgcolor="#ccccff">
<center>
<br><br>
    <h3><font face="Verdana">.NET->选择控件!</font></h3>
<br><br><br>
</center>
    <form method=post runat=server>
    <hr width=600 size=1 noshade>

    <center>
    <asp:ValidationSummary ID="valSum" runat="server"
        HeaderText="你必须填写下面的内容："
        DisplayMode="SingleParagraph"
        Font-Name="verdana"
        Font-Size="12"
    />
    <p>
    <!-- 信用卡信息 -->
```

```
<table border=0 width=600>
<tr>
  <td colspan=3>
    <center>
      <font face=Arial size=2><b>信用卡信息</b></font>
    </center>
  </td>
</tr>
<tr>
  <td align=right>
    <font face=Arial size=2>信用卡类型:</font>
  </td>
  <td>
    <!--列出选择内容-->
    <ASP:RadioButtonList id=ccType Font-Name="Arial"
RepeatLayout="Flow" runat=server>
      <asp:ListItem>招行一卡通</asp:ListItem>
      <asp:ListItem>牡丹卡</asp:ListItem>
    </ASP:RadioButtonList>
  </td>
  <td>
    <asp:RequiredFieldValidator id="ccTypeReqVal"
      ControlToValidate="ccType"
      ErrorMessage="信用卡类型. "
      Display="Static"
      InitialValue=""
      Font-Name="Verdana" Font-Size="12"
      runat=server>
      *
    </asp:RequiredFieldValidator>
  </td>
</tr>
</table>
<p>
<input runat="server" type=submit value="提交">
<p>
<hr width=600 size=1 noshade>
</form>
</center>
</body>
</html>
```

运行结果如图 6-3。



图 6-3

选择一个并提交，则会提交成功；反之，如果没有选择就提交，会出现如图 6-4 的信息：



图 6-4

再来看看多选的情况：

选择象列表

```
<asp:CheckBoxList id=Check1 runat="server">
  <asp:ListItem>北京</asp:ListItem>
  <asp:ListItem>深圳</asp:ListItem>
  <asp:ListItem>上海</asp:ListItem>
  <asp:ListItem>广州</asp:ListItem>
  <asp:ListItem>南宁</asp:ListItem>
  <asp:ListItem>重庆</asp:ListItem>
</asp:CheckBoxList>
```

跟上面的单选控件就相差在定义上，用 `CheckBoxList` 来描述我们的选择框，写一个方法来响应我们的“提交”事件：

```
    '响应按钮事件
Sub Button1_Click(sender As Object, e As EventArgs)
    Dim s As String = "被选择的选项是:<br>"
    Dim i As Int32
    For i = 0 to Check1.Items.Count-1
        If Check1.Items(i).Selected Then
            '列出选择项
            s = s & Check1.Items(i).Text
            s = s & "<br>"
        End If
    Next
    '打印出选择项
    Label1.Text = s
End Sub
```

这个方法为定义打印被选择的选项。具体的内容如下：

(`csbook\form\select01.aspx`)

```
<html>
<head>
<script language="VB" runat="server">
    '响应按钮事件
    Sub Button1_Click(sender As Object, e As EventArgs)
        Dim s As String = "被选择的选项是:<br>"
        Dim i As Int32
        For i = 0 to Check1.Items.Count-1
            If Check1.Items(i).Selected Then
                '列出选择项
                s = s & Check1.Items(i).Text
                s = s & "<br>"
            End If
        Next
        '打印出选择项
        Label1.Text = s
    End Sub
</script>
</head>
<body bgcolor="#ccccff">
<br><br><br>
```



```
<center>
  <h3><font face="Verdana">.NET->CheckBoxList</font></h3>
</center>
<br><br>
<center>
  <form runat=server>
    '选择象列表
    <asp:CheckBoxList id=Check1 runat="server">
      <asp:ListItem>北京</asp:ListItem>
      <asp:ListItem>深圳</asp:ListItem>
      <asp:ListItem>上海</asp:ListItem>
      <asp:ListItem>广州</asp:ListItem>
      <asp:ListItem>南宁</asp:ListItem>
      <asp:ListItem>重庆</asp:ListItem>
    </asp:CheckBoxList>
    <p>
      <asp:Button id=Button1 Text="提交" onclick="Button1_Click"
runat="server" />
    <p>
      <asp:Label id=Label1 font-name="Verdana" font-size="8pt"
runat="server" />
    </form>
  </center>
</body>
</html>
```

结果如图 6-5 所示。



图 6-5

选择几个选项，并点击“提交”按钮，看到如图 6-6 的情况。



图 6-6

6.3 列表控件

在 ASP.NET 中，有几种方法可以应用于列表控件。可以在 aspx 代码中直接嵌入相关的代码，也可以在页面装入的时候加载这些列表信息。

下面是具体的应用，先看看在 aspx 中的列表方法：

```
<!--列表-->列出内容-->
```

```
<asp:DropDownList id=DropDown1 runat="server">
  <asp:ListItem>北京</asp:ListItem>
  <asp:ListItem>深圳</asp:ListItem>
  <asp:ListItem>上海</asp:ListItem>
  <asp:ListItem>广州</asp:ListItem>
  <asp:ListItem>南宁</asp:ListItem>
  <asp:ListItem>重庆</asp:ListItem>
</asp:DropDownList>
```

在需要取出所选的数据时，直接去取 id 值，即 DropDown1；再定义一个方法，响应“提交”按钮的事件就可以了。下面是完整的代码：

```
( csbook\form\list\list.aspx )
```

```
<html>
<head>
```

```
<script language="VB" runat="server">
```

```
'在点击按钮时候响应
    Sub list_Click(sender As Object, e As EventArgs)
        Labell1.Text="你的选择是: " + DropDownList1.SelectedItem.Text
    End Sub

</script>

</head>
<body bgcolor="#ccccff">
<br><br><br>
<center>
    <h3><font face="Verdana">.NET->列表控件</font></h3>
</center>
<br><br>
<center>
    <form runat=server>
    <!--列表->列出内容-->
        <asp:DropDownList id=DropDown1 runat="server">
            <asp:ListItem>北京</asp:ListItem>
            <asp:ListItem>深圳</asp:ListItem>
            <asp:ListItem>上海</asp:ListItem>
            <asp:ListItem>广州</asp:ListItem>
            <asp:ListItem>南宁</asp:ListItem>
            <asp:ListItem>重庆</asp:ListItem>
        </asp:DropDownList>

        <asp:button text="提交" OnClick="list_Click" runat=server/>

    <p>

        <asp:Label id=Labell1 font-name="Verdana" font-size="10pt"
runat="server">

        </asp:Label>

    </form>
</center>
</body>
</html>
```

我们看看运行结果，如图 6-7。

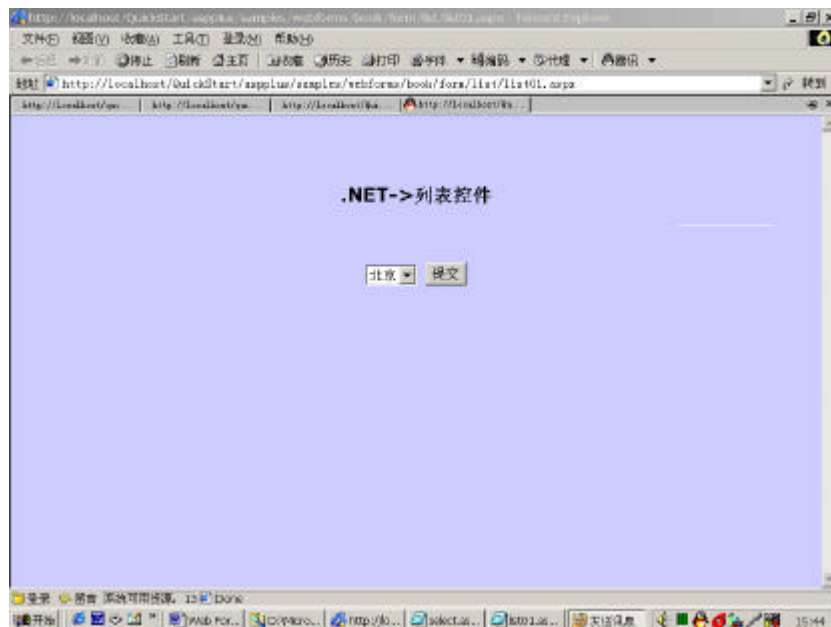


图 6-7

点击提交按钮时候看到图 6-8 的结果。

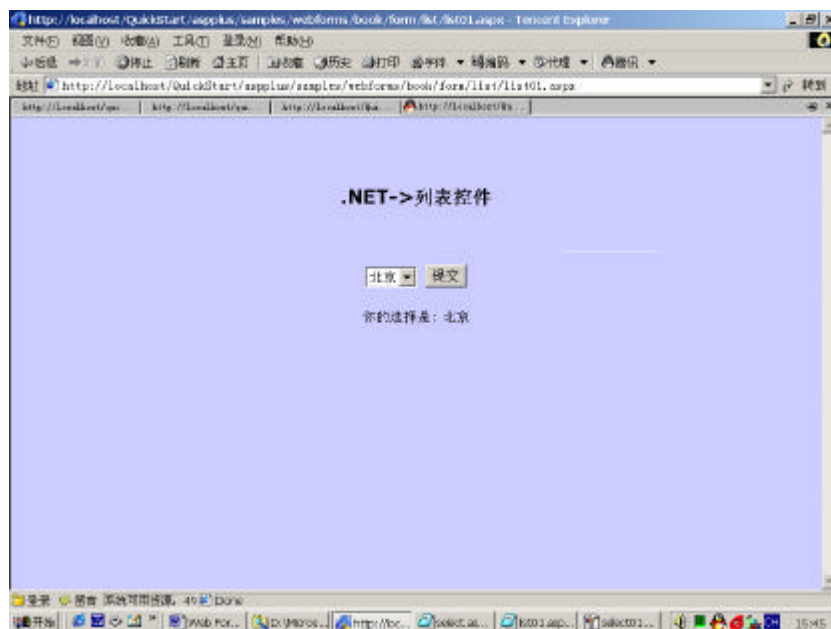


图 6-8

下面再来看看另外一个列表控件的使用，定义一个在页面装载的时候调用的方法：

```
Sub Page_Load(sender As Object, e As EventArgs)
    If Not IsPostBack Then
        Dim values as ArrayList= new ArrayList()
        values.Add ("北京")
        values.Add ("深圳")
        values.Add ("上海")
        values.Add ("广州")
        values.Add ("南宁")
        values.Add ("重庆")
        '设定DropDown1的数据源为values，即上面定义的信息
        DropDownList.DataSource = values
        '数据的绑定
        DropDownList.DataBind
    End If
End Sub
```

在 aspx 代码中调用它：

```
<!--列出列表信息-->
```

```
<asp:DropDownList id="DropDown1" runat="server" />
```

就这样的一个简单的语句就可以了，下面是这个文件的完整的代码：

```
( csbook\form\list\list01.aspx )
```

```
<html>
```

```
<head>
```

```
<script language="VB" runat="server">
```

```
'在页面装载的时候调用的方法：
```

```
Sub Page_Load(sender As Object, e As EventArgs)

    If Not IsPostBack Then
        Dim values as ArrayList= new ArrayList()
        values.Add ("北京")
        values.Add ("深圳")
        values.Add ("上海")
        values.Add ("广州")
        values.Add ("南宁")
        values.Add ("重庆")
        '设定DropDown1的数据源为values，即上面定义的信息
        DropDownList.DataSource = values
        '数据的绑定
        DropDownList.DataBind
    End If
End Sub
```

```
        End If
    End Sub

    '提交按钮响应的方法
    Sub select02_Click(sender As Object, e As EventArgs)
        Label1.Text = "你的选择是: " + DropDownList1.SelectedItem.Text
    End Sub

</script>

</head>
<body BGCOLOR="#CCCCCC">

<br><br><br>
<center>
    <h3><font face="Verdana">.NET->列表控件</font></h3>
</center>
<br><br>
<center>
    <form runat=server>

        <!--列出列表信息-->
        <asp:DropDownList id="DropDown1" runat="server" />

        <asp:button Text="提交" OnClick="select02_Click" runat=server/>
        <p>
            <asp:Label id=Label1 font-name="Verdana" font-size="10pt"
runat="server" />

        </form>
    </center>
</body>
</html>
运行的结果跟上面的一样。
```

6.4 重复列表 Repeater

这种服务器控件会以给定的形式重复显示数据项目，故称之为重复列表。使用重复列表有两个要素，即数据的来源和数据的表现形式。数据来源的指定由控件的 DataSource 属性决定，并调用方法 DataBind 绑定到控件上。这里需要说明的是数据取出以后如何表现的问题，即如何布局。重复列表的数据布局是由给定的模板来决定的，由于重复列表没有缺

省的模板，所以使用重复列表时至少要定义一个最基本的模板“ItemTemplate”。

重复列表支持以下模板标识，所谓模板就是预先定义的一种表现形式，以后还会就这个问题专门讨论，这里就不在多说。

- 1) ItemTemplate 模板，数据项模板，必需的，它定义了数据项极其表现形式。
- 2) AlternatingItemTemplate 模板，数据项替换模板，跟数据项模板基本一致，只是它扫描数据项内容，发现相符的项，则以数据项替换模板去代替原先定义，再显示出来。
- 3) SeparatorTemplate 模板，分割符模板，定义数据项之间的分割符。
- 4) HeaderTemplate 模板，报头定义模板，定义重复列表的表头表现形式。
- 5) FooterTemplate 模板，表尾定义模板，定义重复列表的列表尾部的表现形式。

切记，由于缺乏内置的预定义模板和风格，在使用重复列表时，请一定记住要使用 HTML 格式定义自己的模板。

下面给出一个例子，看它是如何使用重复列表控件的。下面的例子首先在页面加载过程时把数据装载，并绑定到两个重复列表上；然后以一个 2 列的表格显示；最后把所有数据显示到一行上面，并且国家和领导人之间以 3 个中横线分隔，每一国家之间以竖划线分隔。

1. 源代码

```
<!-- 文件名：csbook\form\serverctl\FormRepeater.aspx -->
<html>
<head>
  <script language="vb" runat=server>

      Class Leader
        '定义一个类Leader
        dim strCountry as String
        dim strName as String

        Public Sub New(country As String, name As String)
          MyBase.New
          strName = name
          strCountry= country
        End Sub

        ReadOnly Property Name As String
          Get
            Return strName
          End Get
        End Property
```

```
        ReadOnly Property Country As String
            Get
                Return strCountry
            End Get
        End Property

    End Class

    sub Page_Load(s as object,e as eventargs)
        dim leaders as ArrayList = New ArrayList()
        if Not Page.IsPostBack
            '加载数据
            leaders.add(new leader("美利坚","布 什"))
            leaders.add(new leader("俄罗斯","普 京"))
            leaders.add(new leader("中 国","江泽民"))

            Repeater1.DataSource=leaders
            Repeater2.DataSource=leaders
            Repeater1.DataBind
            Repeater2.DataBind
        end if
    end sub
</script>
<title>
    重复列表使用例子
</title>
</head>

<body bgcolor=#ccccff>
    <center>
        <h2>重复列表的使用</h2>
        <hr>
        <br>
        '以表格形式显示国家，领导人信息
        <asp:Repeater id="Repeater1" runat=server>
            '定义表头
            <template name=HeaderTemplate>
                <table border=2>
                    <tr>
                        <th>
                            国家名
                        </th>
```



```
        <th>
            领导人
        </th>
    </tr>
</template>

'定义数据显示格式
<template name=ItemTemplate>
    <tr>
        <td>
            <%=# DataBinder.Eval(Container.DataItem,"Country") %>
        </td>
        <td>
            <%=# DataBinder.Eval(Container.DataItem,"Name") %>
        </td>
    </tr>
</template>

'定义表尾
<template name=FooterTemplate>
    <tr>
        <td>日期</td>
        <td>2001年</td>
    </tr>
</table>
</template>
</asp:Repeater>

<br>
<asp:Repeater id=Repeater2 runat=server>
    '国家和领导人以|分割显示
    <template name=ItemTemplate>
        <%=# DataBinder.Eval(Container.DataItem,"Country") %>
        ---
        <%=# DataBinder.Eval(Container.DataItem,"Name") %>
    </template>

    <template name=SeparatorTemplate>
        |
    </template>
</asp:Repeater>
</center>
```

```
</body>
```

```
<html>
```

2. 输出结果如图 6-9。



图 6-9

6.5 数据列表 DataList

数据列表显示跟重复列表 (Repeater) 比较类似, 但是它可以选择和修改数据项的内容。数据列表的数据显示和布局也如同重复列表都是通过“模板”来控制的。同样的, 模板至少要定义一个“数据项模板”(ItemTemplate) 来指定显示布局。数据列表支持的模板类型更多, 它们如下:

- 1) ItemTemplate 模板, 数据项模板, 必需的, 它定义了数据项极其表现形式。
 - 2) AlternatingItemTemplate 模板, 数据项替换模板, 跟数据项模板基本一致, 只是它扫描数据项内容, 发现相符的项, 则以数据项替换模板去代替原先定义, 再显示出来。
 - 3) SeparatorTemplate 模板, 分割符模板, 定义数据项之间的分割符。
 - 4) SelectedItemTemplate 模板, 选中项模板, 定义被选择的数据项的表现内容与布局形式, 当未定义“SelectedItemTemplate”模板时, 选中项的表现内容与形式无特殊化, 由 ItemTemplate 模板定义所决定。
 - 5) EditItemTemplate 模板, 修改选项模板, 定义即将被修改的数据项的显示内容与布局形式, 缺省情况下, 修改选项模板就是数据项模板 (ItemTemplate) 的定义。
 - 6) HeaderTemplate 模板, 报头定义模板, 定义重复列表的表头表现形式。
 - 7) FooterTemplate 模板, 表尾定义模板, 定义重复列表的列表尾部的表现形式。
- 数据列表还可以通过风格形式来定义模板的字体、颜色、边框。每一种模板都有它自

己的风格属性。例如，可以通过设置修改选项模板的风格属性来指定它的风格。

此外，还有一些其他属性可以导致数据列表的显示有较大的改变，下面侧重说明。

RepeatLayout：显示布局格式，指定是否以表格形式显示内容。

RepeatLayout.Table 指定布局以表格形式显示。

RepeatLayout.Flow 指定布局以流格式显示，即不加边框。

RepeatDirection：显示方向，指定显示是横向显示还是纵向显示

RepeatDirection.Horizontal 指定是横向显示

RepeatDirection.Vertical 指定是纵向显示

RepeatColumns：一行显示列数，指定一行可以显示的列数，缺省情况下，系统设置为一行显示一列。这里需要注意的是，当显示方向不同时，虽然一行显示的列数不变，但显示的布局和显示内容的排列次序却有可能大不相同。

例如：有 10 个数据需要显示，RepeatColumns 设定为 4，即一行显示 4 列时

当 RepeatDirection=RepeatDirection.Horizontal 横向显示时，显示布局如下：

```
Item1  Item2  Item3  Item4
Item5  Item6  Item7  Item8
Item9  Item10
```

当 RepeatDirection=RepeatDirection.Vertical 纵向显示时，显示布局如下：

```
Item1  Item4  Item7  Item10
Item2  Item5  Item8
Item3  Item6  Item9
```

BorderWidth：当 RepeatLayout=RepeatLayout.Table 即以表格形式显示时，边框的线宽度。

Unit.Pixel(x) $x \geq 0$, 当 x 为 0 时无边框

GridLines: 当 RepeatLayout=RepeatLayout.Table 以表格形式显示时，在表格当中是否有网隔线分离表格各单元。

GridLines=GridLines.Both，有横向和纵向两个方向的分割线。

GirdLines=GridLines.None，无论横向还是纵向均无分割线。

例子：演示以上介绍的各属性的设置对数据列表输出的影响，并且当数据项被选中时，数据项以粉红色来反显。

1. 源程序

```
<!--文件名：csbook\form\serverctl\FormDataList.aspx -->
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">
'创建初始化和载入实验数据
```

```
Function LoadData() As ICollection

    Dim dt As DataTable
    Dim dr As DataRow
    Dim i As Integer

    '创建数据表
    dt = New DataTable
    '建立数据项结构
    dt.Columns.Add(New DataColumn("Content", GetType(String)))

    '载入10个实验数据
    For i = 1 To 10
        dr = dt.NewRow()
        dr(0) = "Info " & i.ToString()
        dt.Rows.Add(dr)
    Next

    '为数据表建立一个数据视图，并将其返回
    LoadData = New DataView(dt)

End Function

Sub Page_Load(s As Object, e As EventArgs)
    If Not IsPostBack Then
        DataList1.DataSource = LoadData()
        DataList1.DataBind
    End If
End Sub

Sub DataList1_ItemCommand(s As Object, e As DataListCommandEventArgs)
    Dim cmd As String = e.CommandSource.CommandName

    If cmd = "select" Then
        DataList1.SelectedIndex = e.Item.ItemIndex
    End If

    DataList1.DataSource = LoadData()
    DataList1.DataBind
End Sub
'当刷新按钮按下后，对数据列表属性重新设置
Sub RefreshBtn_Click(s As Object, e As EventArgs)
    If lstDirection.SelectedIndex = 0
```

```
        DataList1.RepeatDirection = RepeatDirection.Horizontal
    Else
        DataList1.RepeatDirection = RepeatDirection.Vertical
    End If

    If lstLayout.SelectedIndex = 0
        DataList1.RepeatLayout = RepeatLayout.Table
    Else
        DataList1.RepeatLayout = RepeatLayout.Flow
    End If

    If chkBorder.Checked And DataList1.RepeatLayout = RepeatLayout.Table
Then
        DataList1.BorderWidth = Unit.Pixel(1)
    Else
        DataList1.BorderWidth = Unit.Pixel(0)
    End If

        If chkGridLines.Checked And DataList1.RepeatLayout =
RepeatLayout.Table then
            DataList1.GridLines = GridLines.Both
        Else
            DataList1.GridLines = GridLines.None
        End If

        DataList1.RepeatColumns=lstColsPerLine.SelectedIndex + 1
    End Sub

</script>
<head>
<title>
数据列表实验
</title>
</head>

<body>
<center>
<h2>
数据列表属性方法实验
</h2>

<form runat=server>
<font face="Verdana" size="-1">
```

```

<asp:DataList id="DataList1" runat="server"
  BorderColor="black"
  CellPadding="3"
  Font-Name="Verdana"
  Font-Size="8pt"
  HeaderStyle-BackColor="#aaaadd"
  AlternatingItemStyle-BackColor="#ccccff"
  SelectedItemStyle-BackColor="#ffccff"
  OnItemCommand="DataList1_ItemCommand"
  >
  <template name="HeaderTemplate">
    <h><center>内容</center></h>
  </template>
  <template name="ItemTemplate">
    <asp:LinkButton id="DetailBtn" runat="server" Text="详细"
CommandName="select" />
    <%# DataBinder.Eval(Container.DataItem, "Content") %>
  </template>
  <template name="SelectedItemTemplate">
    <%# DataBinder.Eval(Container.DataItem, "Content") %>已经被选
  </template>
</asp:DataList>

<p>
<hr>
显示方向:
<asp:DropDownList id=lstDirection runat="server">
  <asp:ListItem>横向</asp:ListItem>
  <asp:ListItem>纵向</asp:ListItem>
</asp:DropDownList>

布局类型:
<asp:DropDownList id=lstLayout runat="server">
  <asp:ListItem>表方式</asp:ListItem>
  <asp:ListItem>流方式</asp:ListItem>
</asp:DropDownList>

一行列数:
<asp:DropDownList id=lstColsPerLine runat="server">
  <asp:ListItem>1列</asp:ListItem>
  <asp:ListItem>2列</asp:ListItem>
  <asp:ListItem>3列</asp:ListItem>
  <asp:ListItem>4列</asp:ListItem>

```

中

```

    <asp:ListItem>5列</asp:ListItem>
</asp:DropDownList>

```

边框显示:

```
<asp:CheckBox id=chkBorder runat="server" />
```

网格显示:

```
<asp:CheckBox id=chkGridLines runat="server" />
<p>
```

```

    <asp:Button id=RefreshBtn Text="刷新界面" OnClick="RefreshBtn_Click"
runat="server"/>

```

```

</font>
</form>
</center>
</body>
</html>

```

2. 开始时的界面如图 6-10 所示(方向为横向,表方式,一行一列,无边框及网格)。



图 6-10

3. 当选择显示方向为横向, 表方式, 一行含 5 列, 显示边框和网格时, 界面如图 6-11。



图 6-11

4. 选择纵向显示, 表方式, 一行含 5 列, 无边框, 无网格时, 界面如图 6-12。



图 6-12

5. 当在步骤 4 的基础上选择了第 5 项数据项时, 界面如图 6-13。



图 6-13

接下来，讨论一种比较具有实际意义的应用，即对选中数据项的修改的实现。

首先是对模板 `EditItemTemplate` 的定义，通常做法是排列可以进行修改的内容，然后定义一个修改确认键和一个修改取消键。

然后应定义数据列表支持的三种消息处理函数即 `OnEditCommand`、`OnUpdateCommand`、`OnCancelCommand` (编辑事件处理、修改事件处理、撤消修改事件处理)

编辑事件处理：通常设置数据列表的 `EditItemIndex` 属性为选中的数据项索引，然后重载数据列表。

```
Protected Sub DataList_EditCommand(Source As Object, e As
DataListCommandEventArgs)
    DataList1.EditItemIndex = CType(e.Item.ItemIndex, Integer)
    '重新加载并绑定数据
    BindList()
End Sub
```

取消修改事件处理：通常设置数据列表的 `EditItemIndex` 为-1，表示没有数据项需要修改，然后重载数据列表

```
Protected Sub DataList_CancelCommand(Source As Object, e As
DataListCommandEventArgs)
    DataList1.EditItemIndex = -1
    BindList()
End Sub
```

修改事件处理：通常先修改数据源的数据，然后设置数据列表的 `EditItemIndex` 为-1，


```
'有session_book变量,直接引用
Else
    Book = session("session_Book")
end if
'产生数据视图,并按num字段排序
BookView = New DataView(Book)
BookView.Sort="num"
'初次需绑定数据源
if Not IsPostBack then
    BindList
End If

End Sub

'编辑处理函数
Sub DataList_EditCommand(sender As Object, e As
DataListCommandEventArgs)
    DataList1.EditItemIndex = e.Item.ItemIndex
    BindList
End Sub
'取消处理函数
Sub DataList_CancelCommand(sender As Object, e As
DataListCommandEventArgs)
    DataList1.EditItemIndex = -1
    BindList
End Sub
'更新处理函数
Sub DataList_UpdateCommand(sender As Object, e As
DataListCommandEventArgs)
    Dim lbl1 As Label = e.Item.FindControl("lblNum")
    Dim txt2 As TextBox = e.Item.FindControl("txtBook")
    Dim txt3 As TextBox = e.Item.FindControl("txtPrice")

    dim strNum as String
    dim strBook as String
    dim strPrice as String

    strNum=lbl1.text
    strBook=txt2.text
    strPrice=txt3.text
```

```
'用先删除再插入的方式,实现数据的更新操作
BookView.RowFilter = "num='" & strNum & "'"
If BookView.Count > 0 Then
    BookView.Delete(0)
End If

BookView.RowFilter = ""
dim dr as DataRow=Book.NewRow()
dr(0) = strNum
dr(1) = strBook
dr(2) = strPrice
Book.Rows.Add(dr)

DataList1.EditItemIndex = -1
BindList
End Sub

</script>
<head>
<title>
数据列表修改实验
</title>
</head>
<body>
<center>
<h2>数据列表修改实验</h2>
<hr>
<p></p>

<form runat=server>
<font face="Verdana" size="-1">
<!--编辑时显示绿色,并定义编辑、修改、取消时的处理函数-->
<asp:DataList id="DataList1" runat="server"
    BorderColor="black"
    BorderWidth="1"
    GridLines="Both"
    CellPadding="3"
    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="8pt">
```

```

Width="150px"
HeaderStyle-BackColor="#aaaadd"
AlternatingItemStyle-BackColor="Gainsboro"
EditItemStyle-BackColor="green"
OnEditCommand="DataList_EditCommand"
OnUpdateCommand="DataList_UpdateCommand"
OnCancelCommand="DataList_CancelCommand"
>
    <template name="HeaderTemplate">
    <center><h>书籍序号</h></center>
    </template>
    <template name="ItemTemplate">
        <asp:LinkButton id="button1" runat="server" Text="详细"
CommandName="edit" />
        <%# Container.DataItem("name") %>
    </template>
    <template name="EditItemTemplate">
        书籍: 序号
        <asp:Label id="lblNum" runat="server" Text='<%#
Container.DataItem("num") %>' /><br>
        书名:
        <asp:TextBox id="txtBook" runat="server" Text='<%#
Container.DataItem("name") %>' /><br>
        价格:
        <asp:TextBox id="txtPrice" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "price") %>' />
        <br>
    <center>
        <asp:Button id="button2" runat="server" Text="修改"
CommandName="update" />
        <asp:Button id="button3" runat="server" Text="撤消"
CommandName="cancel" />
    </center>
    </template>
</asp:DataList>
</font>
</form>
</center>
</body>
</html>

```

2. 准备对第 2 项进行修改, 此时的对话框如图 6-14 所示。



图 6-14

3. 把序号为 2 的书籍的价格改为 9.99 以后, 重新进入其编辑状态后, 它的输出对话框如图 6-15。



图 6-15

6.6 数据表格 DataGrid

数据表格服务器端控件以表格形式显示数据内容，同时还支持数据项的选择、排序、分页和修改。缺省情况下，数据表格为数据源中每一个域绑定一个列，并且根据数据源中每一个域中数据的出现次序把数据填入数据表格中的每一个列中。数据源的域名将成为数据表格的列名，数据源的域值以文本标识形式填入数据表格中。

通过直接操作表格的 Columns 集合，可以控制数据表格各个列的次序、表现方式以及显示内容。缺省的列为 Bound 型列，它以文本标识的形式显示数据内容。此外，还有许多类型的列类型可供用户选择。

列类型的定义有两种方式：显视的用户定义列类型和自动产生的列类型（AutoGenerateColumns）。当两种列类型定义方式一起使用时，先用用户定义列类型产生列的类型定义，接着剩下的再使用自动列定义规则产生出其他的列类型定义。请注意自动定义产生的列定义不会加入 Columns 集合。

列类型介绍：

1) bound column，列可以进行排序和填入内容。这是大多数列缺省用法。

两个重要的属性为：HeaderText 指定列的表头显示 DataField 指定对应数据源的域

2) hyperlink column，列内容以 hyperlink 控件方式表现出来。它主要用于从数据表格的一个数据项跳转到另外的一个页面，做出更详尽的解释或显示。

重要的属性有：

HeaderText 指定列表头的显示

DataNavigateUrlField 指定对应数据源的域作为跳转时的参数

DataNavigateUrlFormatString 指定跳转时的 url 格式

DataTextField 指定数据源的域作为显示列内容来源

3) button column，把一行数据的用户处理交给数据表格所定义的事件处理函数。通常用于对某一行数据进行某种操作，例如，加入一行或者是删去一行数据等。

重要的属性有：

HeaderText 指定列表头的显示

Text 指定按钮上显示的文字

CommandName 指定产生的激活命令名

4) Template column，列内容以自定义控件组成的模板方式显示出来。通常用作用户需要自定义显示格式的时候。

5) Edit Command column，当数据表格的数据项发生编辑、修改、取消修改时，相应处理函数的入口显示。它通常结合数据表格的 EditItemIndex 属性来使用，当某行数据需要编辑、修改、取消操作时，通过它进入相应的处理函数。例如，当需要对某行数据进行修改（update）时，通过它进入修改的处理步骤中。

其他重要列属性介绍：

1) Visible 属性，控制定义的列是否出现在显示的数据列表中。

2) AllowSorting 属性，是否可以进行列排序。当 AllowSorting=true 时，可以以点击列的列表头的方式，把数据以该列次序进行排序。缺省的（即载入数据后）的排序方式，实际上是以数据在数据源中的排列次序进行排序的。

3) AllowPage 属性，是否以分页方式显示数据。当对有大量数据的数据源进行显示时，可以以例如 10 行一页的方式来显示数据，同时显示一个下页/前页的按钮，按下按钮可以以向前或向后的方式浏览整个数据源的数据。当 AllowPage=true 时，即以分页方式进行显示。可以通过设定 CurrentPageIndex 属性来直接跳转到相应的数据页。

例子：演示以上各种类型的列定义的法

1. 源程序

```
<!-- 文件名：csbook\form\serverctl\FormDataGrid.aspx -->

<%@ Import Namespace="System.Data" %>

<html>
<script language="C#" runat="server" ID="Script1">

    DataTable Cart;
    DataView CartView;

    //创建数据源，
    ICollection CreateDataSource() {
        DataTable dt = new DataTable();
        DataRow dr;

        dt.Columns.Add(new DataColumn("IntegerValue", typeof(Int32)));
        dt.Columns.Add(new DataColumn("StringValue", typeof(string)));
        dt.Columns.Add(new DataColumn("DateTimeValue", typeof(DateTime)));
        dt.Columns.Add(new DataColumn("BoolValue", typeof(bool)));
        dt.Columns.Add(new DataColumn("CurrencyValue", typeof(double)));

        for (int i = 0; i < 9; i++) {
            dr = dt.NewRow();

            dr[0] = i;
            dr[1] = "Item " + Int32.ToString(i);
            dr[2] = DateTime.Now;
            dr[3] = (i % 2 != 0) ? true : false;
            dr[4] = 1.23 * (i+1);
        }
    }
}
</script>
</html>
```



```
        dt.Rows.Add(dr);
    }

    DataView dv = new DataView(dt);
    return dv;
}

//页面导入
void Page_Load(Object sender, EventArgs e)
{
    if (Session["DG5_ShoppingCart"] == null) {
        Cart = new DataTable();
        Cart.Columns.Add(new DataColumn("Item", typeof(string)));
        Cart.Columns.Add(new DataColumn("Price", typeof(string)));
        Session["DG5_ShoppingCart"] = Cart;
    }
    else {
        Cart = (DataTable)Session["DG5_ShoppingCart"];
    }
    CartView = new DataView(Cart);
    ShoppingCart.DataSource = CartView;
    CartView.Sort="Item";
    ShoppingCart.DataBind();

    MyDataGrid.DataSource = CreateDataSource();
    MyDataGrid.DataBind();
}

void Grid_CartCommand(object sender, DataGridCommandEventArgs e) {

    DataRow dr = Cart.NewRow();

    // e.Item is the row of the table where the command fired
    // For bound columns the value is stored in the Text property of
    TableCell
    TableCell itemCell = e.Item.Cells[1];
    TableCell priceCell = e.Item.Cells[2];
    string item = itemCell.Text;
    string price = priceCell.Text;

    if (((LinkButton)e.CommandSource).CommandName == "AddToCart") {
```

```
        dr[0] = item;
        dr[1] = price;
        Cart.Rows.Add(dr);
    }
    else { //Remove from Cart

        CartView.RowFilter = "Item='"+item+"'";
        if (CartView.Count > 0) {
            CartView.Delete(0);
        }
        CartView.RowFilter = "";
    }
    ShoppingCart.DataBind();
}

</script>

<body>

    <h3><font face="Verdana">.NET-> DataGrid</font></h3>

    <form runat=server ID=Form1>

    <table cellpadding="5">
    <tr valign="top">
    <td>

    <b>Product List</b>
    <asp:DataGrid id="MyDataGrid" runat="server"
        BorderColor="black"
        BorderWidth="1"
        GridLines="Both"
        CellPadding="3"
        CellSpacing="0"
        Font-Name="Verdana"
        Font-Size="8pt"
        HeaderStyle-BackColor="#aaaadd"
        AutoGenerateColumns="false"
        OnItemCommand="Grid_CartCommand"
    >
    <property name="Columns">
```

```
<asp:TemplateColumn HeaderText="Add/Remove">
    <template name="ItemTemplate">
        <asp:LinkButton ID=AddButton Text="Add"
CommandName="AddToCart" ForeColor="blue" runat="server" />&nbsp;
        <asp:LinkButton ID=RemoveButton Text="Remove"
CommandName="RemoveFromCart" ForeColor="blue" runat="server" />
    </template>
</asp:TemplateColumn>

<asp:BoundColumn HeaderText="Item" DataField="StringValue" />

<!-- format Price as currency-->
<asp:BoundColumn HeaderText="Price" DataField="CurrencyValue"
DataFormatString="{0:c}" ItemStyle-HorizontalAlign="right" />

<asp:TemplateColumn HeaderText="Assembly required?">
    <template name="ItemTemplate">
        <asp:CheckBox ID=Chk1 Checked='<%#
DataBinder.Eval(Container.DataItem, "BoolValue") %>' Enabled="false"
runat="server" />
    </template>
</asp:TemplateColumn>
</property>
</asp:DataGrid>

</td><td>

<b>Shopping Cart</b>
<asp:DataGrid id="ShoppingCart" runat="server"
    BorderColor="black"
    BorderWidth="1"
    CellPadding="3"
    Font-Name="Verdana"
    Font-Size="8pt"
    HeaderStyle-BackColor="#aaaadd"
    />

</td>
</tr>
</table>

</form>
```

```
</body>
```

```
</html>
```

2. 当选择订购了第一本和第三本后的对话框如图 6-16 所示。

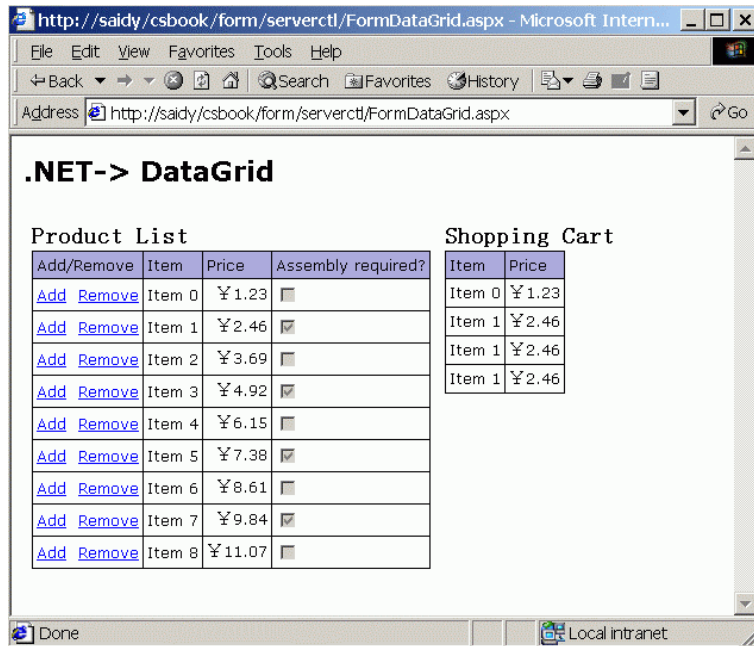


图 6-16

6.7 小 结

本章主要介绍了几个服务器端的控件、它们的校验、取值方法等，从中可以看到 ASP.NET 中各种控件功能是非常强大的，如上面的例子所示，我们甚至可以用一个简单的语句就可以验证输入的合法性。对取值，我们也有简单的方法，对比于用 html 所写的代码，我们觉得用 ASP.NET 所写的是简单了很多。

第 7 章 自定义与 HTML 控件

7.1 自定义控件

ASP.NET 中提供的增加内嵌服务器控件的功能，允许多次的轻松增加你所定义的各种控件。事实上，对于表单等各种控件，可以不用更改或者稍微更改一下就可以多次使用的。在通常情况下，把一个用作服务器控件的 Web 表单统称为用户控件，我们用一个 .ascx 为后缀的文件保存起来，这样的保存使得它不被当作一个 Web 表单来运行，当在一个 .aspx 文件中使用它时，我们用 Register 方法来进行调用，假设我们有一个文件名为 saidy.ascx 的文件，用下面的语句来调用它：

```
<%@ Register TagPrefix="Acme" TagName="Message" Src="saidy.ascx" %>
```

上面的 TagPrefix 标记为用户控件确定个唯一的名字空间，TagName 为用户控件确定一个唯一的名称，也可以用其它的名字代替“Message”，Src 为确定所包含的文件名称和路径。这样，就可以用下面的语句来调用它了：

```
<Acme:Message runat="server"/>
```

下面来看看具体的应用。

7.1.1 小页面控件

建立两个简单文件来说明这个空间的使用方法：con01.aspx、con01.ascx，在 con01.ascx 文件里我们只有一句话：

```
<a href="http://www.yesky.com">欢迎访问天极网站</a>
```

然后在文件 con01.aspx 里面进行注册：

```
<%@ Register TagPrefix="saidy" TagName="info" Src="con01.ascx" %>
```

页面上的应用我们用这句话来表达：

```
<saidy:info runat="server"/>
```

文件的完整代码如下：

```
( csbook\control\con01.aspx )
```

```
<!--注册小页面控件-->
```

```
<%@ Register TagPrefix="saidy" TagName="info" Src="con01.ascx" %>
```

```
<html>
```

```
<body BGCOLOR="#CCCCCC">
```

```
<BR><BR><BR>
```

```
<CENTER>
```

调用结果

```
<BR><BR>
```

```

    <saidy:info runat="server" />
</CENTER>
<BR><BR>
</body>
</html>

```

下面我们访问 con01.aspx，显示如图 7-1。



图 7-1

7.1.2 分离代码(Code Behind)

分离代码英文为 Code Behind，自从微软的 ASP.NET 问世以来，这一直是一个热门的话题。基本的 Code Behind 允许用两个文件来创建一个 ASP.NET 页面，一个表达文件和一个代码文件。表达文件通常为.aspx 文件或者.ascx 文件，而代码文件通常是.vb 文件或者.cs 文件，用这种风格编写的页面允许把内容和代码完全分开来，而且在比如发布你的网站的时候可以有效的保护你的代码。这个内容很重要，所以下面将在 Web form 和 user control 中举两个例子来讲解怎样应用 Code Behind 技术，这两个例子用 vb.net 来实现。

7.1.2.1 Web form 的 Code Behind

在 Web form 中可以用 Code Behind，aspx 文件代码为：

(csbook\control\codebehind\2001012601.aspx)：

```

<%@ Page Language="VB" Inherits="myPage" Src="2001012601.vb" %>
<html>
<body style="font: x-small Verdana, Arial, sans-serif;">
<form runat="server" method="post" ID=Form1>
<%-- The label control is simply a place holder --%>
<asp:label id="soonToBeUserControl" runat="server" />
</form>
</body>
</html>

```

在上面的文件中有一个 server control，一个名为 soonToBeUserControl 的标签，第一句话很重要，在这句话中，指定了两个属性：inherits 和 Src，其中 inherits 用来识别本页面所

用的类的来自何处,Src 识别代码文件。这样这句话指定了页面所用的类名和类所在文件来源,这就是要用的控制页面的所有代码。下面创建一个 Code Behind 文件:

```
' Option Strict is Off to allow dim of myUserControl without an As clause
Option Strict Off

Imports System
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls

Public Class myPage : Inherits Page

    'Declare the Label which is just a place holder for the user control
    Public soonToBeUserControl As Label

    Public Sub Page_Load(Source As Object, E As EventArgs)

        'Dynamically load the user control to the Controls collection
        Dim tempUserControl = Page.LoadControl("2001012601uc.ascx")
        'Set the user control property
        tempUserControl.myProp = "This property was set in Code Behind"
        'Add the user control to the Label place holder
        soonToBeUserControl.Controls.Add(tempUserControl)

    End Sub

End Class
```

这样一个 Code Behind 文件就创建了,首先设置 Option Strict Off,因为需要做一些事情,引入一些 Namespace 使得这个 Code Behind 文件能够使用。其中引入的 Namespace 中, System.Web.UI 允许你访问页面类, System.Web.UI.WebControls 允许访问一个 ASP.NET server control 比如象 Label。之后对类进行定义,制定类的继承类等。

7.1.2.2 User control 中的 Code Behind

就象 Web form 中可以用 Code Behind 技术一样,在其他的场合一样可以的。在 user control 中,道理是一样的,唯一的一个就是继承的类不同。在这里,类的继承类为 UserControl,下面是一个 .ascx 文件:

```
( csbook\control\codebehind\2001012601uc.ascx )

<%@ Control Inherits="myUserControl" src="2001012601uc.vb" %>
<p><asp:Label runat="server" id="myLabel" /></p>
```

```
<p><asp:DataGrid runat="server"
    id="myDataGrid"
    Font-Size="x-small"
    HeaderStyle-Font-Bold="True"
    HeaderStyle-ForeColor="White"
    HeaderStyle-BackColor="Brown"
    AlternatingItemStyle-BackColor="Tan" /></p>
```

在上面的文件中，有一个 Label 控件和一个 DataGrid 控件，这些控件从 Code Behind 文件的类中获得信息，同样得在第一句话里面，引入了类。

Code Behind 文件的代码如下：

(csbook\control\codebehind\2001012601uc.vb)

```
Imports System
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls
Imports System.Data
Imports System.Data.SqlClient

Public Class myUserControl : Inherits UserControl

    Public myDataGrid As DataGrid
    Public myLabel As Label

    Public Property myProp As String
        Get
            Return myLabel.Text
        End Get

        Set
            myLabel.Text = Value
        End Set
    End Property

    Public Sub Page_Load(Source As Object, E As EventArgs)
        Dim ds As New DataSet
        Dim dsc As SQLDataSetCommand

        dsc = New SQLDataSetCommand("SELECT CompanyName, ContactName,
ContactTitle FROM Customers", _"server=localhost;uid=sa;pwd=;database
=Northwind;")
```



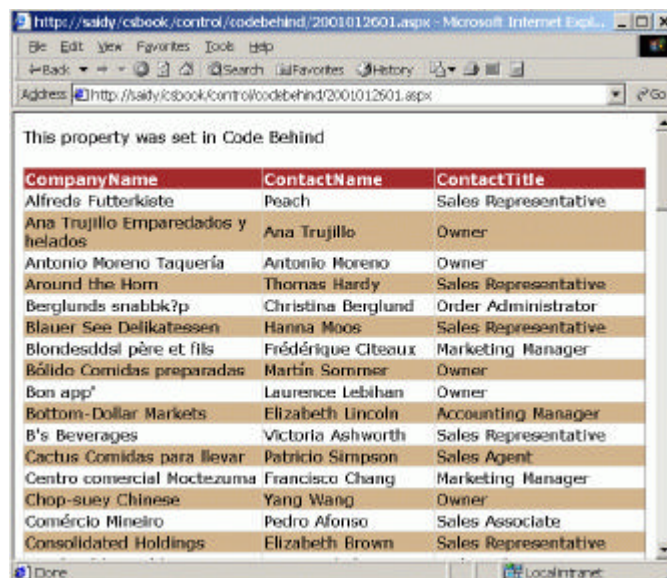
```
dsc.FillDataSet(ds, "Customers")
myDataGrid.DataSource = ds.Tables("Customers").DefaultView
myDataGrid.DataBind()
End Sub
```

```
End Class
```

从数据库中选择一组数据，由于在文件 2001012601.vb 中引用到了 2001012601uc.ascx 文件，所以在页面上看到了数据库中的数据。

7.1.2.3 运行结果

运行结果如图 7-2 所示。



CompanyName	ContactName	ContactTitle
Alfreds Futterkiste	Peach	Sales Representative
Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
Antonio Moreno Taquería	Antonio Moreno	Owner
Around the Horn	Thomas Hardy	Sales Representative
Berglunds snabbkp	Christina Berglund	Order Administrator
Blauer See Delikatessen	Hanna Moos	Sales Representative
Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager
Bólido Comidas preparadas	Martin Sommer	Owner
Bon app'	Laurence Leblhan	Owner
Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager
B's Beverages	Victoria Ashworth	Sales Representative
Cactus Comidas para llevar	Patricio Simpson	Sales Agent
Centro comercial Moctezuma	Francisco Chang	Marketing Manager
Chop-suey Chinese	Yang Wang	Owner
Comércio Mineiro	Pedro Afonso	Sales Associate
Consolidated Holdings	Elizabeth Brown	Sales Representative

图 7-2

7.1.3 自定义控件

在 ASP.NET 中，除了应用的服务端控件之外，还可以创建自己的服务端控件，这样的控件叫 Pagelet。下面介绍如何创建一个 Pagelet，这个 Pagelet 的功能是在被访问时返回一个消息。

创建一个 Pagelet，用来返回一个消息在客户端的浏览器上：

```
Welcome.ascx :
```

```
欢迎来到我这里啊！！！
```

就这么简单，当然你也可以让它复杂一点。当一个 Pagelet 被创建后，就可以通过下面的记录指示来调用它：

```
<% @ Register TagPrefix="wmessage" TagName="wname" Src="Welcome.ascx" %>
```

TagPrefix 为 Pagelet 指定一个唯一的名字空间，TagName 是 Pagelet 的唯一名字，当然也可以换成其他的不是“wname”的名称如：TagName="saidy"。Src 属性是指指向 Pagelet 的虚拟路径。

一旦注册了 Pagelet，就可以象用普通的控件一样来应用它：

```
<wmessage:wname runat="server" />
```

下面的例子示范了自定义的控件的应用：

```
<%@ Register TagPrefix="wmessage" TagName="wname" Src="Welcome.ascx" %>
```

```
<html>
```

```
<body BGCOLOR="#CCCCCC" style="font: 10pt verdana">
```

```
<h3>.NET->Pagelet</h3>
```

```
<wmessage:wname runat="server" />
```

```
</body>
```

```
</html>
```

客户端的访问如图 7-3。

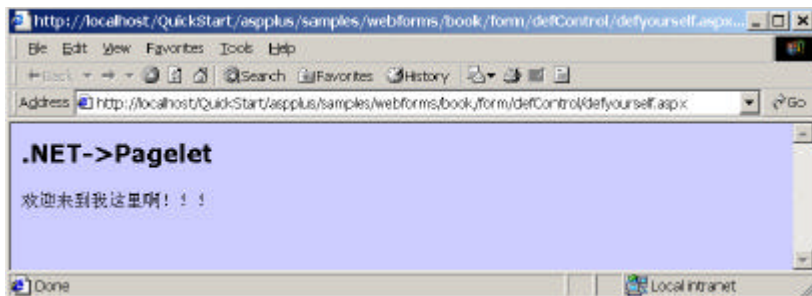


图 7-3

7.1.4 组合控件

7.1.4.1 定义

以类组合形式把已有的控件编译后形成自己定制的控件。实际上组合控件在结果上与利用内置控件形成的用户自定义控件一样，不同处在于，用户自定义控件含有一个.ascx 的纯文本控制文件，而组合控件则利用编译后的代码。

7.1.4.2 步骤

- 1) 重新定义从 Control 继承来的 CreateChildControls 方法。
- 2) 如果组合控件要保持于页面上，须完成 System.Web.UI.INamingContainer 接口。

7.1.4.3 组合控件的例子

演示一个自定义控件，当选择不同按钮时显示不同内容。

1) 控件定义

```
'文件名:FormCustom.vb
Option Strict Off

Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls

Namespace test
'定义类tryVB
Public Class tryVB : Inherits Control : Implements INamingContainer
'定义属性value,实为TextBox控件的Text属性
Public Property value As String
Get
Dim Ctrl As TextBox = Controls(1)
Return Ctrl.text
End Get

Set
Dim Ctrl As TextBox = Controls(1)
Ctrl.Text = value
End Set
End Property

Protected Overrides Sub CreateChildControls()
'重载CreateChildControls方法
Me.Controls.Add(New LiteralControl("选择结果为: "))

Dim Box As New TextBox
Box.Text = " "
Me.Controls.Add(box)

End Sub

End Class

End Namespace
```

2) 定义控件的编译

```
rem filename: FormCustom.bat
vbc /t:library /out:..\bin\testVB.dll /r:System.dll /r:System.Web.dll
FormCustom.vb
```

请注意把生成的 testVB.dll 放到正确的目录中，以便 ASP.NET 解释时能够找到相应的类。

3) 自定义组合控件的使用

```
<!--文件名:FormCustom.aspx-->
<%@ Register TagPrefix="test" Namespace="test" %>
<!--首先注册test命名空间-->
<html>
<script language="VB" runat=server>
    Private Sub LeftBtn_Click(Sender As Object, E As EventArgs)
        '当选择左边的按钮时的显示
        CustControl.Value = "您选择的是Yes按钮"
    End Sub

    Private Sub RightBtn_Click(Sender As Object, E As EventArgs)
        '当选择右边的按钮时的显示
        CustControl.value = "您选择的是No按钮"
    End Sub
</script>

<body>
<center>
    <form method="POST" action="formcustom.aspx" runat=server>
        <!--引用自定义的组合控件tryVB-->
        <test:tryVB id="CustControl" runat=server/>
        <br>
        <!--画两个按钮供选择-->
        <asp:button text="是[Yes]" OnClick="LeftBtn_Click" runat=server/>
    <asp:button text="否[No]" OnClick="RightBtn_Click" runat=server/>
    </form>
</center>
</body>
</html>
```

输出结果如图 7-4 所示。



图 7-4

7.1.5 继承控件

在学习了微软公司的.NET 平台为我们提供的大量功能强大的服务器端控件的使用方法以后,随着应用的深入,一些新的问题又出现了。首先是虽然有着大量的控制灵活的控件,但是否就真的满足了我们所有的需求?有时候,我们需要某种控件部分功能,又希望不要费太大的力气去实现,是否可以利用现有的控件来实现。再则,我们希望对某种控件进行改造,使它具有自己所希望的外形或者结果,而不是它缺省的方式运行。最后我们是否可以把自己经常用到的逻辑规则或者是应用界面做成用户控件,然后使用它就如同使用服务器控件那样方便。

其实以上三个问题,在现代面向对象设计方法中,是可以找到答案的。为最大可能的利用现有的开发成果,我们使用“继承”这一手段来节省开发的费用。光有继承不足以形成自己的应用,我们还可以利用“重载”和“多态”来形成自己的应用特点,使之区别于被继承的对象。为了使应用更加简洁和对外隐藏内部的实现、进一步实现代码重用,我们又使用了“封装”。

微软的.NET 平台是支持面向对象的设计方式的新型平台,所以支持并且鼓励用户在应用中设计和使用自己定义的控件。设计用户自己的控件就如同上面所述,有如下步骤:

1. 从 System.Web.UI.Control 类继承,并形成自己的类

为继承 Control 类,我们需引用 System、System.Web、System.Web.UI 类库,在 VB.NET 环境下使用标识 Import 来引入。为方便使用,我们还需定义一个命名空间以容纳多个类。在 VB.NET 环境中使用 Namespace 空间名和 End Namespace 标识对来定义一个命名空间。定义一个类使用 Class ClassName 和 End Class 标识对。为表明类之间的继承关系,可以使

用 Inherits 标识。

继承控件的类定义框架定义如下：

```
Imports System
Imports System.Web
Imports System.Web.UI

Namespace MyNamespace
    Public Class MyClass:Inherits Control
    ...
    End Class
End Namespace
```

一个最简单的例子是从 Control 继承一个类，然后重载其 Render 方法。调用其 Render 方法即在页面以 h2 字体写出一行字。

```
Imports System
Imports System.Web
Imports System.Web.UI

Namespace MyNamespace
Public Class MyClass:Inherits Control

Protect overrides Sub Render(Output as HtmlTextWriter)
    Output.Write("<h2>这是一个最简单的控件继承例子！</h2>")
End Sub
End Class
End Namespace
```

2. 定义自己的属性和方法，包括重载一些初始化的方法

在 vb 中，以标识 overrides 指明该方法是一个重载函数。例如上面所举的 Render 方法：

```
Protect overrides Sub Render(Output as HtmlTextWriter)
```

属性定义就较为复杂一点，首先是定义内部变量，可以为 Public 或者是 Private, 当为 Public 时可以被外部直接存取，这种方式面向对象方法并不提倡，为 Private 时，不能直接被外部存取，只有通过内部提供的属性定义方式来存取；然后对需要提供给外部使用的内部变量进行属性存取方式定义。在 vb 中使用 Property 属性名 As 类型和 End Property 标识对来定义，Get/End Get 标识对间定义如何通过属性取得内部变量的值，Set/End Set 标识对间定义如何设置内部变量值。

例如：描述一个人的帐号信息，大致需要设定帐号 (AcctNo) 身份证号 (IdNo) 余额 (Balance) 有效状态(Stat)

```
Imports System
Imports System.Web
Imports System.Web.UI
```

```
Namespace MyNamespace
`定义一个枚举变量,0-正常 1-销户 2-其他状态(挂失、冻结等等)
    Public Enum Status
        Active = 0
        Deactive = 1
        Other = 2
    End Enum

    Public Class Account : Inherits Control

        Private _AcctNo As String
        Private _IdNo As String
        Private _Balance As Currency
        Private _Stat As Status

        Public Property AcctNo As String
            Get
                Return _AcctNo
            End Get
            Set
                _AcctNo = Value
            End Set
        End Property

        Public Property IdNo As String
            Get
                Return _IdNo
            End Get
            Set
                _IdNo = Value
            End Set
        End Property

        Public Property Balance As Currency
            Get
                Return _Balance
            End Get
            Set
                _Balance = Value
            End Set
        End Property
    End Class
End Namespace
```

```
Public Property Stat As Status
    Get
        Return _Stat
    End Get
    Set
        _Stat = Value
    End Set
End Property
...
End Class
```

```
End Namespace
```

而方法的定义就比较灵活，可以根据设计要求，提供相应的功能，例如大多数的类一般都会提供创建或者是初始化类的方法。我们仍以上面的帐号类为例，定义一个 New 方法：

```
...
Public Sub New(AcctNo1 As String, IdNo1 As String, Balancel As Currency, Stat1
As Status)
    MyBase.New
    Me.AcctNo = AcctNo1
    Me.IdNo = IdNo1
    Me.Balance = Balancel
    Me.Stat = Stat1
End Sub
...
```

3. 定义自己应用界面

一个用户自定义的控件一般来说较为复杂，由至少一个以上的内置控件构成，这时就需要重载从 Control 类继承来的 CreateChildControls 方法，并在其中生成界面控件。如果用户定义的控件会在一个页面中反复使用，最好 implements System.Web.UI.INamingContainer，它会为该控件创建一个唯一的命名空间。

例如：下面的例子将创建一个控件，它由一段说明文字和一个文本输入框构成。

```
Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls

Namespace MyNamespace
Public Class Myclass : Inherits Control : Implements INamingContainer
...
Protected Overrides Sub CreateChildControls()
```



```

        Me.Controls.Add(New LiteralControl("<h3>请输入: "))

        Dim txtBox As New TextBox
        txtBox.Text = ""
        Me.Controls.Add(txtBox)

        Me.Controls.Add(New LiteralControl("</h3>"))
    End Sub

    ...
End Class
End Namespace

```

4. 定义自己控件的消息处理函数

自己定义的控件含有两种类型的消息，一是包含的子控件所产生的消息，二是自定义的控件消息。

子控件产生的消息处理函数可由 AddHandler 函数来指定，其用法如下：

AddHandler 子控件.消息, AddressOf 消息处理函数

例如：自定义控件中含有一个 Button 控件，并定义其处理函数 MyBtn_Click()

```

    ...
    Private Sub MyBtn_Click(Sender as Objects, E as EventArgs)
    ...
    End Sub

```

```

Protected override Sub CreateChildControls()
    ...
    Dim MyBtn As New Button
    MyBtn.text=""
    AddHandler MyBtn.Click, AddressOf MyBtn_Click
    Me.Controls.Add(MyBtn)
    ...
End Sub

```

自定义的控件消息则需要先定义事件说明，格式如下

Public Event 消息名(Sender as Object,E as EventArgs)

例如：Public Event Change(Sender as Object,E as EventArgs)

然后定义事件发出函数，例如：

```

Protected Sub OnChange(E as EventArgs)
    RaiseEvent Change(Me,E)
End Sub

```

再然后定义引起事件发生的过程（可不写）

例如：

```

Private Sub TextBox_Change(Sender As Object, E As EventArgs)
    OnChange(EventArgs.Empty)
End Sub

```

最后定义何时触发事件函数，同样使用 AddHandler 函数。

例如：

```

...
Protected override Sub CreateChildControls()
...
Dim MyBox as New TextBox
MyBox.Text=""
AddHandler MyBox.TextChanged , AddressOf TextBox_Change
Me.Controls.Add(MyBox)
...
End Sub
...

```

5. 最后，谈一谈继承控件的使用，首先应把预先写好的继承控件编译成.DLL 文件编译格式为：

```
vbc /t:library /out:MyDll.dll /r:System.Web.dll MyVb.vb
```

vbc为vb.net的编译器

/t:表示编译类型，library为链接库，exe为独立可执行文件

/out:指定输出文件名

/r:表示需要引用的DLL文件

MyVb.vb:指自己编写的继承控件vb源程序

然后，为在自己的页面中引用自己定义的控件，需在 aspx 文件头进行注册，

```
<%@ Register TagPrefix="标记前缀" Namespace="命名控件" %>
```

最后，就如同使用内置控件一样，在页面中使用自己定义的控件：

```
<命名空间名:类名 ..... runat=server />
```

下面举一个具体的例子来说明：

仍然以开始定义的用户帐号为例来定义一个继承控件，该类有 4 个属性分别为客户帐号、身份证号、帐户余额、帐户状态，其用户界面设定为 4 个文本框供输入属性值以供修改，另外加 2 个按钮以供确认，同时为该控件设定一个事件 Click,当按下确认键后，修改控件属性值，并且在页面中显示自定义控件的属性值，以确认事件确实生效了。

1. 控件定义文件

'文件名：Inherit.vb

```
Option Strict Off
```

```
Imports System
```

```
Imports System.Web
```

```
Imports System.Web.UI
```

```
Imports System.Web.UI.WebControls

Namespace MyNamespace

    Public Enum Status
        Active    = 0
        Deactive  = 1
        Other     = 2
    End Enum

Public Class MyAccount:Inherits Control:Implements INamingContainer
'从Control类继承,并且有自己的命名空间
    Private _AcctNo As String
    Private _IdNo As String
    Private _Balance As Decimal
    Private _Stat As Status

    Public Event Click(Sender as Object,E as EventArgs)
'定义控件自身的Click事件

'对属性存取的定义
    Public Property AcctNo As String
        Get
            Return _AcctNo
        End Get
        Set
            _AcctNo = Value
        End Set
    End Property

    Public Property IdNo As String
        Get
            Return _IdNo
        End Get
        Set
            _IdNo = Value
        End Set
    End Property

    Public Property Balance As Decimal
        Get
            Return _Balance
        End Get
        Set
```

```
        _Balance = Value
    End Set
End Property

Public Property Stat As Status
    Get
        Return _Stat
    End Get
    Set
        _Stat = Value
    End Set
End Property

Public Sub New()
    MyBase.New
    Me.AcctNo = ""
    Me.IdNo = ""
    Me.Balance = "0.0"
    Me.Stat = "0"
End Sub

Protected Sub OnClick(E as EventArgs)
    RaiseEvent Click(Me,E)
End Sub

'界面定义为4个属性文本输入框，加一个确定和取消键
Protected Overrides Sub CreateChildControls()
    Me.Controls.Add(New LiteralControl("<h3>客户帐号："))
    dim txtAcctNo as New TextBox
    txtAcctNo.text=_AcctNo
    Me.Controls.Add(txtAcctNo)

    Me.Controls.Add(New LiteralControl("<br>身份证号："))
    dim txtIdNo as New TextBox
    txtIdNo.text=_IdNo
    Me.Controls.Add(txtIdNo)

    Me.Controls.Add(New LiteralControl("<br>帐户余额："))
    dim txtBalance as New TextBox
    txtBalance.text=_Balance
    Me.Controls.Add(txtBalance)

    Me.Controls.Add(New LiteralControl("<br>帐户状态："))
    dim txtStat as New TextBox
```

```
txtStat.text=_Stat
Me.Controls.Add(txtStat)

Me.Controls.Add(New LiteralControl("<br><br><br>"))
dim Btn1 as New Button
Btn1.text="确认"
AddHandler Btn1.Click,AddressOf Btn1_Click
Me.Controls.Add(Btn1)

dim Btn2 as New Button
Btn2.text="取消"
Me.Controls.Add(Btn2)

Me.Controls.Add(New LiteralControl("</h3>"))
End Sub

Private Sub Btn1_Click(Sender as Object,E as EventArgs)
dim ctrl1 as TextBox=controls(1)
Me.AcctNo=ctrl1.text
dim ctrl2 as TextBox=controls(3)
Me.IdNo=ctrl2.text
dim ctrl3 as TextBox=controls(5)
Me.Balance=Cdbl(ctrl3.text)
dim ctrl4 as TextBox=controls(7)
Me.Stat=Cint(ctrl4.text)
OnClick(E)
End Sub

End Class
```

End Namespace

2. 编译文件

rem inherit.vb 的编译文件 文件名:i.bat

```
vbc /t:library /out:.\bin\MyNamespaceVB.dll /r:System.Web.dll inherit.vb
```

3. 页面使用文件

<!--文件名:FormInherit.aspx-->

```
<%@ Register TagPrefix="MyNamespace" Namespace="MyNamespace" %>
```

```
<html>
```

```
<script language="vb" runat=server>
```

```
sub acct_click(s as object, e as eventargs)
```

```
dim strTxt as string
```

```
strTxt="<hr>AcctNo=" & acct1.AcctNo & "<br>"
```

```
strTxt=strTxt & "IdNo=" & acct1.IdNo & "<br>"
```

```
strTxt=strTxt & "Balance=" & acct1.Balance & "<br>"
```

```
strTxt=strTxt & "Stat=" & acct1.Stat
response.write(strTxt)
end sub
</script>
<head>
<title>
继承控件实验
</title>
</head>

<body bgcolor=#ccccff>
<center>
<h2>继承控件MyAccount的使用</h2>
<hr>
<br>
<form action="forminherit.aspx" method="post" runat=server>
<MyNamespace:MyAccount id="acct1" AcctNo="1234" IdNo="5678"
OnClick="acct_click" runat=server />
</form>
</center>
</body>
</html>
```

4. 开始的输出对话框如图 7-5。



图 7-5

5. 修改后，按“确认”键后的结果，如图 7-6 所示。

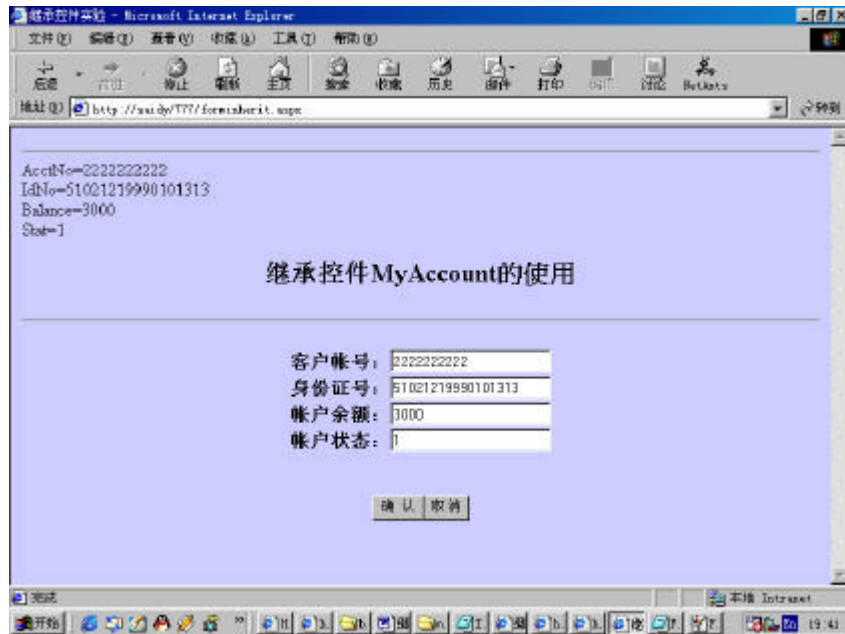


图 7-6

7.1.6 小结

本章主要介绍页面控件、分离代码、自定义控件、组合控件和继承控件。

7.2 HTML 控件

HTML 控件在服务器端是可见的，所以可以用它来按照我们的要求来编写。HTML 控件表现为一些可见的控件。

7.2.1 HtmlButton

HtmlButton server control 就象 HTML4.0 中的 `<button>` 一样，但是这与 `<Input type="button">` 不一样的，看下面的例子：

(csbook\form\htmlform\button.aspx)

响应按钮事件：

```
<script language="VB" runat="server">
    Sub Button1_OnClick(sender As Object, e As EventArgs)
        Span1.InnerHtml="你点击了Button1"
    End Sub
    Sub Button2_OnClick(sender As Object, e As EventArgs)
        Span1.InnerHtml="你点击了Button2"
    End Sub
```

```
</script>
```

对两个 button 的描述：

button1：

```
<button id="Button1" onServerClick="Button1_OnClick" style="font: 8pt
verdana;background-color:lightgreen;border-color:black;height=30;width:1
00" runat="server">
```

```
 Click me!
```

```
</button>
```

button2，我们增加了鼠标事件：

```
<button id=Button2 onServerClick="Button2_OnClick" style="font: 8pt
verdana;background-color:lightgreen;border-color:black;height=30;width:
100"
```

```
onmouseover="this.style.backgroundColor='yellow' "
```

```
onmouseout="this.style.backgroundColor='lightgreen' "
```

```
runat="server">
```

```
Click me too!
```

```
</button>
```

点击 button2，并把鼠标移到它的上面，如图 7-7 所示。

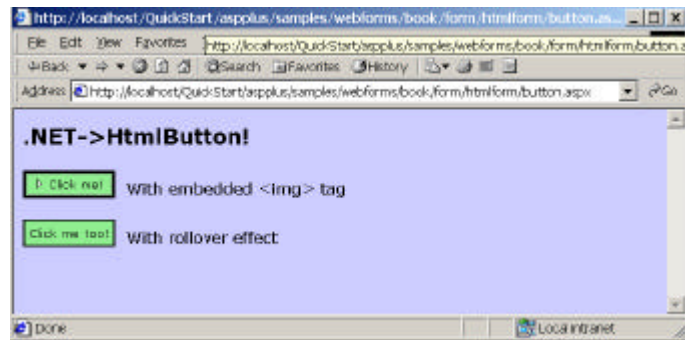


图 7-8

看到图 7-8 的结果。

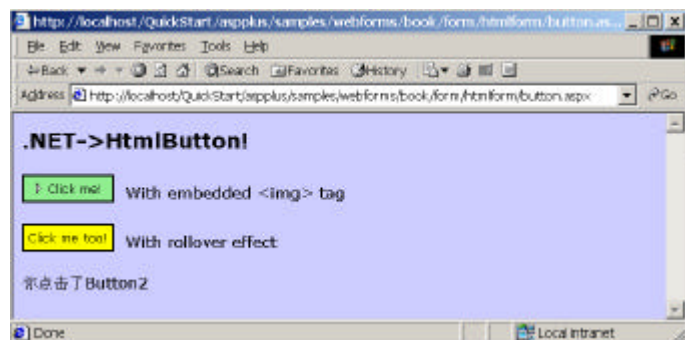


图 7-8

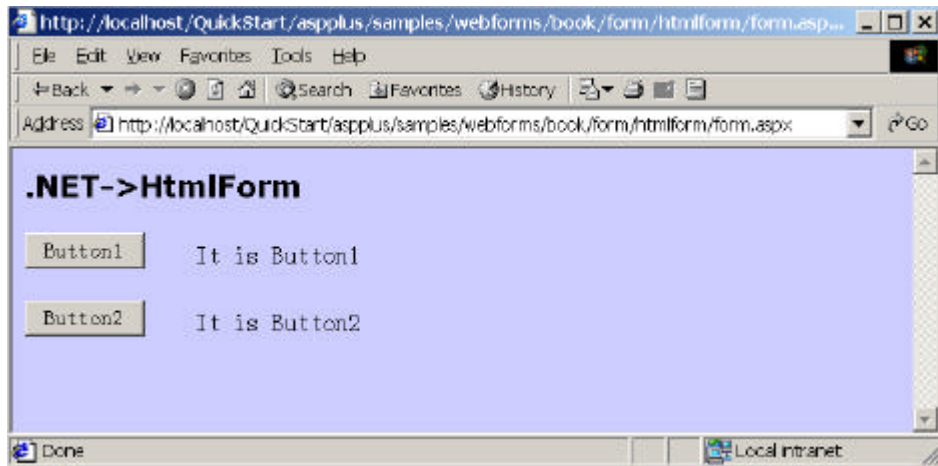
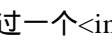


图 7-10

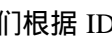
7.2.3 HtmlImages

通过一个这个标记来显示对话框：

(csbook\form\htmlform\images.aspx)

```

```

我们根据 ID 号为提供图片来源：

```
Sub SubmitBtn_Click(sender As Object, e As EventArgs)
```

```
    Image1.Src="images/" & Select1.Value
```

```
End Sub
```

建立一个选择控件来与用户的交互：

选择面部表情文件：

```
<select id="Select1" runat="server">
```

```
    <option Value="4.gif">4</option>
```

```
    <option Value="5.gif">5</option>
```

```
    <option Value="6.gif">6</option>
```

```
    <option Value="7.gif">7</option>
```

```
    <option Value="8.gif">8</option>
```

```
    <option Value="9.gif">9</option>
```

```
</select>
```

```
    <input type="submit" runat="server" Value="提交"
```

```
OnServerClick="SubmitBtn_Click">
```

我们运行如图 7-11 所示。

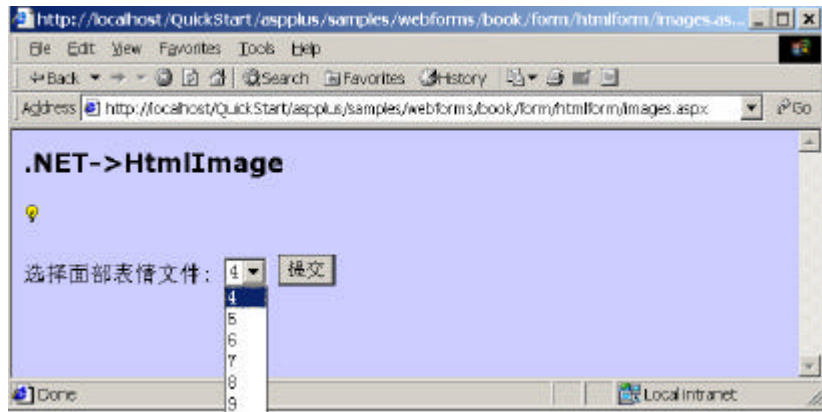


图 7-11

选择相应的文件号，点击按钮，对话框就显示出来。

7.2.4 TextArea

像在 HTML 中的一样，在 ASP.NET 中的 TextArea 也是一个多行输入框。TextArea 的宽度由 Cols 属性决定，长度由 Rows 属性决定。

(csbook\form\htmlform\Textarea.aspx)中定义输入：

```
<textarea id="TextAreal" cols=40 rows=4 runat=server />
```

用 TextAreal.Value 取得输入的值。具体如下：

```
<html>
<head>
  <script language="VB" runat="server">
    Sub SubmitBtn_Click(sender As Object, e As EventArgs)
      Span1.InnerHtml = "下面是你所写的 : <br>" & TextAreal.Value
    End Sub
  </script>
</head>
<body bgcolor="#ccccff">
  <h3><font face="Verdana">.NET->HtmlTextArea</font></h3>
  <form runat=server>
    <font face="Verdana" size="-1">
      写吧: <br>
      <textarea id="TextAreal" cols=40 rows=4 runat=server />
      <input type=submit value="Submit" OnServerClick="SubmitBtn_Click"
runat=server>

    <p>
      <span id="Span1" runat="server" />
```

```

    </font>
  </form>
</body>
</html>

```

程序结果如图 7-12。

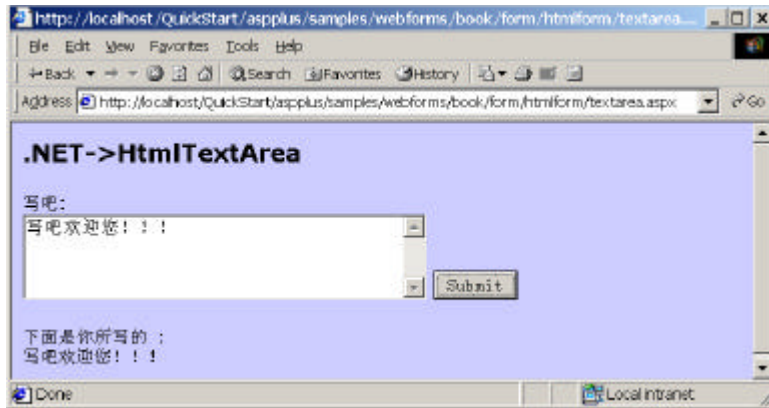


图 7-12

7.2.5 InputHidden

可以用隐藏输入控件来处理一些要传送而又不想在页面上显示出来的信息，例如在电子商务网站中，向银行网关接口传送订单信息，就可以用隐藏输入控件来处理。

下面的例子用不可见的值来取得输入值，再把不可见值显示出来。

Inputhidden.aspx 隐藏输入控件：

```
<input id="HiddenValue" type="hidden" value="隐藏的字符" runat="server">
```

初始值为“隐藏的字符”，在第一次点击按钮时候显示出来，方法如下：

```
Sub SubmitBtn_Click(sender As Object, e As EventArgs)
```

```
    HiddenValue.Value = StringContents.Value
```

```
End Sub
```

这个方法把输入值赋给不可见的控件。完整的代码如下：

```
<html>
```

```
<head>
```

```
  <script language="VB" runat="server">
```

```
    Sub Page_Load(sender As Object, e As EventArgs)
```

```
      If IsPostBack Then
```

```
        Span1.InnerHtml="隐藏值: <b>" & HiddenValue.Value & "</b>"
```

```
      End If
```

```
    End Sub
```

```

        Sub SubmitBtn_Click(sender As Object, e As EventArgs)
            HiddenValue.Value = StringContents.Value
        End Sub
    </script>
</head>
<body bgcolor="#ccccff">
    <h3><font face="Verdana">.NET->HtmlInputHidden</font></h3>
    <form runat=server>
        <input id="HiddenValue" type=hidden value="隐藏的字符" runat=server>
        请输入: <input id="StringContents" type=text size=40 runat=server>
        <p>
        <input type=submit value="确定" OnServerClick="SubmitBtn_Click"
runat=server>
        <p>
        <span id=Span1 runat=server>显示隐藏的字符</span>
    </form>
</body>
</html>

```

输入 InputHidden，便点击按钮，则显示出默认的隐藏值，如图 7-13。

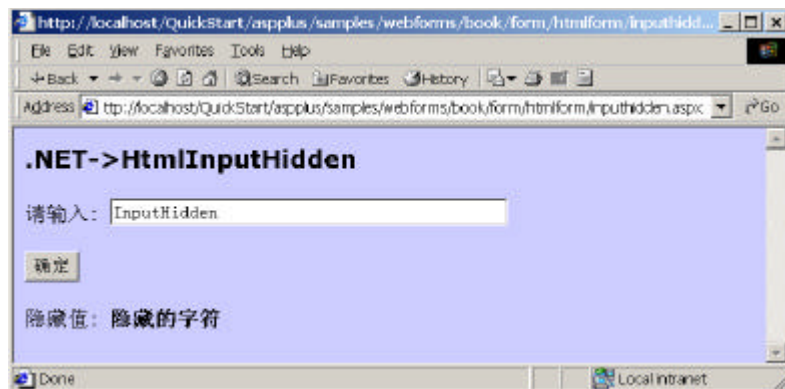


图 7-13

7.2.6 HtmlTable

HtmlTable 服务控件可轻松地创建表格的行和列，也可以按照程序的模式自动生成表格。

下面的例子展示了这个特性：

```

<table id="Table1" CellPadding=4 CellSpacing=0 Border="1" runat="server" />
这就是在 ASP.NET 中，表格的表示。做两个 Select 控件来让用户选择表格的属性：
<p>

```

行:

```
<select id="Select1" runat="server">
  <option Value="1">1</option>
  <option Value="2">2</option>
  <option Value="3">3</option>
  <option Value="4">4</option>
</select>
```


列:

```
<select id="Select2" runat="server">
  <option Value="1">1</option>
  <option Value="2">2</option>
  <option Value="3">3</option>
  <option Value="4">4</option>
</select>
```

在用户提交的时候,实际上是对页面进行了刷新,即在 Page_Load 方法里面处理,具体如下:

```
<html>
<head>
  <script language="VB" runat="server">
    Sub Page_Load(sender As Object, e As EventArgs)
      Dim numRows As Integer
      Dim numcells As Integer
      Dim i As Integer = 0
      Dim j As Integer = 0
      Dim Row As Integer = 0
      Dim r As HtmlTableRow
      Dim c As HtmlTableCell

      ' 产生表格
      numRows = CInt(Select1.Value)
      numcells = CInt(Select2.Value)
      For j = 0 To numRows-1
        r = new HtmlTableRow()
        If (row Mod 2 <> 0) Then
          r.BgColor = "Gainsboro"
        End If
        row += 1
        For i = 0 To numcells-1
          c = new HtmlTableCell()
          c.Controls.Add(new LiteralControl("row " & j & ", cell "
& i))

          r.Cells.Add(c)
        Next i
        Table1.Rows.Add(r)
```

```
        Next j
    End Sub
</script>
</head>
<body bgcolor="#ccccff">
    <h3><font face="Verdana">.NET->HtmlTable</font></h3>
    <form runat=server>
        <font face="Verdana" size="-1">
            <p>
                <table id="Table1" CellPadding=4 CellSpacing=0 Border="1"
runat="server" />
            <p>
                行:
                <select id="Select1" runat="server">
                    <option Value="1">1</option>
                    <option Value="2">2</option>
                    <option Value="3">3</option>
                    <option Value="4">4</option>
                </select>
                <br>
                列:
                <select id="Select2" runat="server">
                    <option Value="1">1</option>
                    <option Value="2">2</option>
                    <option Value="3">3</option>
                    <option Value="4">4</option>
                </select>
                <input type="submit" value="产生表格" runat="server">
            </font>
        </form>
    </body>
</html>
```

运行结果如图 7-14。

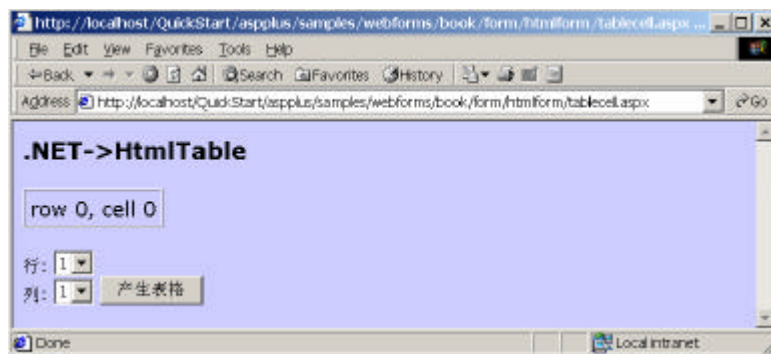


图 7-14

选择并提交，表格就出来了，如图 7-15。

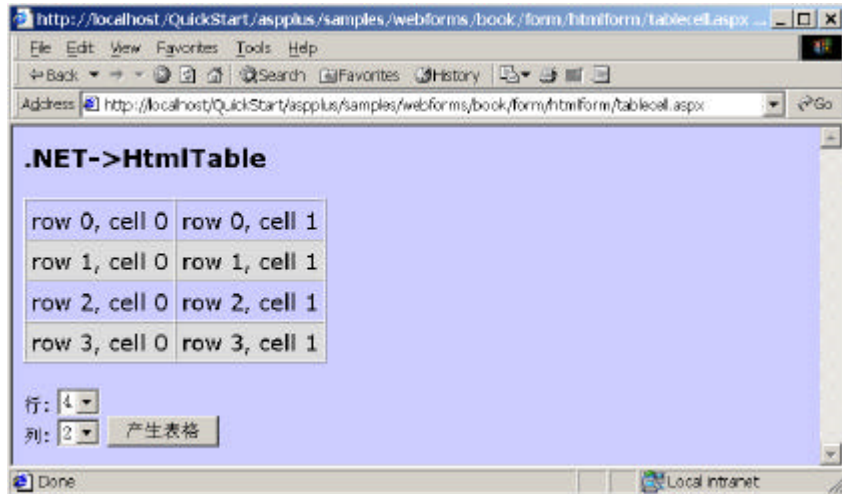


图 7-15

7.2.7 HtmlGenericControl

HtmlGenericControl 提供一个服务器控件，用来执行那些不直接的表现出来的未知的 Html Control 标示，

Gerecolor.aspx：

```
<html>
<head>
  <script language="VB" runat="server">
    Sub SubmitBtn_Click(sender As Object, e As EventArgs)
      Body.Attributes("bgcolor") = ColorSelect.Value
    End Sub
  </script>
</head>
<body id=Body runat=server>
  <h3><font face="Verdana">.NET->HtmlGenericControl</font></h3>
  <form runat=server>
    <p>
      Select a background color for the page: <p>
      <select id="ColorSelect" runat="server">
        <option>White</option>
        <option>Wheat</option>
        <option>Gainsboro</option>
        <option>LemonChiffon</option>
```



```

</select>
  <input type="submit" runat="server" Value="Apply"
OnServerClick="SubmitBtn_Click">
</form>
</body>
</html>

```

运行结果如图 7-16 所示。

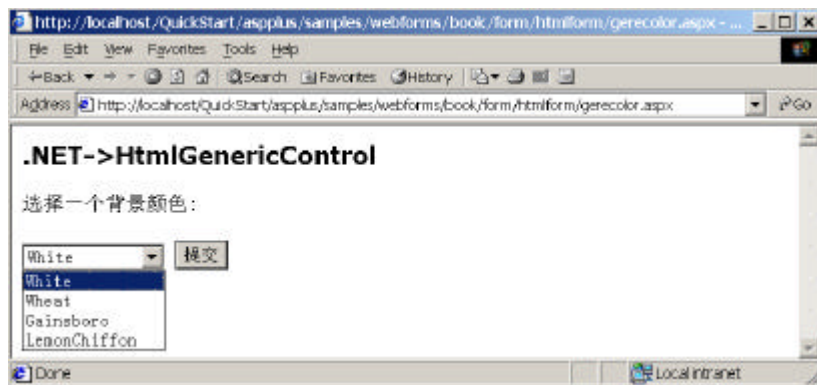


图 7-16

选择你所要的颜色，则页面背景颜色就会改变。

7.2.8 HtmlInputButton

其实在上面的应用的时候就大概的应用过这个控件了，现在专门来介绍它。这个控件有几个功能，可以是普通的按钮来响应一般的事件；可以是 Submit 按钮；也可以是 Reset 按钮。

7.2.8.1 一般性的按钮

这个控件不是响应表单中通常的 Submit 或者 Reset 事件的，而是响应为它定制的事件。

```

<html>
<head>
  <script language="VB" runat="server">
    Sub Button1_Click(sender As Object, e As EventArgs)
      Span1.InnerHtml = "你点击了这个按钮！"
    End Sub
  </script>
</head>
<BODY BGCOLOR="#CCCCCC">
  <h3><font face="Verdana">.NET->HtmlInputButton->button</font></h3>

```



```
        End If
    End Sub
    ' Sub ResetBtn_Click(sender As Object, e As EventArgs)
    '     Name.Value = ""
    '     Password.Value = ""
    ' End Sub

</script>
</head>
<BODY BGCOLOR="#CCCCCC">
    <h3><font face="Verdana">.NET->Submit and Reset</font></h3>
    <form runat=server>
        输入名字: <input id="Name" type=text size=40 runat=server>
        <p>
        输入密码:< input id="Password" type=password size=40 runat=server> (密
码是 : saidy2001)
        <p>
        <input type=submit value="提交" OnServerClick="SubmitBtn_Click"
runat=server>
        <input type=reset value="重写" OnServerClick="ResetBtn_Click"
runat=server>
        <p>
        <span id="Span1" style="color:red" runat=server></span>
    </form>
</body>
</html>
```

看到如图 7-18 的结果。

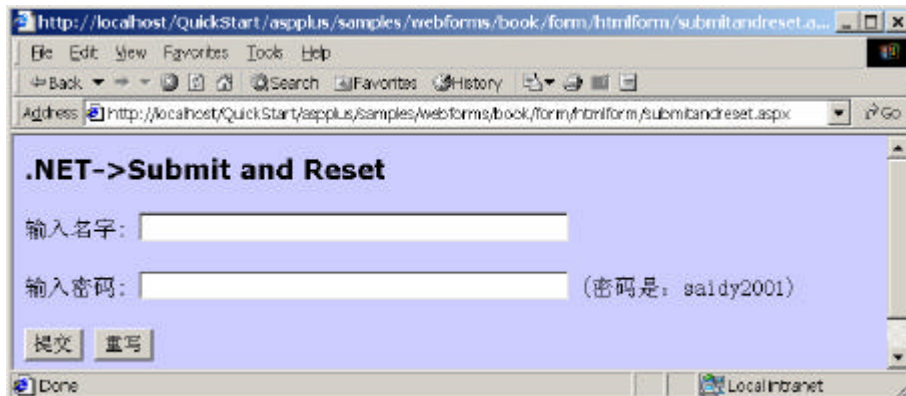


图 7-18

第 8 章 .NET 的数据库编程技术

HOPE



声 明

本电子版不包括本章内容，请看配套图书相关章节。

北京希望电子出版社

2001

第 9 章 应用程序

9.1 什么是应用程序

在 ASP.NET 中，可以这样来定义一个应用程序 Application：能够运行在一个 Web 应用服务器的子目录或者虚拟目录上的所有的文件、页面、操作、模块或者能被执行的代码。比方说，在一个 Web 服务器上，一个“order”应用程序将会在“/order”这个目录下被发布。

每一个在 Web 服务器上的 ASP.NET 应用程序都在一个唯一的应用程序域运行时间里被执行，以保证类的隔离（没有版本、名称上的冲突）、安全屏蔽（防止有权访问某些机器/网络的资源）、静态变量的隔离。

在一个 Web 应用程序的生命周期中，ASP.NET 维护一个 HttpApplication 实例。ASP.NET 对一个 Http 的请求会自动分配一个来处理，这个特别的 HttpApplication 实例对管理这个在全部的生命周期里的请求是可靠的，并且在处理完成后可以唯一的被重用。

在应用程序环境下，ASP.NET 并发处理客户端的请求，所以可能存在多线程对 Application 对象的同时存取。在这种情况下，对 Application 对象的草率处理，可能会导致不可预知的错误。例如以下代码：

```
<% Application("counter") = CType(Application("counter") + 1, Int32) %>
```

原本希望对实例进行计数，但如果同时到达两个以上请求时，则有可能漏计。正确的方法应该是在操作以前，对 Application 对象上锁，操作完成以后，再对 Application 对象解锁。代码如下：

```
<%  
Application.Lock()  
Application("counter") = CType(Application("counter") + 1, Int32)  
Application.Unlock()  
%>
```

9.1.1 配置应用程序的步骤

9.1.1.1 设置应用程序的目录结构

一个 Web 站点可以有多个应用程序运行，而每一个应用程序可以用唯一的 URL 来访问，所以首先应利用 IIS 开放应用程序的目录为“虚拟目录”。各个应用程序的“虚拟目录”可以不存在任何物理上的关系。例如：

应用URL:	物理路径:
http://www.my.com	c:\inetpub\wwwroot
http://www.my.com/myapp	c:\myapp

```
http://www.my.com/myapp/myapp1          \\computer2\test\myapp
```

如果从“虚拟目录”上看来，http://www.my.com/myapp 和 http://www.my.com/myapp/myapp1 似乎存在某种联系，但实际上看到两者完全分布于不同的机器上，更不用说物理目录了。

9.1.1.2 设置相应的配置文件

根据应用的具体需要，可以拷入相应的 global.asax 和 config.Web 配置文件，并且设置相应的选项（配置文件的设置具体见相关章节）。

global.asax 主要配置 application_start、applicatoin_end、session_start、session_end 等事件。

把应用所涉及的各种文件放入“虚拟目录”中。

把.aspx 文件、.ascx 文件以及各种资源文件分门别类放入应用目录中，把类引用所涉及的集合放入应用目录下的 bin 目录中。

9.1.2 应用程序框架

```
<%@ Application attribute="value" [attribute=value ... ]%>
<%@ Import namespace="value" %>...
<%@ Assembly Name="assemblyname" %>

<script language="vb" runsat=server>
...
</script>

<body>
<form runat=server>
...
</form>
</body>

</html>
```

说明：

```
<%@ Application attribute="value" [attribute=value ... ]%>
```

让 ASP.NET 运行环境动态从另一个应用中动态编译出一个类来继承使用。例如：

```
<%@ Application Inherits="MyApp.Object" Description="Ourapp" %>
```

指定应用环境从 Myapp 应用中动态编译一个 MyApp.Object 的类以供使用，它的说明为“ Ourapp”。

```
<%@ Import namespace="value" %>...
```

显示导入一个命名空间到应用程序，这样应用程序就可以使用命名空间中定义的各种

类和接口来完成特定的功能，大大加快了程序的开发速度。例如：

```
<%@ Import Namespace="System.IO" %>
```

```
<%@ Import Namespace="System.NET" %>
```

就可以利用系统提供的大量文件和网络对象，快速的开发文件和网络应用程序。

```
<%@ Assembly Name="assemblyname" %>
```

在页面编译时产生到 assemblyname 的连接，这样就可以使用集合中类及接口。缺省情况下，应用会把应用程序目录下 bin 中的集合都动态载入。该项功能可以在应用程序的 config.Web 中配置，缺省情况下，config.Web 中有如下形式：

```
<assembly>
```

```
<add assembly="*" />
```

```
</assembly>
```

即缺省情况下，加载 bin 下的所有集合。例如：

```
<%@ Assembly Name="myassembly.dll" %>
```

<script>、</script>对之间的代码通常是各种事件的定义，诸如页面开始时、某个事件被触发时所要做的事情。<body>、</body>和<form>、</form>之间通常是页面的界面要素，为显示给客户端的可视界面。

9.1.3 创建应用程序的典型步骤

9.1.3.1 配置 config.Web

主要定义为 gb2312 字符集，以利于中文显示

```
<configuration>
```

```
<globalization requestencoding="gb2312" responseencoding="gb2312" />
```

```
</configuration>
```

9.1.3.2 配置 global.asax

主要定义应用初始化、结束，会话开始、结束，请求开始、结束等事件发生时，应用要做的事情：

```
<script language="C#" runat="server">
```

```
public Application_Start(Object Sender,EventArgs E){
```

```
}
```

```
public Application_End(Object Sender,EventArgs E){
```

```
}
```

```
public Session_Start(Object Sender,EventArgs E){
```

```
}  
  
public Session_End(Object Sender,EventArgs E){  
  
}  
  
public Application_BeginRequest(Object Sender, EventArgs E ){  
  
}  
  
Sub Application_EndRequest(Object Sender, EventArgs E){  
  
}  
  
</script>
```

9.1.3.3 主程序

创建一个应用程序可以先在 Web 服务器上创建一个虚拟目录或者在发布目录下创建一个新的目录。装过 Windows 2000 Advance Server 的读者会知道，安装完成后，会有一个 c:/inetpub/wwwroot 的目录，可以通过 IIS 管理工具来创建一个新的目录或者虚拟目录。

下面创建一个简单的 aspx 页面来说明一个 Application 的应用：

```
<%@Page Language="C#"%>  
<html>  
<body>  
<h1><% Response.Write("Hello World!") %></h1>  
</body>  
</html>
```

下面就是我们的运行结果，如图 9-1 所示。

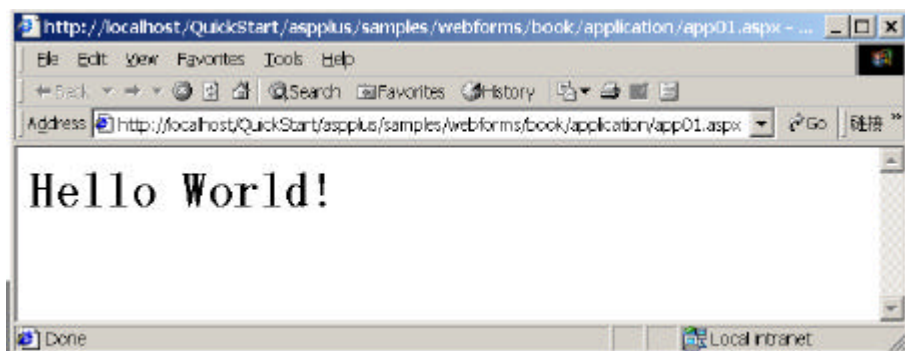


图 9-1

9.1.4 小结

ASP.NET 使一个主机多个地址多个应用成为可能，ASP.NET 的应用概念带来的好处是：程序集中可方便打包，配置的层次结构更加灵活方便，应用独立运行于自身的应用环境中更加安全可靠。

配置一个应用的过程大致为：1) 指定应用目录为 IIS 的虚拟目录，2) 为应用设置适当的配置权限（配置 global.asax 和 config.Web 文件），3) 在自己的应用目录下放置事先编好的程序。

从应用所支持的 Application、Import、Assembly 等命令看来，ASP.NET 对对象重用的支持大大加强了，ASP.NET 的“通用语言运行库”概念的提出，为实现各种开发语言的合作编程奠定了基础。

9.2 配置 Config.Web

9.2.1 ASP.NET 配置简介

ASP.NET 提供了一个丰富而可行的配置系统，以帮助管理人员轻松快速的建立自己的 Web 应用环境。ASP.NET 提供的是一个层次配置架构，可以帮助 Web 应用、站点、机器分别配置自己的扩展配置数据。ASP.NET 的配置系统具有以下优点：

- ✍ ASP.NET 允许配置内容可以和静态内容、动态页面和商业对象放在同一应用的目录结构下。当管理人员需要安装新的 ASP.NET 应用时，只需要将应用目录拷贝到新的机器上即可。
- ✍ ASP.NET 的配置内容以纯文本方式保存，可以以任意标准的文本编辑器、XML 解析器和脚本语言解释、修改配置内容。
- ✍ ASP.NET 提供了扩展配置内容的架构，以支持第三方开发者配置自己的内容。
- ✍ ASP.NET 配置文件的修改被系统自动监控，无须管理人员手工干预。

9.2.2 配置文件的规则

ASP.NET 的配置文件是基于 XML 格式的纯文本文件，存在于应用的各个目录下，统一命名为“config.Web”。它决定了所在目录及其子目录的配置信息，并且子目录下的配置信息覆盖其父目录的配置。

WINNT\Microsoft.NET\Framework\版本号\下的 config.Web 为整个机器的根配置文件，它定义了整个环境下的缺省配置。

缺省情况下，浏览器是不能够直接访问目录下的 config.Web 文件。

在运行状态下，ASP.NET 会根据远程 URL 请求，把访问路径下的各个 config.Web 配置文件叠加，产生一个唯一的配置集合。举例来说，一个对 URL：http://localhost\Webapp\owndir\test.aspx 的访问，ASP.NET 会根据以下顺序来决定最终的配置情况：

- 1 . \Microsoft.NET\Framework\v.1.00\config.Web (缺省配置文件)
- 2 . \Webapp\config.Web (应用的配置)
- 3 . \Webapp\owndir\config.Web (自己的配置)

9.2.3 配置文件的语法规则

9.2.3.1 标识

配置内容被置于 config.Web 文件中的标记<configuration>和</configuration>之间。

格式：

```
<configuration>
  //配置内容
  ...
</configuration>
```

9.2.3.2 配置段句柄说明

ASP.NET 的配置文件架构并未指定任何文件格式或者是支持的配置属性。相反的，它提出了“配置段句柄申明”的概念来支持任意的用户定义配置段。

格式：

```
<configsections>
  <add name=欲定义配置段名 type=处理的句柄函数 />
</configsections>
```

9.2.3.3 配置段

具体定义配置的内容，供应用使用。

以下例子定义了一个“httpmodules”配置段，设置了系统 http 相关的处理模块

```
<configuration>

  <configsections>
    <add name="httpmodules"
type="System.Web.Configuration.HttpModules
ConfigurationHandler" />
  </configsections>

  <httpmodules>
    <add type="System.Web.SessionState.CookielessSessionModule" />
    <add type="System.Web.Caching.OutputCacheModule" />
    <add type="System.Web.SessionState.SessionStateModule" />
```

```

    <add type="System.Web.Security.WindowsAuthenticationModule" />
    <add type="System.Web.Security.CookieAuthenticationModule" />
    <add type="System.Web.Security.PassportAuthenticationModule" />
    <add type="System.Web.Security.CustomAuthenticationModule" />
    <add type="System.Web.Security.UrlAuthorizationModule" />
    <add type="System.Web.Security.FileAuthorizationModule" />
  </httpmodules>

```

```
</configuration>
```

9.2.4 ASP.NET 定义的标准配置段

1. httpmodule 段：定义了应用的 http 请求的处理模块以及诸如安全、日志之类的应用方式
2. httphandlers 段：负责映射 URLs 到 IHttpHandler 类
3. sessionstat 段：负责配置 http 模块的会话状态
4. globalization 段：配置应用的公用设置
5. compilation 段：配置 ASP.NET 的编译环境
6. trace 段：配置 ASP.NET 的跟踪服务
7. security 段：ASP.NET 的安全配置
8. iisprocessmodel：在 IIS 上配置 ASP.NET 的处理模式
9. browercaps 段：配置浏览器的兼容部件

9.2.5 一个配置读出的例子

9.2.5.1 config.Web 配置文件

```

<!--config.Web 请放入FormCfg.aspx所在目录-->
<configuration>
<!--申明一个test配置段-->
  <configsections>
    <add name="test"
type="System.Web.Configuration.DictionarySectionHandler" />
  </configsections>

  <test>
<!--配置一个键key,其内容为just a configure test-->

    <add key="key" value="just a configure test" />
  </test>

</configuration>

```

9.2.5.2 读出其内容

```
<!--文件名:FormCfg.aspx-->
<html>
  <head>
    <script language="C#" runat=server>
public Page_Load(Object s,EventArgs e){
//取出test配置段的key键的值
    Hashtable CfgSection = Context.GetConfig("test");
    String Msg = CStr(CfgSection("key"));
    lblMsg.text=Msg;
}
    </script>
    <title>
配置信息的读取
    </title>
  </head>
  <body>
    <center>
config.web中"test"配置段中key的内容为:
    <asp:label id=lblmsg runat=server />
    </center>
  </body>
</html>
```

9.2.5.3 运行结果

运行结果如图 9-2 所示。

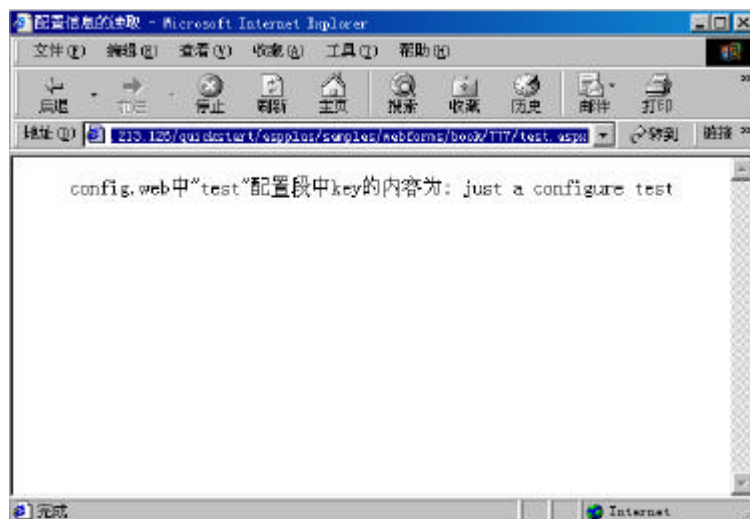


图 9-2

9.2.6 Config.Web 配置实例

```
<configuration>
<!--定义用户应用的公用设置，如SQL的sql连接串等等-->
<appsettings>
</appsettings>

<!--设置浏览器的兼容性部件-->
<browsercaps>
</browsercaps>

<!--编译环境设置，非调试模式-->
<compilation debugmode="false">
<!--缺省编译语言为vb，以后可以不再在Page中定义脚本语言-->
<compilers defaultlanguage="vb">
<!--以MSVSA.dll编译.vb为后缀的VB文件-->
<compiler language="VB" extension=".vb"
type="MSVSA.dll#Microsoft.VB.Compiler"/>
</compilers>

<assemblies>
<!--加入对System.Data的引用-->
<add assembly="System.Data" />
<!--去掉对System.Data的引用-->
<remove assembly="System.IO" />
<!--去掉config.Web中包含或继承来的引用-->
<clear />
</assemblies>

</compilation>

<!--设置应用全局环境-->
<!--文件、请求、返回以gb2312编码，以保证浏览器正确显示中文-->
<globalization fileencoding="gb2312" requestencoding="gb2312"
responseencoding="gb2312"/>

<!--定义用户出错的处理-->
<!--出错缺省显示defaultredirect指定的页面，mode为on时，遵循customerrors配置段-->
<!--mode为off时，忽略用户出错，mode为remoteonly时，本地才显示真正的出错原因-->
<customerrors defaultredirect="AnErrorHasOccured.aspx?ErrNum=-1"
mode="remote">
<!--当出错码为500时，显示redirect指定的页面-->
```

```
<error statusCode="500" redirect="AnErrorHasOccured.aspx?ErrNum=500" />
</customerrors>

<!--指定目录Webapp的访问权限-->
<location path="Webapp" >
<!--非授权用户不能进入Webapp目录-->
<security>
<authorization>
<deny users="*" />
</authorization>
</security>
</location>

<!--定义安全属性-->
<security>
<authorization>
<!--角色为Adminstrators和所有的用户访问其指定的资源-->
<allow roles="Adminstrators" />
<allow users="*" />
</authorization>
</security>

</configuration>
```

9.3 编写和配置 global.asax 文件

为了编写用户界面的应用程序，开发者可以把应用程序标准的逻辑和时间处理的代码加到 Web Application 里面。这些代码不产生用户界面，也不响应单个页面的请求。事实上，这些代码处理更高水平的事件，如 Application_Start, Application_End, Session_Start, Session_End 等等。开发者通过放在 Web 应用程序根目录下面的 Global.asax 来响应这些事件。

ASP.NET 通过一个动态的 .NET Framework 类自动解析和编译这个文件，这个类就是 HttpApplication 基类，在第一时间里面，在这个文件里面的应用程序的资源将会被响应。

首先，在包含有请求的应用程序名字空间中被访问之前，Global.asax 将被解析和编译成 .NET Framework 的一个类。这个文件本身有拒绝被访问的配置。

下面看看这个文件里面的具体内容，首先声明这个文件的使用语言、运行环境：

```
<script language="C#" runat=server>
//相关方法
</script>
```

然后就可以定义各种方法了：

```
public void Application_Start(){
//方法的属性等
}
```

如果事件处理代码需要用到名字空间，可以这样来引用它：

```
<%@ Import Namespace="System.Data.SQL"%>
```

下面来看看这个文件的具体应用，首先在 Web Server 上建立一个 Global.asax 文件，在里面加上下面的代码：

```
<script language="C#" runat=server>
//相关方法

public void Application_Start()
//方法的属性等
}

public void Application_Start(Object Sender, EventArgs E){
Application.Lock();
Application("counter") = CType(Application("counter") + 1, Int32);
Application.Unlock();
}

public void Application_End(Object Sender, EventArgs E){
//Clean up application resources here
}

public void Session_Start(Object Sender, EventArgs E){
Response.Write("Session 正在启动...<br>");
}

public void Session_End(Object Sender, EventArgs E){
//Clean up session resources here
}
</script>
```

当然，还要配置 Config.Web，用来指定出错信息的打印页面。根据上面配置 Config.Web 的经验，很容易的就可以对这个文件进行配置：

```
<configuration>
<customerrors mode="on" defaultredirect="error.htm" />
<globalization requestencoding="gb2312" responseencoding="gb2312" />
</configuration>
```

第二句话就是配置指定的出错页面语句。写两个页面来实现它，一个为出错页面，一个为实现这个功能的 aspx 页面。出错页面很简单的，就是报告程序出错时显示的信息，就

写“在 config.Web 里面配置的连接！”，是经过 aspx 页面甩出来的。

在 aspx 页面，用下面的语句来响应出错按钮点击事件：

```
public void Error_Click(Object Sender, EventArgs E){
    //甩出异常！
    throw new Exception();
}
```

里外响应 Session 的方法用下面的语句来说明：

```
public void Session_Click(Object Sender, EventArgs E){
    Session.Abandon();
    Response.Redirect("global.aspx");
}
```

下面是完整的代码：

```
<html>
  <script language="C#" runat="server">
    '页面导入
    public void Page_Load(Object Sender, EventArgs E){
        Response.Write("正在装入页面..."+"<br>");
    }

    //Session事件
    public void ssaidy(Object Sender, EventArgs E){
        Session.Abandon();
        Response.Redirect("global.aspx");
    }
    //响应错误方法
    public void esaidy(Object Sender, EventArgs E){
        //抛出异常
        throw New Exception();
    }
  </script>
  <body>
    <br><br><br>
    <center>
      <form runat="server">
        <input type="submit" Value="刷新这个页面" runat="server" />
        <input type="submit" OnServerClick="ssaidy" Value="结束这个Session"
runat="server" />
        <input type="submit" OnServerClick="esaidy" Value="错误表示"
runat="server" /><p>
      <hr>
    </form>
```



```
</center>  
<br><br>  
</body>  
</html>
```

运行结果如图 9-3 所示。

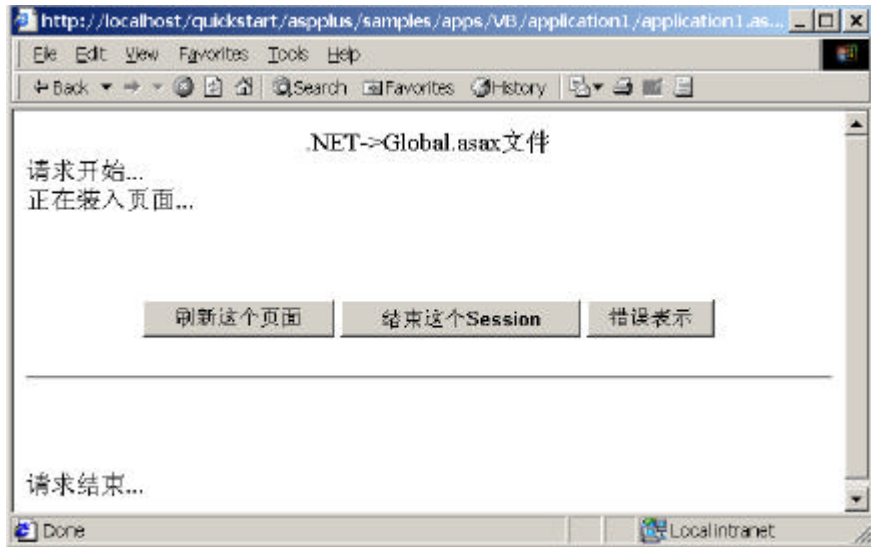


图 9-3

点击“错误表示”按钮，显示如图 9-4 所示。

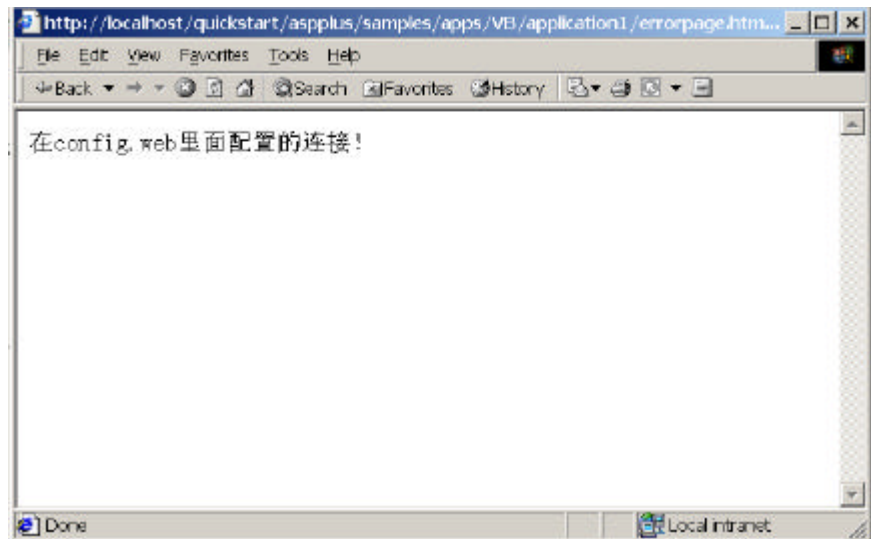


图 9-4

9.4 Application 和 Session

9.4.1 Application 对象

在介绍 ASP.NET 的 Application 对象之前，先来回顾一下 ASP 的 Application 对象。由于变量的生命周期受限于网页，所以每当 .asp 文件被解释执行完毕之后时，变量的内容会不存在，为了把变量的内容记录下来，最笨的方法是使用文件，比较简单的方法就是使用 Application 对象。通过 Application 对象，来记录变量的值。

9.4.1.1 使用 Application 对象的基础

如果希望在 .aspx 文件被执行后，还能够将变量（对象）的内容记录下来，可以利用以下的语句将变量的内容存储在 Application 对象中：

```
Application ("变量名称")=对象名称
```

```
Set Application ("对象名称")=对象名称
```

当然也可以直接把 Application (“变量名称”)当成变量来使用，例如：

```
Application ("counter") = Application ("counter") +1
```

Application 对象还有另一个重要的特性，假设 a.aspx 使用了 Application 对象，而 b.aspx 也使用了 Application 对象，而且这两个 aspx 程序都有 Application (“counter”) = Application (“counter”) +1 语句，如果 a.aspx 执行了这一条语句，Application (“counter”) =1 此时如果 b.aspx 也执行了这条语句，则 Application (“counter”) =2。

9.4.1.2 制作计数器

Application 对象对不同的连接者是共用的，因此适合制作计数器。程序代码如下：

```
<%  
Application.Lock()  
Application("counter") = CType(Application("counter") + 1, Int32)  
Application.Unlock()  
%>  
<html>  
你是第<%=Application("counter")%>位访问者！  
</html>
```

结果如图 9-5。



图 9-5

9.4.1.3 计数器非同步更新的问题

假设有两个上网者同时启动网页（这种情况很常见），同时执行了上面的步骤，他们读出的值是相同的，但结果都是相同的。很明显，访问量少了一次。如何处理呢？我们在这里举一个例子，假设我们都在使用一个数据库管理系统，有可能访问者同时对一张表进行操作，如果一人在修改这张表的数据，而另一个人在读数据，很明显此时数据前后就不一致。通过我们需要对表进行锁定。那么，我们可以锁定对象，程序修改为：

```
Application.lock  
Application("counter") = CType(Application("counter") + 1, Int32)  
Application.unlock
```

这样，当某一程序执行了 `Application.lock` 之后，其他应用程序就暂时不能使用 `Application` 对象，直到 `Application unlock`。

9.4.1.4 Application 的事件

当 `Application` 对象的生命周期开始时，`Application_onstart` 事件会被启动，当 `Application` 对象的生命周期结束时 `Application_onend` 事件会被启动。通常会在 `global.asax` 中定义 `Application_onstart` 事件。这和 ASP 程序相类似，有一点差别是 `Application_onstart` 事件是在 `global.asa` 中定义。还有就是除了以前的四个事件又增加了两个事件 `Application_BeginRequest` 事件和 `Application_EndRequest` 事件。

在上一个例子中我们实现了用计数器来对页面进行统计，但是这样的程序有这样的一个问题，就是只能统计单个的页面，我们在 `asp+` 中可以很轻松的实现对整个站点页面的统计。

为了达到这个目的，我们将使用 Application_BeginRequest 事件和 Application_EndRequest 事件。这两个事件在站点的任意一个文件被请求的时候都会被激发，因此我们便利用这个事件实现对站点的访问统计。

首先来看看这个 global.asax 文件。

```
<script language="VB" runat="server">
Sub Application_End(Sender As Object, E As EventArgs)
'我们捎带实现了站点的当前在线人数
dim intOnlineNumber as integer
intOnlineNumber=cInt(Application("ONLINENUMBER"))-1
Application("ONLINENUMBER")=intOnlineNumber
End Sub

Sub Session_Start(Sender As Object, E As EventArgs)
Application.Lock
intOnlineNumber=cInt(Application("ONLINENUMBER"))+1
Application("ONLINENUMBER")=intOnlineNumber+1
Application.Unlock
End Sub

Sub Application_BeginRequest(Sender As Object, E As EventArgs)
response.write("当前访问的页面是 " + Request.FilePath + "<br>")
'既然可以得到FilePath 则我们只要把这个参数进行详细的各种各样的统计就可以了
End Sub
</script>
'好了一切完结之后，我们访问站点的任意一个aspx 文件，都会在最上方发现一行文字：当前访问的页面是.....
```

怎么样，还不赶快尝试一下？

9.4.2 Session

9.4.2.1 ASP.NET 里的 Session

对 Session 的应用，我们通过在 Global.asax 中设置，然后在 aspx 页面中调用来进行。下面就一个例子来说明这个问题。

9.4.2.2 一个 Session 例子

我们在 Global.asax 中加上一个 Session_Start 方法，在里面定义响应的属性：

```
Sub Session_Start(Sender As Object, E As EventArgs)
    Session("name") = "saidy"
    Session("email") = "saidychan@sina.com"
    Session("tel") = "130000121553"
End Sub
```

赋予属性一个初始值，这样当在调用页面（session.aspx）装入时直接就用这些默认值：

```
<script language="VB" runat="server">
    '返回Session值方法：
    Function getinfo(Key As String) As String
        Return Session(Key).ToString()
    End Function
</script>
```

注意 Key 的定义，我们用这个语句获得方法的具体返回值，如“name”值：

```
<%=getinfo("name")%>
```

在另外一个文件（info.aspx）中，我们用下面的语句来获得数据：

```
Sub sc(Sender As Object, E As EventArgs)
    Session("name") = name.Value
    Session("email") = email.Value
    Session("tel") = tel.Value
    Response.Redirect(State("Referer").ToString())
End Sub
```

保存在 Session 中。

下面是具体的代码：

Global.asax文件：

```
<script language="VB" runat="server">

    Sub Session_Start(Sender As Object, E As EventArgs)

        Session("name") = "global_saidy"
        Session("email") = "global_saidychan@sina.com"
        Session("tel") = "global_130000121553"
    End Sub

</script>
```

session.aspx文件：

```
<html>
    <script language="VB" runat="server">
        '返回Session值方法：
        Function getinfo(Key As String) As String
            Return Session(Key).ToString()
        End Function
    </script>

    <body bgcolor="#ccccff">
        <br><br><br>
```

```
<center>
  <h3><font face="Verdana">.NET->Session</font></h3>
</center>
<br><br>
  <center>
<b><a href="info.aspx">员工档案! </a></b><p>
  <div align="center">
<center>
<table border="0" width="35%" cellspacing="0" cellpadding="0">
  <tr>
    <td width="50%">名称: </td>
    <td width="50%"><%=getinfo("name")%></td>
  </tr>
  <tr>
    <td width="50%">邮箱: </td>
    <td width="50%"><%=getinfo("email")%></td>
  </tr>
  <tr>
    <td width="50%">电话: </td>
    <td width="50%"><%=getinfo("tel")%></td>
  </tr>
</table>
</center>
</div>
</center>
</body>
</html>
```

info.aspx文件:

```
<html>
  <script language="VB" runat="server">
    '取得上一页的信息
    Sub Page_Load(Sender As Object, E As EventArgs)

      If Not (Page.IsPostBack)
        State("Referer") = Request.Headers("Referer")
      End If
    End Sub

    '按钮事件,把数据保存在Session中,并返回上一页
    Sub sc(Sender As Object, E As EventArgs)

      Session("name") = name.Value
```

```
Session("email") = email.Value
Session("tel") = tel.Value
Response.Redirect(State("Referer").ToString())
End Sub
```

·按钮事件, 返回上一页

```
Sub cc(Sender As Object, E As EventArgs)
```

```
Response.Redirect(State("Referer").ToString())
```

```
End Sub
```

```
</script>
```

```
<body bgcolor="#ccccfc">
<br><br><br>
<center>
<form runat="server">
<h3><font face="Verdana">.NET->Session!</font></h3>
</center>
<br><br>
<center>
<b>选择你的信息: </b><p>
<table bgcolor="#ccccff">
<tr>
<td>名称:</td>
<td>
<select id="name" runat="server">
<option>saidy chan</option>
<option>贺禧</option>
<option>吕兵</option>
</select>
</td>
</tr>
<tr>
<td>邮箱:</td>
<td>
<select id="email" runat="server">
<option>saidychan@sina.com</option>
<option>chenyx@staff.yesky.com</option>
<option>hexil224@sina.com</option>
</select>
</td>
</tr>
</table>
```

```
<tr>
  <td>电话:</td>
  <td>
    <select id="tel" runat="server">
      <option>023-77398202</option>
      <option>013011235488</option>
      <option>1398855566588</option>
    </select>
  </td>
</tr>
</table>
<p>
  <input type="submit" OnServerClick="cc" Value="取消" runat="server"/>
  <input type="submit" OnServerClick="sc" Value="提交" runat="server"/>
</form>
</center>
</body>
</html>
```

程序运行结果如图 9-6。

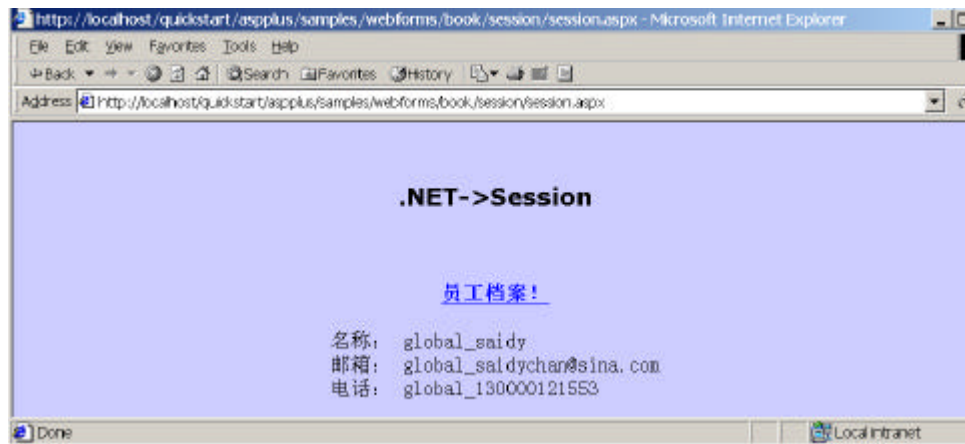


图 9-6

我们看到页面把储存在 Global.asax 文件中的数据取出来了，点击“员工档案”连接，如图 9-7 所示。

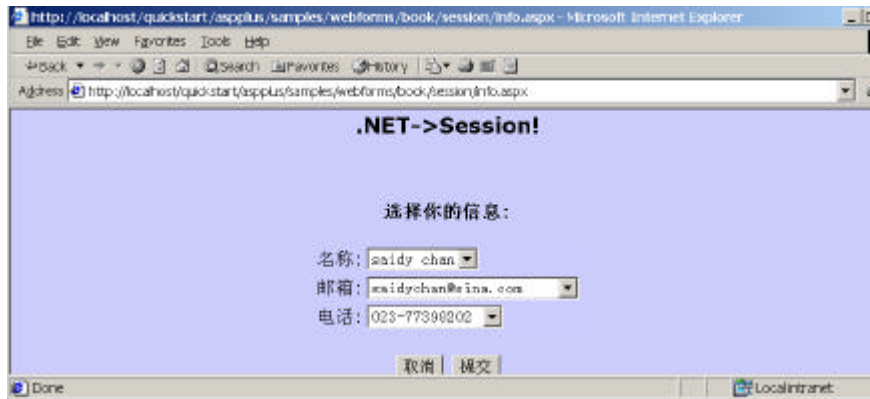


图 9-7

点击“提交”按钮，看到图 9-8 的结果所示。

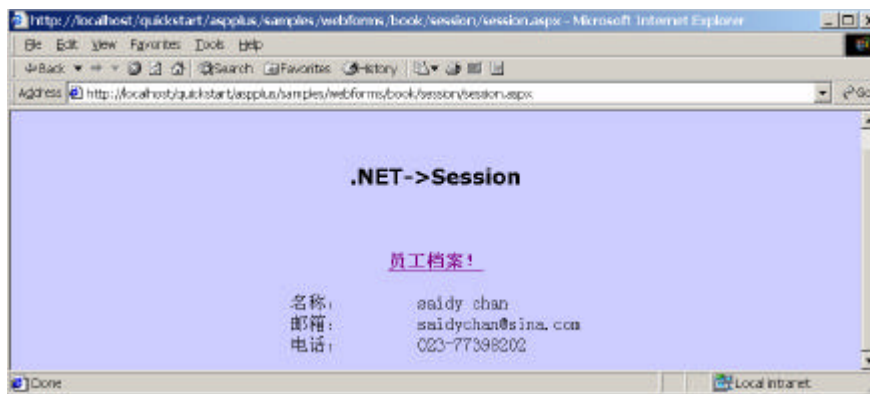


图 9-8

我们看到提交的结果是 Session 中保存的数据被替代了。

9.5 安全访问控制

9.5.1 ASP.NET 提供了两种机制来保护资源和代码

1. 代码访问安全机制：使用许可协议（permissions）来控制访问的代码要保护资源。它可以防止计算机系统遭到恶意代码的袭击，同时通过使用一种方法来让这些代码安全运行。

2. 用户访问安全机制：它可以为用户建立各种角色，这样就可以限制用户的操作。

对于一个 Web 应用程序来说，很重要的一点是能够辨别访问者的角色和对资源访问的限制。因为我们不排除人们对数据的破坏和对系统的破坏。为了做到这一点，我们就要

对其身份进行验证，因此我们就会看到在应用系统中，使用用户名和密码，对其使用者身份的鉴定以及访问内容的限制。因此我们可以说鉴定(authentication)和授权(authorization)是密不可分，缺一不可。就 ASP.NET 来说，它和 IIS 协同工作，对 Web 应用程序提供鉴定和授权的服务。

对于 Web 应用程序来说，还有一个重要的特点是能够控制服务器端的代码被执行。当一个服务器端的应用程序能够按照访问者的角色的不同执行相应的代码时，这被称为扮演 (impersonation)。就 ASP.NET 来说，能够随意的按照访问者角色的不同来执行相应的代码。有了上面的叙述后，ASP.NET 的鉴定体系也就详细的说明。

9.6 鉴定和授权

ASP.NET 使用 Windows 的一套鉴定机制,和 IIS 一起支持鉴定。ASP.NET 支持 Windows 的认证 (passport) 鉴定机制。有一点需要说明的是 ASP.NET 的鉴定服务需要依赖于 IIS 提供的服务。举个例子，在一个基于 IIS 的应用程序需要使用一个基本的鉴定，我们就必须使用 Internet 服务工具来配置。

为了使 ASP.NET 的认证机制有效，我们需要对<authentication>进行配置。需要说明的是，<authentication>必须包含在<security>和</security>中，那么有哪些鉴定模式呢？如下表 9-1 所示。

表 9-1

模式	描述
None	没有任何 ASP.NET 的鉴定服务被激活。
Windows	ASP.NET 和 Windows 的鉴定机制一起配合使用，可以授权限制 Windows NT 的用户和工作组的访问。
Cookie	ASP.NET 鉴定服务可以管理 cookies，可以引导使一个未经授权的用户去登录网页。这种模式经常和 IIS 一起配合，可以让一个匿名用户去访问应用程序。
Passport	ASP.NET 鉴定服务通过使用护照 (passport) 软件包，附属在服务的外层，更加强了鉴定服务，这种软件包必须安装在机器里才能生效。

举例说明，下面的配置文件对于一个应用程序来说，使用的 Cookie 鉴定模式：

```
<configuration>
  <security>
    <authentication mode="Cookie" />
  </security>
</configuration>
```

Windows-based 鉴定

当我们使用 ASP.NET 的 Windows 鉴定模式的时候，对于当前的请求，ASP.NET 使用

了 WindowsPrincipal 对象。当我们在授权的时候，这个对象被 URL 授权使用。通过这个对象，还可以在一个应用程序中判断请求者是否是我们被授权的对象。举个例子说明：

```
If User.IsInRole("Administrators") Then
    DisplayPrivilegedContent()
End If
```

在这个例子中，使用了一个判断，指定如果请求者是以管理员身份登录的话，将会执行什么操作。

Forms-based 鉴定

Forms-based 鉴定，有点类似我们在使用数据库管理系统时，第一步需要用户登录，系统通过判断用户的权限，来决定应该执行的操作。通过 Forms-based 鉴定，系统可以引导用户登录对应的网页。为了使用这种鉴定，需要在 config.Web 文件中进行配置。

下面的这个例子是要求使用 cookie 鉴定模式，同时拒绝访问者以匿名的形式访问。

```
<configuration>
  <security>
    <authentication mode="Cookie" />
    <authorization>
      <deny users="?" />
    </authorization>
  </security>
</configuration>
```

授权用户和角色

The ASP.NET URL 可以很容易的授权是否用户能访问某一资源。下面的例子授权了姓名是 TOM，他的角色是管理员身份，而其它的用户被拒绝访问。

```
<security>
  <authorization>
    <allow users="TOM" />
    <allow roles="Admins" />
    <deny users="*" />
  </authorization>
</security>
```

我们要授权某人的话，使用 allow，如果要拒绝的话，则使用 deny。

如果要授权多个用户，或者拒绝多个用户的话，用户名之间用逗号（“，”）隔开。如：

```
<allow users="Tom,Jack" />
```

也可以为用户定义 http 方法（get/post）。如下：

```
<allow VERB="GET" users="*" />
<deny VERB="POST" users="*" />
```

也可以使用一些特殊的字符来表示用户。在 ASP.NET 中，使用“？”来表示匿名用户，使用“*”来表示所有的用户。如：

```
<authorization>
```

```
<deny users="?" />
</authorization>
```

表示应用程序拒绝用户以匿名身份登录。

下面将介绍一个完整的例子：

default.aspx 文件：

```
@ Import Namespace="System.Web.Security " %>

<html>

  <script language="VB" runat=server>

    Sub Page_Load(Src As Object, E As EventArgs)
      Welcome.Text = "Hello, " + User.Identity.Name
    End Sub

    Sub Signout_Click(Src As Object, E As EventArgs)
      CookieAuthentication.SignOut()
      Response.Redirect("login.aspx")
    End Sub

  </script>

  <body>

    <h3><font face="Verdana">Using Cookie Authentication</font></h3>

    <form runat=server>

      <h3><asp:label id="Welcome" runat=server/></h3>

      <asp:button text="Signout" OnClick="Signout_Click" runat=server/>

    </form>

  </body>

</html>
```

login.aspx文件：

```
<%@ Import Namespace="System.Web.Security " %>

<html>
```

```
<script language="VB" runat=server>

    Sub Login_Click(Src As Object, E As EventArgs)
        If (UserEmail.Value = "jdoe@somewhere.com" Or UserEmail.Value =
"mary@somewhere.com") And UserPass.Value = "password"
            CookieAuthentication.RedirectFromLoginPage(UserEmail.Value,
PersistCookie.Checked)
        Else
            Msg.Text = "Invalid Credentials: Please try again"
        End If
    End Sub

</script>

<body>

    <form runat=server>

        <h3><font face="Verdana">Login Page</font></h3>

        <table>
            <tr>
                <td>Email:</td>
                <td><input id="UserEmail" type="text" runat=server/></td>
                <td><ASP:RequiredFieldValidator ControlToValidate="UserEmail"
Display="Static" ErrorMessage="*" runat=server/></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input id="UserPass" type=password runat=server/></td>
                <td><ASP:RequiredFieldValidator ControlToValidate="UserPass"
Display="Static" ErrorMessage="*" runat=server/></td>
            </tr>
            <tr>
                <td>Persistent Cookie:</td>
                <td><ASP:CheckBox id=PersistCookie runat="server" /> </td>
                <td></td>
            </tr>
        </table>

        <asp:button text="Login" OnClick="Login_Click" runat=server/>
```

```
<p>

    <asp:Label id="Msg" ForeColor="red" Font-Name="Verdana"
Font-Size="10" runat=server />

</form>
</body>

</html>
config.Web 文件：
<configuration>

    <security>

        <authentication mode="Cookie">
            <cookie decryptionkey = "autogenerate" loginurl = "login.aspx" cookie
= ".ASPXUSERDEMO" />
        </authentication>
        <authorization>
            <deny users="jdoe@somewhere.com" />
            <deny users="?" />
        </authorization>
    </security>
    <globalization requestencoding="UTF-8" responseencoding="UTF-8" />
</configuration>
```

这个例子是一个比较典型的例子，通过对 config.Web 文件的设置，设置了拥护权限。

第 10 章 Web Service

10.1 Web Service 简介

Web Service 是微软.net 策略计划的基础,按微软的说法,一个 Web Service 就是一个应用 Web 协议的可编程的应用程序逻辑。这些协议的其中的一个就是简单对象访问协议 SOAP(Simple Object Access Protocol)。

Internet 在不断的发展,刚开始的时候浏览的网页只是静态的,是不可交互的。而现在随着技术的日益发展,将提供给网页浏览者一个可编程的 Web 站点。这些站点将通过一些应用软件直接连接到另一个 Web 站点,这些可编程的 Web 站点较传统的 Web 站点来说,将变得更加能重复使用,也更加智能化!

.NET 平台提供了一种运行环境,即公用语言运行环境 (CLR, Common Language Runtime)。对 CLR 来说,它提供了一种内置机制来创建一个可编程的站点,对于 Web 程序开发者和 VB 程序员来说,是一致、熟悉的。这种模型是可以重复使用,也可以再扩展。它包含了开放的 Internet 标准 (HTTP, XML, SOAP, SDL)。以便它被网页浏览者访问。

ASP.NET 使用 .asmx 文件来对 Web Services 的支持。.asmx 文件和 .aspx 文件一样都属于文本文件。它包含在 .aspx 文件之中,成为 ASP.NET 应用程序的一部分。

下面用简单的例子来介绍 .asmx 文件,还是从“Hello, World”这个经典的范例说起,代码如下:

```
<%@ WebService Language="C#" Class="HelloWorld" %>
    using System.Web.Services;
    public class HelloWorld : Inherits WebService{
        [WebMethod] public string SayHelloWorld( ){
            Return("Hello World");
        }
    }
```

为了使这个 .asmx 文件有效,需要把这个文件放置在一个服务器上运行。假设有一个服务器名为 server1,有一个虚拟目录为 aaa 的机器。在 IE 的地址栏上输入 <http://server1/aaa/helloworld.asmx>,程序执行后,将显示在类 helloworld 中可调用的方法。同时可以使用如 SOAP 或者 HTTP GET 协议来调用类 helloworld 下面的方法。当键入 <http://server1/iceberg/HelloWorld.asmx?SDL>,可以看到作为 XML 文件的输出。这些依赖于服务器描述语言的语法 (Service Description Language (SDL) grammar)。SDL 文件很重要,它能在客户端访问服务器时使用。

看一下程序运行的结果,如图 10-1 所示。



图 10-1

现在对这一文件进行说明：在文件中，直接指出使用 Web Service，然后导入 System.Web.Services 名字空间（name space）。这个名字空间是必须的，必须使用 Imports System.Web.Services 这条语句。紧接着声明 Hello World 类。它继承了基类 WebService。这样 WebService 的方法可以被使用。然后定义了一个属性 [WebMethod]（或 <"WebMethod">）。

如果这些代码作为客户端的角度来看的话，将会很简单：

```
HelloWorld myHelloWorld = new HelloWorld();
string sReturn = myHelloWorld.SayHelloWorld();
在客户端将会显示 hello, world。
```

10.2 一个简单的 Web Service 案例

这个例子中将定义一个 mathService 类，来对两个数字分别进行加，减，乘，除。当然这个类需要从基类 Web Service 中继承。请先看该程序的源代码：

```
<%@ WebService Language="C#" Class="MathService" %>

using System;
using System.Web.Services;

public class MathService {

    [WebMethod]
    public int Add(int a, int b)
```



```
{  
    return a + b;  
}  
  
[WebMethod]  
public int Subtract(int a, int b)  
{  
    return a - b;  
}  
  
[WebMethod]  
public int Multiply(int a, int b)  
{  
    return a * b;  
}  
  
[WebMethod]  
public int Divide(int a, int b)  
{  
    if (b==0) return -1;  
    return a / b;  
}  
}
```

对于该程序，首先要用`<%@ WebService Language="C#" Class="MathService" %>`来标识。然后定义对应的方法。程序很简单，来看一下运行结果，假使使用“加”。要计算 5+1 的值，如图 10-2、图 10-3 所示。

Invoke the Add Web Method:

Enter parameter values and then click the 'Invoke' button to invoke the Add web method.

Parameter	Value
A:	<input type="text" value="5"/>
B:	<input type="text" value="1"/>

图 10-2

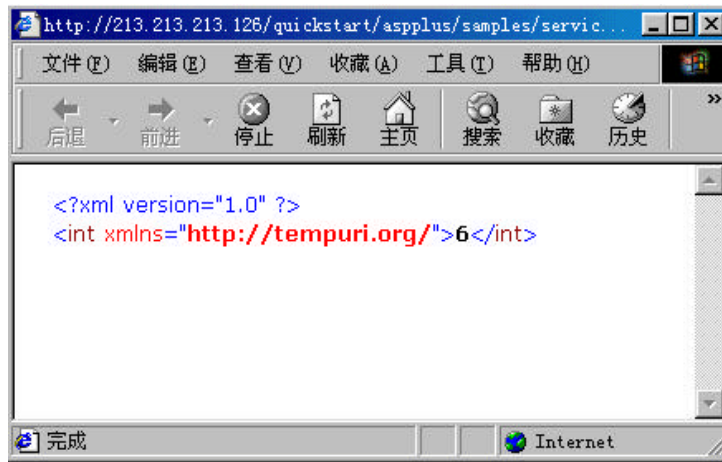


图 10-3

通过上面的计算，结果就显示出来了。

一个代理类（proxy class）被创建出来，我们就可以很容易的创建对象。当然，类中的方法就能够被对象调用。创建代理类，可以使用 WebServiceUtil.exe 来创建一个代理类。还是举“加”，“减”，“乘”，“除”的例子。

先创建一个文件用于客户端的用户浏览：

```
<%@ Import Namespace="MathServiceSpace" %>
<html>
<script language="C#" runat="server">
    int operand1 = 0;
    int operand2 = 0;
    public void Submit_Click(Object sender, EventArgs E)
    {
        try
        {
            operand1 = Int32.Parse(Operand1.Text);
            operand2 = Int32.Parse(Operand2.Text);
        }
        catch (Exception) { /* ignored */ }
        MathService Service = new MathService();
        switch (((Control)sender).ID)
        {
            case "Add" : Result.Text = "<b>Result</b> = " +
Service.Add(operand1, operand2).ToString(); break;
            case "Subtract" : Result.Text = "<b>Result</b> = " +
Service.Subtract(operand1, operand2).ToString(); break;
            case "Multiply" : Result.Text = "<b>Result</b> = " +
```

```

Service.Multiply(operand1, operand2).ToString(); break;
        case "Divide" : Result.Text = "<b>Result</b> = " +
Service.Divide(operand1, operand2).ToString(); break;
    }
}
</script>
<body style="font: 10pt verdana">
    <h4>Using a Simple Math Service </h4>
    <form runat="server">
        <div
style="padding:15,15,15,15;background-color:beige;width:300;border-color:bla
ck;border-width:1;border-style:solid">
            Operand 1: <br><asp:TextBox id="Operand1" Text="15"
runat="server"/><br>
            Operand 2: <br><asp:TextBox id="Operand2" Text="5" runat="server"/><p>
                <input type="submit" id="Add" value="Add" OnServerClick="Submit_Click"
runat="server">
                <input type="submit" id="Subtract" value="Subtract"
OnServerClick="Submit_Click" runat="server">
                <input type="submit" id="Multiply" value="Multiply"
OnServerClick="Submit_Click" runat="server">
                <input type="submit" id="Divide" value="Divide"
OnServerClick="Submit_Click" runat="server">
            <p>
                <asp:Label id="Result" runat="server"/>
            </div>
        </form>
    </body>
</html>

```

还需要一个 sdl 文件，当然这个文件不用手工输入，在浏览一个 .asmx 的时候，后缀名后直接加上 ?sdl 可以自动生成 sdl 文件。然后在一个 .vb 文件里，将定义一个名字空间，.vb 文件的内容：

```

Imports System.Xml.Serialization
Imports System.Web.Services.Protocols
Imports System.Web.Services

Namespace MathServiceSpace
    Public Class MathService
        Inherits System.Web.Services.Protocols.SoapClientProtocol

        Public Sub New()

```

```
MyBase.New
Me.Url =
"http://localhost/QuickStart/aspplus/samples/Services/MathService/VB/MathSer
vice.a"& _
    "smx"
End Sub
Public Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Add")
> Add(ByVal <System.Xml.Serialization.XmlElementAttribute("A",
IsNullable:=false)> a As Integer, ByVal
<System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As
Integer) As Integer
    Dim results() As Object = Me.Invoke("Add", New Object() {a, b})
    Return CType(results(0),Integer)
End Function
Public Function BeginAdd(ByVal a As Integer, ByVal b As Integer, ByVal
callback As System.AsyncCallback, ByVal asyncState As Object) As
System.IAsyncResult
    Return Me.BeginInvoke("Add", New Object() {a, b}, callback,
asyncState)
End Function
Public Function EndAdd(ByVal asyncResult As System.IAsyncResult) As
Integer
    Dim results() As Object = Me.EndInvoke(asyncResult)
    Return CType(results(0),Integer)
End Function
Public Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Subtr
act")> Subtract(ByVal <System.Xml.Serialization.XmlElementAttribute("A",
IsNullable:=false)> a As Integer, ByVal
<System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As
Integer) As Integer
    Dim results() As Object = Me.Invoke("Subtract", New Object() {a,
b})
    Return CType(results(0),Integer)
End Function
Public Function BeginSubtract(ByVal a As Integer, ByVal b As Integer,
ByVal callback As System.AsyncCallback, ByVal asyncState As Object) As
System.IAsyncResult
    Return Me.BeginInvoke("Subtract", New Object() {a, b}, callback,
asyncState)
End Function
```

```
Public Function EndSubtract(ByVal asyncResult As System.IAsyncResult)
As Integer
    Dim results() As Object = Me.EndInvoke(asyncResult)
    Return CType(results(0),Integer)
End Function
Public Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Multi
ply")> Multiply(ByVal <System.Xml.Serialization.XmlElementAttribute("A",
IsNullable:=false)> a As Integer, ByVal
<System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As
Integer) As Integer
    Dim results() As Object = Me.Invoke("Multiply", New Object() {a,
b})
    Return CType(results(0),Integer)
End Function
Public Function BeginMultiply(ByVal a As Integer, ByVal b As Integer,
ByVal callback As System.AsyncCallback, ByVal asyncState As Object) As
System.IAsyncResult
    Return Me.BeginInvoke("Multiply", New Object() {a, b}, callback,
asyncState)
End Function
Public Function EndMultiply(ByVal asyncResult As System.IAsyncResult)
As Integer
    Dim results() As Object = Me.EndInvoke(asyncResult)
    Return CType(results(0),Integer)
End Function
Public Function
<System.Web.Services.Protocols.SoapMethodAttribute("http://tempuri.org/Divid
e")> Divide(ByVal <System.Xml.Serialization.XmlElementAttribute("A",
IsNullable:=false)> a As Integer, ByVal
<System.Xml.Serialization.XmlElementAttribute("B", IsNullable:=false)> b As
Integer) As Integer
    Dim results() As Object = Me.Invoke("Divide", New Object() {a,
b})
    Return CType(results(0),Integer)
End Function
Public Function BeginDivide(ByVal a As Integer, ByVal b As Integer,
ByVal callback As System.AsyncCallback, ByVal asyncState As Object) As
System.IAsyncResult
    Return Me.BeginInvoke("Divide", New Object() {a, b}, callback,
asyncState)
End Function
```

```

        Public Function EndDivide(ByVal asyncResult As System.IAsyncResult)
As Integer
            Dim results() As Object = Me.EndInvoke(asyncResult)
            Return CType(results(0),Integer)
        End Function

    End Class
End Namespace

```

有了这四个文件，可以编辑一个批处理文件，执行如下的语句：

```
WebServiceutil -c:proxy /pa:MathService.sdl /l:VB /n:MathServiceSpace
```

这样就可以在客户端执行了。

10.3 数据交换

这个例子说明了 DataSet-----一个基于 XML 技术的强大的数据分离技术，能够用 Web Service 方法返回。DataSet 能够在智能化的结构中存储复杂的信息和关系，这是 Web Service 的一个非常有用的方法。

通过 DataSets 的显示，能够限制通过连接数据库服务器的测试。

GetTitleAuthors 方法连接一个数据库并且运行两个 SQL 语句，第一个返回颜色的列表，另外一个返回字体大小的列表。方法把两个结果用一个 DataSet 来存储，并返回一个 DataSet。

PutTitleAuthors 说明一个 Web Service 方法把 DataSet 当作一个参数并返回一个整数，这个整数就是在 DataSet 中的“Table”表的行数。虽然这个方法执行起来有点简单，但是这个方法也能够与数据库服务器把过剩的数据聪明的合并在一起。

来看看这个例子，首先：

```
<%@ WebService Language="C#" Class="DataService" %>
```

这句话应该包括。我们还要引入这个名字空间：

```
using System.Web.Services;
```

第一我们两个方法，Getcolor()：

```

    [WebMethod] public DataSet Getcolor(){
        SqlConnection myConnection = new
SqlConnection("server=localhost;uid=sa;pwd=;database=howff");
        SQLDataSetCommand myCommand1 = new
SQLDataSetCommand("select * from color", MyConnection);
        SQLDataSetCommand myCommand2 = new
SQLDataSetCommand("select * from size", MyConnection);
        //数据的填充
        DataSet ds = new DataSet();
        MyCommand1.FillDataSet(ds, "color");
    }

```

```

        MyCommand2.FillDataSet(ds, "size");
        Return ds;
    }

```

Putcolor()方法

```

[WebMethod] public Integer Putcolor(DataSet ds){
    //返回行数
    return ds.Tables[0].Rows.Count;
}

```

文件保存为 WebService.asmx，放在虚拟目录下，具体代码如下：

```

<%@ WebService Language="C#" Class="DataService" %>
using System;
using System.Data;
using System.Data.SqlClient;
//引入System.Web.Services名字空间
using System.Web.Services;

public class DataService{
    [WebMethod] public DataSet Getcolor(){
        SqlConnection myConnection = new
        SqlConnection("server=localhost;uid=sa;pwd=;database=howff");
        SqlDataAdapter myCommand1 = new
        SqlDataAdapter("select * from color", MyConnection);
        SqlDataAdapter myCommand2 = new
        SqlDataAdapter("select * from size", MyConnection);
        //数据的填充
        DataSet ds = new DataSet();
        MyCommand1.FillDataSet(ds, "color");
        MyCommand2.FillDataSet(ds, "size");
        Return ds;
    }
    [WebMethod] public Integer Putcolor(DataSet ds){
        //返回行数
        return ds.Tables[0].Rows.Count;
    }
}

```

运行结果如图 10-4。

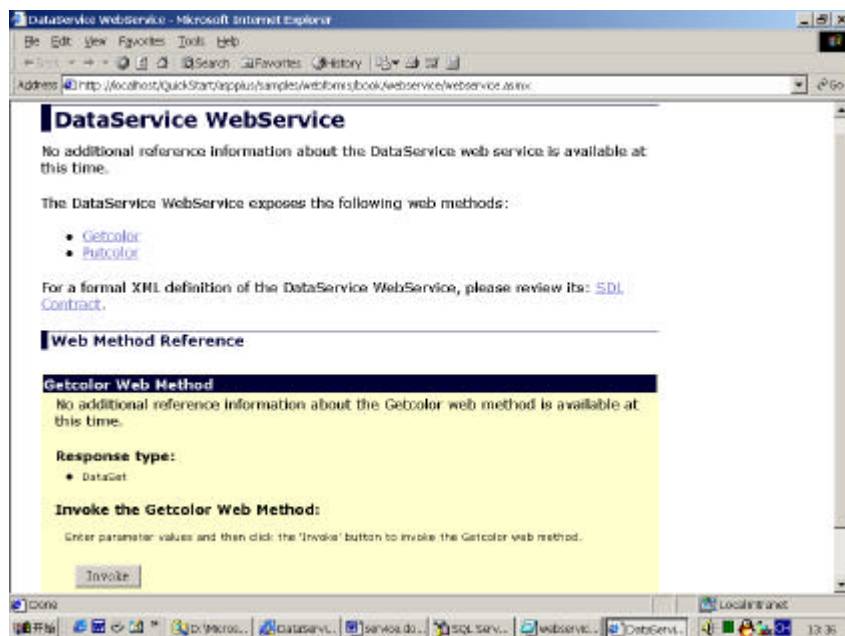


图 10-4

点击“Invoke”按钮，可以浏览到一个xml形式的文件，如图10-5所示。

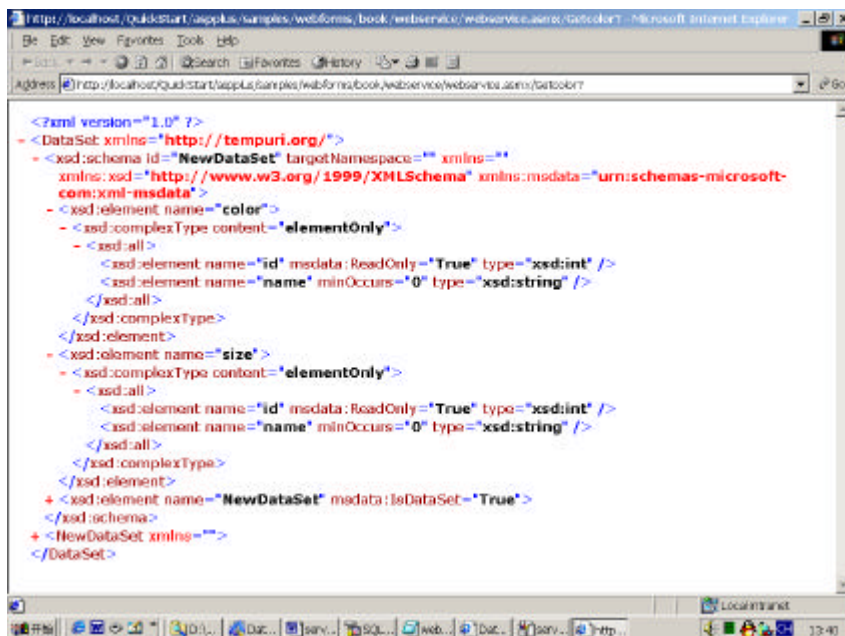


图 10-5

如果点击了“putcolor”连接后再点击“Invoke”按钮，则得到不同的结果，有兴趣的读者不妨试一试。

10.4 存取站点对象

下面怎么样通过 Web Service 来访问 Web 站点的固有的东西，如 Session 和 Application 属性，以及如何关闭 Session。

asmx 文件例子中的第一个方法，UHC 访问 Session，并在“HitCount”变量上加 1。定义 Session 计数器方法：

```
Public Function <WebMethod(EnableSession:=true)> UHC() As String
    If Session("HitCounter") Is Nothing
        Session("HitCounter") = 1
    Else
        Session("HitCounter") = Cint(Session("HitCounter")) + 1
    End If
    Return "你已经访问这个服务器 " & Session("HitCounter").ToString() & "
次了."
End Function
```

注意在方法名称的前面，我们加上了“<WebMethod(EnableSession:=true)>”这个修饰语句。在更新计数器的方法中：

```
Public Function <WebMethod(EnableSession:=false)> UAC() As String
    If Application("HitCounter") = Nothing
        Application("HitCounter") = 1
    Else
        Application("HitCounter") = Cint(Application("HitCounter")) + 1
    End If
    Return "服务被访问了 " & Application("HitCounter").ToString() & "
次."
End Function
```

注意与上面方法的区别，我们在修饰方法名称的语句

“<WebMethod(EnableSession:=false)>”不一样。下面是我们完整的例子代码：

```
<%@ WebService Language="VB" Class="SessionService1" %>
Imports System
Imports System.Web.Services

Public Class SessionService1 : Inherits WebService
    '增加计数器的值方法
    Public Function <WebMethod(EnableSession:=true)> UHC() As String
        If Session("HitCounter") Is Nothing
            Session("HitCounter") = 1
```

```
Else
    Session("HitCounter") = Cint(Session("HitCounter")) + 1
End If
Return "你已经访问这个服务器 " & Session("HitCounter").ToString() & "
次了."
End Function
```

'更新计数器值的方法

```
Public Function <WebMethod(EnableSession:=false)> UAC() As String
    If Application("HitCounter") = Nothing
        Application("HitCounter") = 1
    Else
        Application("HitCounter") = Cint(Application("HitCounter")) + 1
    End If
    Return "服务被访问了 " & Application("HitCounter").ToString() & " 次."
End Function
```

End Class

下面是我们的运行结果如图 10-6。

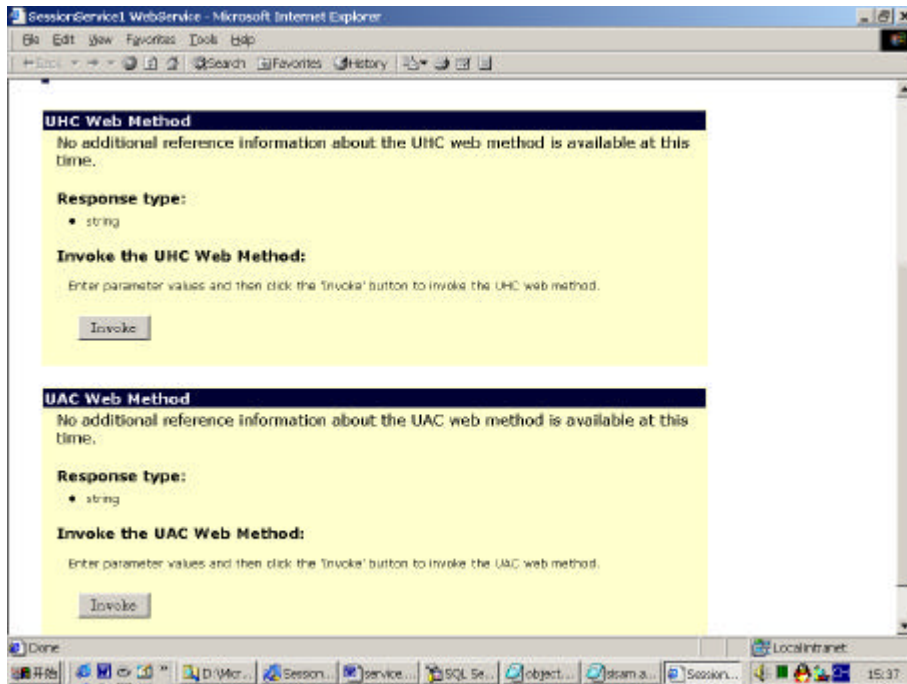


图 10-6

点击上面的“Invoke”按钮，有如下结果如图 10-7。



图 10-7

点击下面的“Invoke”按钮，有如下结果（在此之前我们已经点击了很多回），如图 10-8 所示。

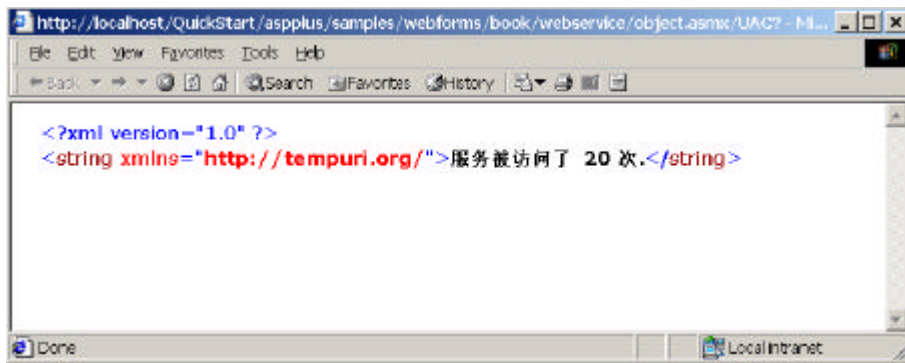


图 10-8

第 11 章 性能优化

在计算机科学领域，广泛应用缓冲技术来提高系统的性能，它的原理是把经常存取的或者是比较重要的数据保存于内存中以减少系统的响应时间。对于 Web 应用领域，缓冲技术主要是把 HTTP 请求的页面或数据保存于内存，以减少下次使用时重建它们的费用。

ASP.NET 有两种用于 Web 应用的缓冲技术：输出缓冲和数据缓冲。输出缓冲指把一次请求所产生的动态输出保存于内存中。数据缓冲指按照一定的策略把事先不确定的对象保存于内存中。

输出缓冲常用于把整个输出页面缓冲起来。对于一个存取繁忙的站点来说，把一些常用页面放入内存会带来性能上的极大提高。当一个页面被放入输出缓存，那么接下来的对该页面的请求将不再执行创建它的代码，而是从内存中直接返回该页面。

但实际上，保存整个输出页面的方法并不一定都行得通，因为有些页面的输出取决于客户端的不同请求，称之为“定制”。这时，采取的方法即找出不同中的相同，把一些并不需要经常重新创建的对象和数据识别出来，进行缓冲。一旦这些部分被识别，那么它们将被一次创建并在缓存中保持一定的时间。

选择缓存的时间是提高性能的关键。对一些部分来说，它们需要隔一定时间进行刷新，而另一些部分来说，可能仅仅只是需要保存一段时间。此种情况下，都可以设定“过期策略”来实现。一旦这些对象和数据到期，它们都将被从缓存中清除出去。当存取对象和数据的代码发现所要求的部分在内存中不存在时，将重建该对象或数据。

ASP.NET 支持文件和缓存关键字的依赖关系，它允许开发人员创建缓存依赖于一个外部文件或另一个缓存事物。利用这项技术可以更新一个缓存事物当其依赖的源文件发生改变时。

11.1 页面输出缓存

页面输出缓存通过保存动态页面的输出内容，大大提高了服务器应用的能力。缺省情况下，输出缓存选项是被打开的，但并不是任意给定的输出响应都将被缓存，除非显式地指定页面应被缓存。

为使输出能够被缓存，输出响应至少应有一个有效的过期/有效策略以及公用 cache 的访问权限。当一个 GET 请求被送往页面，一个输出缓冲入口将被创建。接下来，对该页面的 GET 请求和 HEAD 的请求将直接从该缓冲入口中取出返回给用户，而对该页面的 POST 请求通常是显式地产生动态内容，却并非如同 GET 和 HEAD 请求一样从缓冲入口中取出。

输出缓存还支持带请求串的 GET 方法，把请求串作为页面识别的一部分。这就意味着带有相同键值但排列次序不同的请求串的 GET 请求，可能导致缓存中认为不存在该输出页

面。

输出缓存需要知道页面缓存的过期/有效时间策略。如果一个页面在输出缓存中，而且又被指定为 60 分钟的页面过期时间，那么从它进入输出缓存开始，60 分钟后该页面将从输出缓存中被清除。如果恰在此时，有一个对该页面的请求到达，页面的代码将被执行，页面输出又将重新进入输出缓冲。这种方式的过期策略称之为“强制过期”，页面只在一定时间内有效。

```
<%@ OutputCache Duration=秒数 %>
```

用来显示指出页面在输出缓冲中的保存时间。

下面举一个简单的例子来证实 ASP.NET 中的页面缓存功能：

在一个页加载时显示它的时间，在页面过期时间（设为：10 秒）到达之前，把页面刷新（相当于重发 GET 请求），看一看显示的时间；然后，在过期时间到达之后，再看显示的时间。如果第一次和第二次显示的时间相同，那么就证明了系统存在有页面输出缓存功能，作为对比，当过期时间到达后，新的请求将导致重新执行页面代码，产生新的时间显示。

1. 程序源程序

```
<!--文件名:OutputCache/FormPageCache.aspx-->
<%@ OutputCache Duration="10" %>
<!--过期时间设为10秒-->
<html>

<head>
<title>
页面输出缓存测试
</title>
</head>
<script language="C#" runat="server">
    public void Page_Load(Object s, EventArgs E){
        lblTime.Text ="现在是:" + DateTime.Now.ToString();
    }
</script>

<body bgcolor=#ccccff>
    <center>
        <h2><font face="Verdana">测试页面输出缓存实验</font></h2>
        <p><p><p>
        <hr>
        </center>
        <asp:label id="lblTime" runat="server"/>
    </body>
```

</html>

2. 第一次输出结果 11-1。



图 11-1

3. 第二次输出结果如图 11-2。

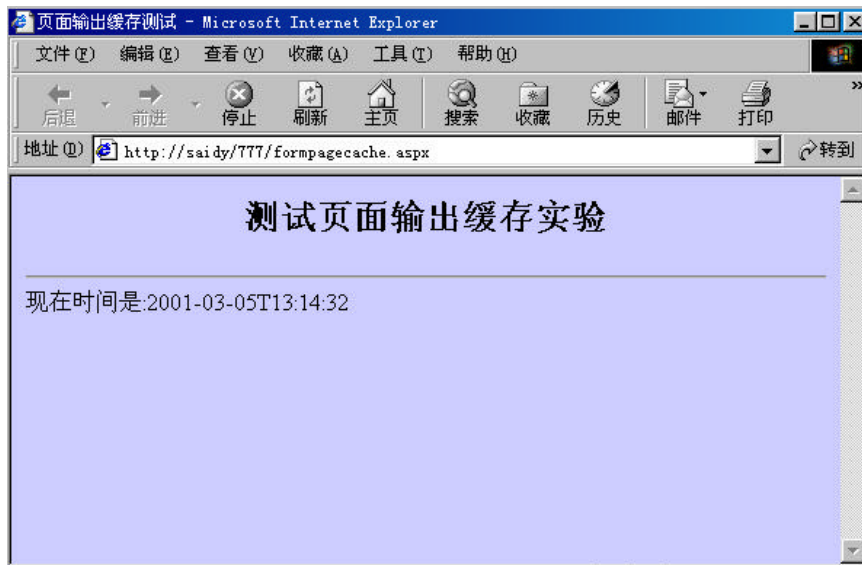


图 11-2

4. 第三次输出结果如图 11-3。



图 11-3

实现页面缓存的另一种方法

```
<%@ Import Namespace="System.Web.HttpCachePolicy" %>
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10));
//指定页面输出缓存下一个10秒到期
Response.Cache.SetCacheability(HttpCacheability.Public);
//指定所有用户都有对缓存的访问权力
/*
```

如果不希望进行页面缓存，可采用 `Response.Cache.SetSlidingExpiration` 方法，当其为 `True` 时，每次页面请求到达时，相当于页面过期时间到了，就要对页面输出重新刷新。

```
*/
Response.Cache.SetSlidingExpiration(True);
//当每次页面请求时,重置到期时间计数器,并且页面到期
```

例如上面的例子可以改写成如下例子：

```
<!--文件名:OutputCache/FormPageCache01.aspx-->
<%@ Import Namespace="System.Web.HttpCachePolicy" %>

<html>
<head>
<title>
页面输出缓存测试 1
</title>
</head>
```

```
<script language="C#" runat="server">

    public void Page_Load(s As Object, E As EventArgs){
        response.cache.setexpires(DateTime().Now.addseconds(10));
        response.cache.setcacheability(Httpcacheability.Public);
        lblTime.Text ="现在是:" +DateTime.Now.ToString();
    }
</script>

<body bgcolor=#ccccff>

    <center>
    <h2><font face="Verdana">测试页面输出缓存实验1</font></h2>
    <p><p><p>
    <hr>
    </center>
    <asp:label id="lblTime" runat="server"/>

</body>

</html>
```

输出的结果和上一例子大体相同。

另外，在 ASP 中，对于上面的例子还可以写为：

```
Response.CacheControl = "Public";
Response.Expires = 10;
```

从兼容性出发，ASP.NET 中依然具有相同的结果，但建议尽量使用 ASPNET 的形式。

11.2 页面数据缓存

ASP.NET 提供了一个相当出色的缓存引擎机制，它允许页面保存和索引 HTTP 请求所要求的各种各样的对象。ASP.NET 的缓存对各个应用来说是私有的，是存储各种对象的存储器。缓存的生存周期取决于应用的生存周期，也就是说，当应用重新启动时，缓存实际上也已重建。

缓存提供了一个简单的字典接口，以便于开发者能够轻而易举放置对象到缓存中，并在以后使用。最简单的情况下，放置一个对象到缓存中，就如同对字典增加一个条目。

例如：

```
Cache("myKey")=MyValue;
```

即是把 MyValue 放入缓存中一个叫 myKey 的缓存对象中，当需要引用 myKey 值时，可以采用下面的使用方式：


```

myValue1=Cache("myKey");
if (myValue1 != Null){
//非空时的动作
...
}

```

ASP.NET 提供了三种缓存替换的策略：

1. “腐烂搜索”(Scavenging)

比较类似于“最近最少使用”替换原则，当内存变得比较紧张时，缓存机制会找出最不常用和最不重要的对象，把它从内存中移出，以减轻系统压力。为控制“腐烂搜索”的具体行为，编程者必须在插入缓存对象时，指明插入它的耗费和多少时间内它必须被存取一次才能继续留在缓存中，以供替换时进行抉择。

2. “到期控制”(Expiration)

编程者可以指定缓存对象的生存周期，这种指定的时间可以是绝对的也可以是相对的。例如绝对的时间（下午 6:00 到期），相对时间（该对象距最近一次存取它的时间满 10 分钟即到期）。当一个缓存对象到期后，它将从缓冲内存中移出，此时对该对象的索引将得到 null 值，除非又重新插入该对象。

3. “文件和键值依赖”

从外部文件或者是其他缓存键值是否改变，来决定本身键值是否有效。如果依赖发生改变，缓存对象将变得不可使用，并从缓存中移动出来。试想这样一种情况，应用从一个 XML 文件中读出金融信息，而该文件又被定期地修改。应用的作用是利用从该文件读出的信息构造一个图形对象以表示销售的情况。当应用读入文件时，它把数据插入缓存中，并记录下文件的依赖关系。当文件发生修改时，应用使开始产生的缓存对象无效并从内存中移出已经无用的数据，此后应用重新读入文件的数据，再把更新后的数据放入缓存，这样就完成了信息的更新，并返回给最终用户。

例子：从一个 XML 中读出用户信息并显示在页面上，页面提供了两个按钮，一个为增加用户，一个为刷新。我们在内存中建立了一个缓存对象“DataCache5”，它与客户信息文件“custom1.xml”建立了依赖关系。当 custom1.xml 文件未发生变化时，我们按下“刷新”按钮，可以看到信息是从缓存中读出的；当我们输入客户相应的资料，增加了一个客户以后，再按下“刷新”按钮后，我们可以看到客户信息变成了从文件读出。

ASPX 源程序

```

<!-- 文件名：FormDataCache.aspx -->
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">
sub LoadData1

```

```
'当缓存对象DataCache5有效时，从缓存中读出客户信息；无效时，从文件读出信息
dim dv1 as DataView
    dv1=Cache("DataCache5")
    if dv1 = Nothing
        dim ds as DataSet
        dim fs as FileStream
        dim sr as StreamReader

        ds=New DataSet
        fs=New
FileStream(Server.MapPath("custom1.xml"), FileMode.Open, FileAccess.Read)
        sr=New StreamReader(fs)
        ds.ReadXml(sr)
        fs.Close()

        dv1=new DataView(ds.Tables(0))
        Cache.Insert("DataCache5", dv1, New
cachedependency(Server.MapPath("custom1.xml")))
        lblMsg.text="数据从文件中读出..."

    Else
        lblmsg.text="数据从缓存中读出..."
    end if
'绑定到DataGrid1对象
    DataGrid1.datasource = dv1
    DataGrid1.databind()

end sub

sub Page_Load(s as object,e as eventargs)
'加载页面时，从文件中读出客户信息
    if Not IsPostBack
        LoadData1()
    end if
end sub

sub AddBtn_Click(s as object,e as eventargs)
'增加一个客户信息到文件中
dim FS as FileStream
dim Reader as StreamReader
```

```
dim DS as DataSet
dim dr1 as DataRow
dim tw1 as TextWriter

    if Not Page.IsValid
        lblMsg.text="还有域未曾填充..."
    else
        DS=New DataSet()
        FS=New
FileStream(Server.MapPath("custom1.xml"), FileMode.Open, FileAccess.Read, FileS
hare.ReadWrite)
        Reader=New StreamReader(FS)
        DS.readxml(Reader)
        FS.Close()

        dr1=DS.tables(0).newrow()
        dr1("CustName")=txtName.text
        dr1("CustIdno")=txtIdno.text
        dr1("CustCard")=txtCard.text
        DS.tables(0).rows.add(dr1)

        FS=New
FileStream(Server.MapPath("custom1.xml"), FileMode.Create, FileAccess.ReadWrit
e, FileShare.ReadWrite)
        tw1=New StreamWriter(FS)
        tw1=textwriter.synchronized(tw1)
        DS.writexml(tw1)
        tw1.close()
        LoadData1()
    end if

end sub

sub RefreshBtn_Click(s as object,e as eventargs)
    LoadData1()
end sub

</script>
<head>
<title>
```

```
数据缓冲实验
</title>
</head>

<body>
<center>
<form runat=server>
<h2>XML文件缓存测试</h2>
<ASP:DataGrid id="DataGrid1" runat="server"
    Width="600"
    BackColor="#ccccff"
    BorderColor="black"
    ShowFooter="false"
    CellPadding=3
    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="8pt"
    HeaderStyle-BackColor="#aaaadd"
    />
<hr>
<h3>添加一个客户信息</h3>
<table>
<tr>
<td>姓 名:</td>
<td><ASP:textbox id=txtName runat=server /></td>
<td><ASP:RequiredFieldValidator ControlToValidate="txtName"
Display="static" ErrorMessage="*" runat=server /> </td>
</tr>

<tr>
<td>身份证号:</td>
<td><ASP:textbox id=txtIdno runat=server /></td>
<td><ASP:RequiredFieldValidator ControlToValidate="txtIdno"
Display="static" ErrorMessage="*" runat=server /> </td>
</tr>

<tr>
<td>信用卡号:</td>
<td><ASP:textbox id=txtCard runat=server /></td>
<td><ASP:RequiredFieldValidator ControlToValidate="txtCard"
```

```

Display="static" ErrorMessage="*" runat=server /> </td>
</tr>
</table>
<p>

<asp:button text="增加" onclick="AddBtn_Click" runat=server />
<asp:button text="刷新" onclick="RefreshBtn_Click" runat=server />

<p>
<p>
<p>
<asp:label id=lblMsg runat=server />
</form>
</center>
</body>
</html>

```

初始运行结果如图 11-4。



图 11-4

初始时，应用空间中无“DataCache”缓存对象，故文件从 custom1.xml 中读出客户信息，开始时为空，只显示了字段名。

当添加一个用户信息后，文件 custom1.xml 发生改变，导致“DataCache5”对象无效，LoadData1 过程依然从文件 custom1.xml 中读取信息，并更新 DataCache5 对象，如图 11-5。



图 11-5

由于 custom1.xml 文件未曾发生改变，所以当按下“刷新”按钮后，信息却是从缓存对象 DataCach5 中读出来，如图 11-6。



图 11-6

第 12 章 高级应用

12.1 XML 及其应用

XML 是标准扩展语言，是未来 Web 编程的标准。下面将介绍 XML 在 ASP.NET 中的应用。

12.1.1 制作广告条

在这个程序中，通过 xml 语言实现每次访问网页时，将显示不同的广告条。在本例中只调用了两条广告。

<!--Intro.aspx的代码如下：</p></div>

```
<html>
<center><title>广告条演示</title></center>
<head>
  <link rel="stylesheet"href="intro.css">
</head>
<body>
  <center>
    <form action="intro6.aspx" method="post" runat="server">
<h2>广告条演示</h2>
    <asp:adrotator AdvertisementFile="intro.xml" BorderColor="black"
BorderWidth=1 runat="server"/>
  </form>
  </center>
</body>
</html>
```

intro.xml 的代码如下：

```
<Advertisements>
  <Ad>
```

```
<ImageUrl>/quickstart/aspplus/samples/Webforms/book/liu/hp-idc.gif</ImageUrl>
>
```

```
  <NavigateUrl>http://www.yesky.com</NavigateUrl>
  <AlternateText>欢迎访问！</AlternateText>
  <Keyword>Computers</Keyword>
  <Impressions>80</Impressions>
</Ad>
<Ad>
```

```

<ImageUrl>/quickstart/asplus/samples/Webforms/book/liu/intel-streaming.gif<
/ImageUrl>
    <NavigateUrl>http://www.yesky.com</NavigateUrl>
    <AlternateText>欢迎访问</AlternateText>
    <Keyword>Computers</Keyword>
    <Impressions>80</Impressions>
</Ad>
</Advertisements>

```

在 intro.aspx 中使用了 `<asp:adrotator AdvertisementFile="intro.xml" BorderColor="black" BorderWidth=1 runat="server"/>` 这条语句来嵌入 intro.xml 文件。运行结果如图 12-1。



图 12-1

当点击刷新按钮或者按 F5 键，将看到另一条广告条。在本例中用到了 AdRotator 服务器控件，在 xml 文件中，可以定义其属性，如表 12-1 所示。

表 12-1

| 属性 | 描述 |
|---------------|-----------------------------|
| ImageUrl | 图像文件的绝对或相对地址 |
| NavigateUrl | 当图像被点击时，可访问相应的网页 |
| AlternateText | 当鼠标移动到图片上时，将显示提示信息 |
| Keyword | 指定广告条的分类，我们可以利用此属性来对广告条进行分类 |
| Impressions | 指定图片在表格中的大小 |

1. XML 和 dataset 控件结合使用

数据访问是一个应用系统的核心。公用语言运行环境 (Common Language Runtime)

提供了管理数据访问应用程序接口 (API) 的方法。而这些 API 将不论它的数据源是什么, 如 SQL Server, OLEDB, XML 等, 都能提取出所需要的数据。在具体编程的时候, 有 3 个对象将常常用到: Connections, Commands, and DataSets, 如表 12-2 所示。

表 12-2

| 对象 | 描述 |
|------------|--|
| Connection | 与数据源进行连接。如 SQL Server 或者一个 XML 文件。 |
| Command | 对数据进行操纵。如进行删除 (delete), 提取 (select), 更新 (update) |
| Dataset | 显示出所需的数据 |

为了能使用 SQL 语句, 要先导入 System.Data 和 System.Data.SQL 这两个名字空间。

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SQL" %>
```

几个典型的 SQL 语句的说明, 如表 12-3。

表 12-3

| 查询 | 示例 |
|------------|---|
| 简单的 select | SELECT * from Student WHERE stuname = '小李'; |
| 联合的 select | SELECT * from Student S, Dept D WHERE S.dept= D.dept; |
| Insert | INSERT into Student VALUES ('小王', 21, '男'); |
| Delete | DELETE from Student WHERE name='小王'; |
| Update | UPDATE Student SET age = 21 WHERE name='小李'; |

在执行查询之前, 要先构建一个 SqlDataAdapter 对象, 在执行查询以后, 需要把数据转移到 DataSet 中, 就要使用下面的代码:

```
SqlConnection myConnection = new
SqlConnection("server=localhost;uid=sa;pwd=;database=test");

SqlDataAdapter myCommand = new SqlDataAdapter("select * from
student",
myConnection);
DataSet ds = new DataSet();
myCommand.FillDataSet(ds, "student");
```

来看如何从 xml 文件中读取数据。DataSet 控件提供了 ReadXml 方法。同时在 XML 文件中, 必须存在 schema 和所需要的数据 (Data)。

先看一下 data1.aspx 文件:

```
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Data" %>
<html>
```

```
<script language="C#" runat="server">

    public void Page_Load(Src As Object, E As EventArgs){
        DataSet DS = new DataSet();
        FileStream FS;
        StreamReader Reader;
        FS = new
FileStream(Server.MapPath("data1.xml"), FileMode.Open, FileAccess.Read) ;
        Reader = new StreamReader(FS) ;
        DS.ReadXml(Reader) ;
        FS.Close();
        DataView Source;
        Source = new DataView(ds.Tables(0)) ;
        MySpan.InnerHtml = Source.Table.TableName();
        MyDataGrid.DataSource = Source;
        MyDataGrid.DataBind();
    }
</script>
<body>
    <h3><font face="Verdana">XML 演示 <span runat="server"
id="MySpan"/></font></h3>
    <ASP:DataGrid id="MyDataGrid" runat="server" />
</body>
</html>
```

再来看看 xml 文件的结构。

data1.xml 的文件如下：

```
<root>
<schema id="DocumentElement" targetNamespace=""
xmlns=http://www.w3.org/1999/XMLSchema
xmlns:xdo="urn:schemas-microsoft-com:xml-xdo"
xdo:DataSetName="DocumentElement">
    <element name="student">
        <complexType content="elementOnly">
            <all>
                <element name="name" type="string"></element>
                <element name="age" type="int"></element>
                <element name="sex" type="string"></element>
                <element name="grade" type="string"></element>
            </all>
        </complexType>
    </element>
```

```
</schema>
<DocumentElement>
  <student>
    <name>jimmy</name>
    <age>20</age>
    <sex>boy</sex>
    <grade>freshman</grade>
  </student>
  <student>
    <name>Mary</name>
    <age>20</age>
    <sex>girl</sex>
    <grade>sophomore</grade>
  </student>
  <student>
    <name>Tom</name>
    <age>19</age>
    <sex>boy</sex>
    <grade>freshman</grade>
  </student>
  <student>
    <name>Susan</name>
    <age>20</age>
    <sex>girl</sex>
    <grade>freshman</grade>
  </student>
</DocumentElement>
</root>
```

程序结果如图 12-2。

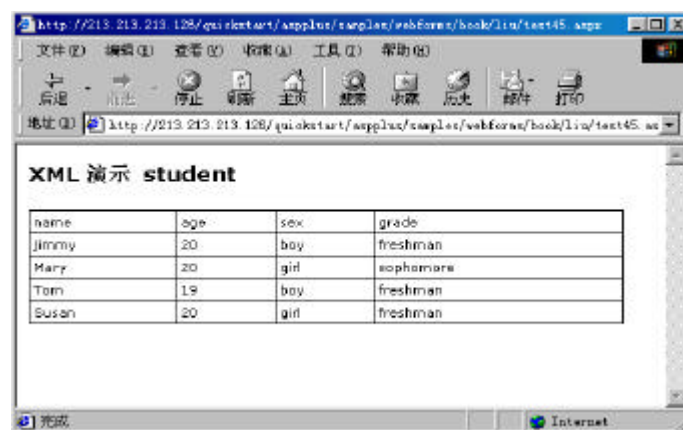


图 12-2

当然也可以把 schema 和数据分开成对立的文件。在主文件中，要分别使用 the ReadXmlData 和 ReadXmlSchema 方法。

如：

读取 schema.xml 的内容：

```
FS = new
FileStream(Server.MapPath("schema.xml"), FileMode.Open, FileAccess.Read);
Schema = new StreamReader(FS);
DS.ReadXmlSchema(Schema);
FS.Close();
```

读取 data.xml 的内容：

```
FS = New
FileStream(Server.MapPath("data.xml"), FileMode.Open, FileAccess.Read);
Reader = New StreamReader(FS);
DS.ReadXmlData(Reader);
FS.Close();
```

上面提到的 data1.xml 分成两部分：

第一部分：生成 schema.Xml 文件：

```
<schema id="DocumentElement" targetNamespace=""
xmlns=http://www.w3.org/1999/XMLSchema
xmlns:xdo="urn:schemas-microsoft-com:xml-xdo"
xdo:DataSetName="DocumentElement">
  <element name="student">
    <complexType content="elementOnly">
      <all>
        <element name="name" type="string"></element>
        <element name="age" type="int"></element>
        <element name="sex" type="string"></element>
        <element name="grade" type="string"></element>
      </all>
    </complexType>
  </element>
</schema>
```

第二部分生成 data.xml：

```
<DocumentElement>
  <student>
    <name>jimmy</name>
    <age>20</age>
    <sex>boy</sex>
    <grade>freshman</grade>
```

```
</student>
<student>
  <name>Mary</name>
  <age>20</age>
  <sex>girl</sex>
  <grade>sophomore</grade>
</student>
<student>
  <name>Tom</name>
  <age>19</age>
  <sex>boy</sex>
  <grade>freshman</grade>
</student>
<student>
  <name>Susan</name>
  <age>20</age>
  <sex>girl</sex>
  <grade>freshman</grade>
</student>
</DocumentElement>
```

12.2 三层结构及其应用

12.2.1 概念和环境

ASP.NET 中的三层结构开发方法，其实其思想跟 Java 的一样。Java 中的三层架构为前端的 html、Jsp、Servlet，中间层为 JavaBean、EJB，后面为数据库服务器。而在 ASP.NET 中，前端为 html、asp、aspx 等，中间层为有 .vb、.cs 等文件编译而成的 .dll 控件，后面为数据库服务器。

在三层架构中，数据库层通过中间层来连接以及操作，前端给中间层传递参数，并接受中间层的参数。在 ASP.NET 中，主要关注的是中间层与前端的数据交互。

一般统称中间层为组件，组件可以用 .vb 编译而成，也可以用 .cs 文件编译而成。中间层一般为 .dll 文件。微软的 .NET 技术在这个方面比他的以前的任何版本都要来的简单，这也是它的一大好处之一。以前我们要注册一个 .dll 文件，且注册时必须重新启动，而在 .NET 上，我们的 .dll 文件拿来就用，不用再考虑注册的问题。

在没有 Visual Studio.NET 之前，用写成的 .bat 文件来把 .vb 和 .cs 文件编译成 .dll 文件，在 .bat 文件里，写入编译的文件名称、相关联的名字空间、要编译成的文件名以及对应的命令名称，然后运行就行了。听起来很复杂，这也是很多初学者在编译第一个 .dll 文件时所害怕的事情。但是做起来很简单的。下面我们举一个例子来说明 .bat 文件的写法，假设我

们有一个文件名为：saidy.vb 的文件，要把它编译成 saidy.dll 的文件，其中用到 System、System.Data、System.Data.SQL 名字空间，我们可以创建一个 saidy.bat 文件，内容如下：

```
vbc /out:..\bin\saidy.dll /t:library /r:system.dll /r:system.data.dll
/ r:system.data.sql.dll
saidy.vb
```

这是编译.vb 程序的命令，如果是编译.cs 文件，则命令会是不一样，我们假定有一个 saidy.cs 的文件，按照上面的要求，我们编译如下：

```
cs /out:..\bin\saidy.dll /t:library /r:system.dll /r:system.data.dll
/ r:system.data.sql.dll
saidy.cs
```

我们可以看出来，大部分是一样的。

当然，如果我们有微软公司的 vs.net 编程环境，则我们不用这么麻烦，我们可以象编译 vb 或者 vc 程序一样方便的编译.dll 文件。微软公司的 vs.net 是一个集大成者，把各种语言整合起来，在这个环境下都可以写出不同语言的程序。具体的应用我们会在专门的章节中介绍。

12.2.2 一个基于三层架构的例子

我们通过具体的例子来说明三层架构的应用，接下来将建一个小项目来说明这个问题。有时为了安全性，通常把与数据库的连接用一个动态连接库文件封装起来，这样就要把写数据库连接的.vb 或者.cs 文件编译成动态连接库.dll 文件。甚至我们把对数据库的相关操作页编译成.dll 文件。

下面是与数据库连接以及操作的文件 dblink.cs 的主要部分，对数据库的连接：

```
SqlConnection dbl = new
SqlConnection("server=localhost;uid=sa;password=;database=howff")
对数据库的操作，我们把它写在一个方法里面，再返回相应值：
public DataView getdata() {
    string sStr;
    sStr = "select * from color";
    SQLDataSetCommand sComm = new SQLDataSetCommand(sStr, dbl);
    DataSet sDS;
    //填充数据
    sDS = new DataSet();
    sComm.FillDataSet(sDS, "color");

    //返回
    return sDS.Tables["color"].DefaultView;
}
```

第六个语句就用到上面的与数据库的连接变量，这个函数的功能是从表“color”中选

出所有的元素，并返回表结构的形式。完整的代码如下：

```
(ThreeLayer/saidy.cs)
using System;
using System.Data;
using System.Data.SqlClient;
//创建名字空间
namespace db{
    //创建一个类
    public class dblink{
        //建立数据库的连接
        SqlConnection dbl = new
SqlConnection("server=localhost;uid=sa;password=;database=bbs");
        //方法
        public DataView getdata() {
            string sStr;
            sStr = "select * from color";
            SqlDataAdapter sComm = new SqlDataAdapter(sStr, dbl);
            DataSet sDS;
            //填充数据
            sDS = new DataSet();
            sComm.FillDataSet(sDS, "color");
            //返回
            return sDS.Tables["color"].DefaultView;
        }
    }
}
```

我们再写一个前端调用页面 saidy.aspx，首先要引入创建的名字空间：

```
<%@ Import Namespace="db" %>
```

在页面装入的时候，用此方法：

```
public void Page_Load(Object Sender, EventArgs E){

    //建立一个新的对象
    dblink newdb = new dblink();

    //数据来源
    Products.DataSource = newdb.getdata();

    //数据绑定
    Products.DataBind();
}
```

下面看看我们完整的代码：

```
(ThreeLayer/saidy.aspx)
<%@ Page language="c#" %>
<%@ Import Namespace="db" %>
<html>
<script runat="server">
    public void Page_Load(Object Sender, EventArgs E){
        //建立一个新的对象
        dblink newdb = new dblink();
        //数据来源
        Products.DataSource = newdb.getdata();
        //数据绑定
        Products.DataBind();
    }
</script>
<body style="font: 10pt verdana" bgcolor="CCCCCCF">
<BR><BR><BR>
<CENTER>
    <h3>.NET->三层架构!</h3>
</CENTER>
<BR><BR>
<CENTER>
    <ASP:DataList id="Products" ShowHeader=false ShowFooter=false
RepeatColumns="2" RepeatDirection="horizontal" BorderWidth=0 runat="server">

        <template name="itemtemplate">
            <table>
                <tr>
                    <td width="150" style="text-align:center; font-size:8pt;
vertical-align:top; height:50">
                        <p>
                            <%# DataBinder.Eval(Container.DataItem, "USERNICKNAME",
"{0:C}").ToString() %> <br>
                            <%# DataBinder.Eval(Container.DataItem, "SUBJECT",
"{0:C}").ToString() %>
                        </td>
                    </tr>
                </table>
            </template>

        </ASP:DataList>
</CENTER>
</body>
```



```
</html>
```

我们看到，在这个页面当中，没有出现与数据库交互的语句，这样就很好的把数据操作封装起来了，运行结果如图 12-3。

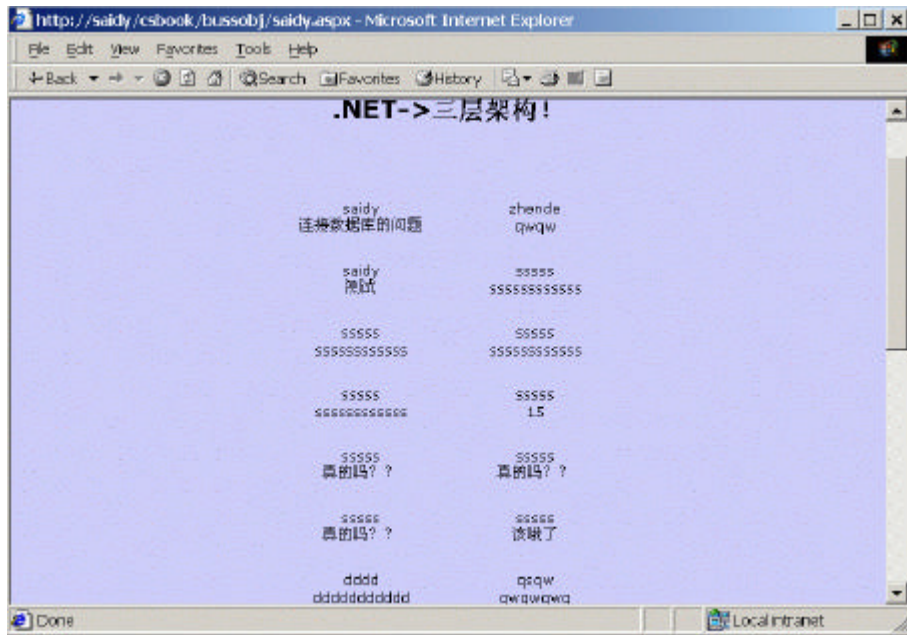


图 12-3

这只是一个非常简单的例子，.NET 在这方面的功能是非常强大的，你可以用它来写非常复杂的组件。

12.3 ASP 与 ASP.NET 的交互使用

前面讲过，asp 与 aspx 文件可以同时在一个目录下面运行而互相不影响的。比如我们在运行 aspx 文件的 Web 目录下面建立一个 asp 文件 在 Web 根目录下面建立一个 Global.asa 文件，这个 Global.asa 文件与我们应用于 ASP.NET 的 Global.asax 文件是互相不影响的，当然也不会与 Config.Web 文件互相影响。

也可以在 asp 文件中调用 C#或者 VB.NET 写的组件，反过来，在 aspx 文件中也可以调用以前写的组件。比如我们有一个注册了的组件 BmanagerServer.dll，在编程过程中，可以这样来把它加到我们的环境：

先展开项目名称，右键点击“References”，点击“Add References...”如图 12-4：

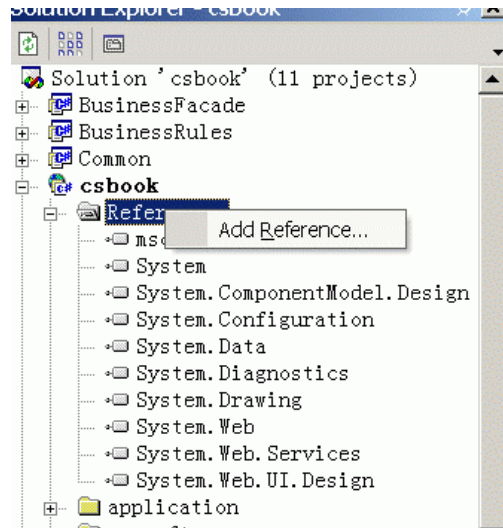


图 12-4

弹出下面的对话框，点击“COM”选项，找到我们的组件，点击“Select”，在下面的选区中出现组件，点击“Ok”，如图 12-5 所示。

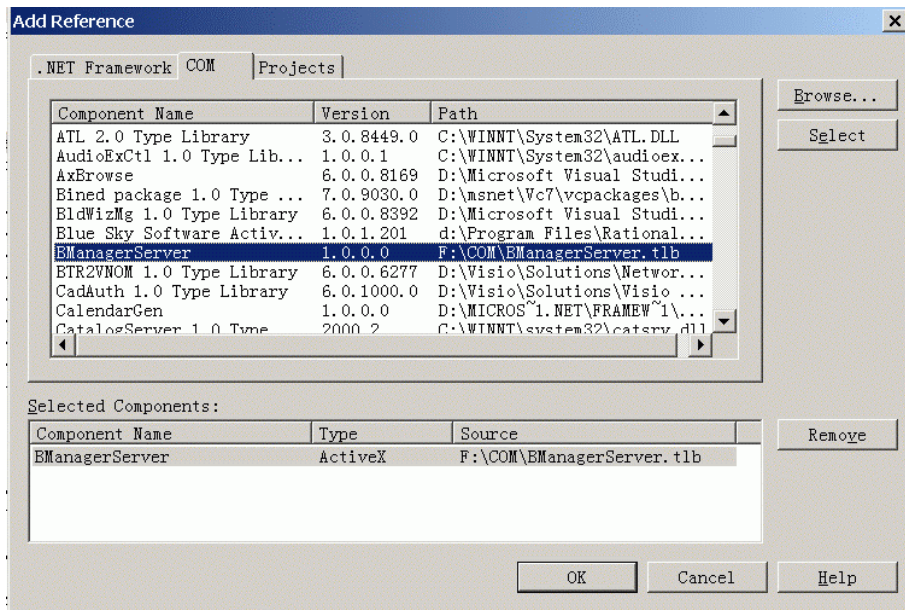


图 12-5

就加到了我们的“References”中，你就可以像引用其他的名字空间一样来引用它了。

第 13 章 ASP.NET 实战篇

13.1 多线程

多线程的内容是很丰富，如果你从来没有在 C#或者.NET 中写过多线程程序，那么我们这个例子对你是一个很大的帮助。两个线程打印信息到屏幕上，Thread 类中把泛函性压到.NET 里面。

你可以在通过 Thread 类的构造器来创建一个新的线程，可以用 Abort 方法来杀死一个线程，在杀死一个线程之前应该确定一下线程是否活着。

下面是我们的代码 (first\mt1.cs):

```
namespace First
{
    using System;
    using System.Threading;
    /// <summary>
    ///     Summary description for mt1.
    /// </summary>

    public class mt1
    {
        public static void PrintCurDateTime()
        {
            Console.WriteLine( " Secondary Thread" );
            Console.WriteLine( DateTime.Now.ToLongTimeString() );
            Console.WriteLine( "很不错的一个程序!" );
        }

        public static int Main(String[] args)
        {
            Console.WriteLine("Main Thread \n");
            try
            {
                Thread secThread = new Thread(new
ThreadStart( PrintCurDateTime ));
                secThread.Priority = ThreadPriority.Highest;
                secThread.Start();
                if ( secThread.IsAlive)
                {
```

```
        secThread.Abort();
    }
}
catch (Exception e)
{
    Console.WriteLine( e.ToString());
}

return 0;
}
}
}
```

在 vs.net 中编译成一个.exe 文件，再执行，结果如图 13-1。

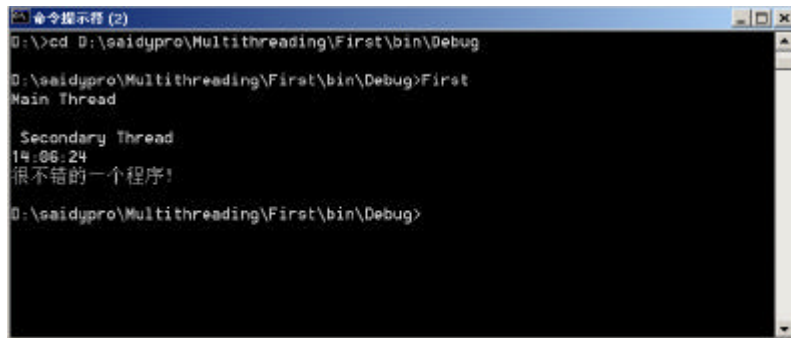


图 13-1

13.2 写、读和删除文件

13.2.1 写文件

在 ASP.NET 中，有一个类 File，File 类中有一个 CreateText()方法，我们直接应用它来创建一个文本文件。

文件 (WriteFile.aspx):

```
<%@ Import Namespace="System.IO" %>
<html>
<head>
<title>Writing to a text file using ASP+</title>
</head>
<BODY BGCOLOR="#CCCCCCF">
<%
    Dim txtWriter As StreamWriter
    txtWriter= File.CreateText( "F:\nettest\WriteFile.txt" )
```

```
txtWriter.WriteLine( "Hello" )
txtWriter.WriteLine( "大家好啊!" )
txtWriter.WriteLine( "Goodbye" )
txtWriter.Close
Response.Write( "创建文件成功!" )
%>
</body>
</html>
```

运行结果如图 13-2。

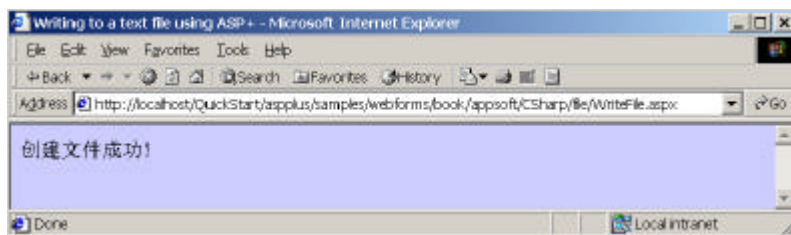


图 13-2

其中创建的文件内容如图 13-3 所示。



图 13-3

13.2.2 读文件

File 类的 OpenText()方法可以打开一个文件，我们用它来打开一个.xml 文件。

Xml 文件 (saily.xml)：

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Profile>
<Profile>
  <firstName>Saidy</firstName>
  <lastName>Chan</lastName>
  <phoneNumber>(010)-000-00000</phoneNumber>
  <address1>清华大学</address1>
  <address2>海淀区</address2>
  <city>北京</city>
  <state>北京</state>
  <postal>100000</postal>
```

```
</Profile>
```

读文件 (ReadFile.aspx) :

```
<%@ page language="c#"%>
<%@ Import Namespace="System.IO" %>
<html>
<head>
<title>读文件</title>
</head>
<body>

<%

    StreamReader sr = File.OpenText
("F:\\csbook\\appsoft\\file\\saily.xml");
    String strLine = null;
    do
    {
        Response.Write(strLine + "<br>");
    }
    while (null != (strLine = sr.ReadLine ()));
    sr.Close() ;

%>

</body>
</html>
```

运行结果如图 13-3 所示。

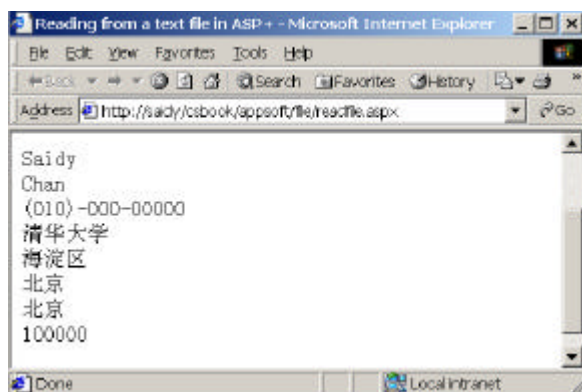


图 13-4

13.2.3 删除文件

文件的删除在 ASP.NET 中是很容易实现的,在名字空间 System.IO 中有一个 file 类,类中有一个 delete 方法,后面跟上我们要删除的文件,就可以实现删除文件的目的。

具体如 DeleteFile.aspx :

```
<%@ page language="c#" %>
<%@ Import Namespace="System.IO" %>
<html>
<head>
<title>删除文件</title>
</head>
<body>

<%

    File.Delete("F:\\csbook\\appsoft\\file\\sample.txt");
    Response.Write("删除文件成功!");

%>

</body>
</html>
```

删除操作完成后输出“删除文件成功”的提示。

13.3 发送邮件

在 ASP.NET 中发送邮件是一件轻松的事情,因为有专门的对象来处理邮件的发送。对象 MailMessage 就是这个对象,在下面的例子中,我们先用 MailMessage 来创建一个新的对象:

```
MailMessage MailObj = new MailMessage();
```

然后得到邮件的发送和接收地址:

```
MailObj.From = mailfrom.Value;
```

```
MailObj.To = to.Value;
```

当然,你也可以加上你的附件,下面是代码:

文件 (mail01.aspx):

```
<%@page language="C#" %>
<%@Import Namespace="System.Web" %>
<%@Import Namespace="System.Web.Util" %>

<HTML><BODY>
```

```
<SCRIPT LANGUAGE="C#" RUNAT="server" ID=SCRIPT1>
//当在客户端按下提交按钮时,这个方法在服务器端被调用。
public void SendMail (Object Obj, EventArgs E)
{
    //初始化Object
    MailMessage mailObj = new MailMessage();

    //设置发送和接受地址
    mailObj.From = mailfrom.Value;
    mailObj.To = to.Value;

    mailObj.Subject = "邮件标题";
    mailObj.Body = "邮件内容";

    // 设置为html格式。
    mailObj.BodyFormat = MailFormat.Html;

    // mailObj.BodyEncoding = MailFormat.Base64;

    //设置安全性
    mailObj.Priority = MailPriority.High;

    //增加附件
    mailObj.Attachments.Add(new MailAttachment("d:\\saily.xml"));

    // 用SmtpMail对象发送邮件
    SmtpMail.Send(mailObj);
}
</SCRIPT>

<asp:label ID="Headingmsg" Text="输入你的邮件地址:" RUNAT="server"/>

<FORM METHOD="post" RUNAT="server" ID=Form1>

发送给: <INPUT TYPE="text" NAME="to" id = "to" runat = server > <br>
发送者Email: <INPUT TYPE="text" NAME="mailfrom" id = "mailfrom" runat = server
>

    <INPUT TYPE="submit" NAME="Submit" VALUE="立即发送" RUNAT="server"
OnServerClick="SendMail" ID=Submit1>
```



```
</FORM>  
</BODY>
```

13.4 BBS 程序

我们用 ASP.NET 技术来编写一个 BBS，下面是程序的说明：

13.4.1 数据库结构

13.4.1.1 数据库结构文件 (bbs/forum.sql)

```
ALTER TABLE [dbo].[articleinfo] DROP CONSTRAINT  
FK__articlein__FORUM__123EB7A3  
GO  
  
ALTER TABLE [dbo].[FriendInfo] DROP CONSTRAINT  
FK__FriendInf__UserI__0B91BA14  
GO  
  
if exists (select * from sysobjects where id = object_id(N'[dbo].[pro_bt]'))  
and  
OBJECTPROPERTY(id, N'IsProcedure') = 1)  
drop procedure [dbo].[pro_bt]  
GO  
  
if exists (select * from sysobjects where id =  
object_id(N'[dbo].[pro_golistartinfo]')) and  
OBJECTPROPERTY(id, N'IsProcedure') = 1)  
drop procedure [dbo].[pro_golistartinfo]  
GO  
  
if exists (select * from sysobjects where id =  
object_id(N'[dbo].[pro_listartinfo]')) and  
OBJECTPROPERTY(id, N'IsProcedure') = 1)  
drop procedure [dbo].[pro_listartinfo]  
GO  
  
if exists (select * from sysobjects where id =  
object_id(N'[dbo].[pro_userdetail]')) and  
OBJECTPROPERTY(id, N'IsProcedure') = 1)  
drop procedure [dbo].[pro_userdetail]  
GO
```

```
if exists (select * from sysobjects where id =
object_id(N'[dbo].[articleinfo]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[articleinfo]
GO

if exists (select * from sysobjects where id = object_id(N'[dbo].[bbsForum]')
and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[bbsForum]
GO

if exists (select * from sysobjects where id =
object_id(N'[dbo].[BBSUserInfo]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[BBSUserInfo]
GO

if exists (select * from sysobjects where id =
object_id(N'[dbo].[FavoriteInfo]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[FavoriteInfo]
GO

if exists (select * from sysobjects where id =
object_id(N'[dbo].[FriendInfo]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[FriendInfo]
GO

-----
CREATE TABLE [dbo].[articleinfo] (
    [ARTICLEID] [int] IDENTITY (1, 1) NOT NULL ,
    [ARTICLEFATHERID] [int] NULL ,
    [FORUMID] [int] NOT NULL ,
    [BESTID] [int] NULL ,
    [SUBJECT] [varchar] (200) NOT NULL ,
    [CONTENT] [varchar] (2000) NULL ,
    [SENDERID] [int] NULL ,
    [MODIFYID] [int] NULL ,
    [FACEID] [int] NOT NULL ,
    [SUBMITTIME] [datetime] NULL ,
```

```
[LASTMODIFYTIME] [datetime] NULL ,
[IP] [varchar] (20) NULL ,
[STATUS] [int] NULL ,
[VISITS] [int] NULL ,
[FROMFORUMID] [int] NULL ,
[REPLY] [int] NULL ,
[ZIPFILENAME] [varchar] (100) NULL ,
[PERMITEMAIL] [int] NULL ,
[USERNAME] [varchar] (20) NULL ,
[USERNICKNAME] [varchar] (20) NOT NULL ,
[FFID] [int] NULL ,
[CONTENTS] [text] NULL
)
GO
```

```
-----
CREATE TABLE [dbo].[bbsForum] (
    [ForumID] [int] NOT NULL ,
    [ForumFatherID] [int] NOT NULL ,
    [ForumNameC] [varchar] (50) NOT NULL ,
    [ForumNameE] [varchar] (50) NOT NULL ,
    [ForumTitle] [varchar] (200) NULL ,
    [Visits] [int] NULL ,
    [Onlineint] [int] NULL ,
    [Limit] [int] NULL ,
    [Idle] [datetime] NULL ,
    [type] [tinyint] NULL ,
    [Status] [tinyint] NULL ,
    [IsLeaf] [tinyint] NULL ,
    [Bulletin] [varchar] (1000) NULL ,
    [PresidentID] [int] NULL ,
    [PresidentName] [varchar] (20) NULL ,
    [ArticleCount] [int] NULL
)
GO
```

```
-----
CREATE TABLE [dbo].[BBSUserInfo] (
    [UserID] [int] IDENTITY (1, 1) NOT NULL ,
    [UserName] [varchar] (20) NOT NULL ,
    [UserNickName] [varchar] (20) NOT NULL ,
    [OICQ] [int] NULL ,
```

```
[ICQ] [int] NULL ,
[HobbyNo] [int] NULL ,
[Admit] [tinyint] NULL ,
[RegisterDate] [datetime] NULL ,
[Sign1] [varchar] (100) NULL ,
[Sign2] [varchar] (100) NULL ,
[Sign3] [varchar] (100) NULL ,
[PerFectWork] [varchar] (100) NULL ,
[LikeMusic] [varchar] (100) NULL ,
[LikeChat] [varchar] (100) NULL ,
[LikeBook] [varchar] (100) NULL ,
[LikeBall] [varchar] (100) NULL ,
[LikeFilm] [varchar] (100) NULL ,
[LikeMan] [varchar] (20) NULL ,
[LikeArt] [varchar] (100) NULL ,
[LikePlace] [varchar] (100) NULL ,
[Inamorato] [varchar] (20) NULL ,
[Explain] [varchar] (500) NULL ,
[Publish] [smallint] NULL ,
[EXPERIENCE] [int] NULL ,
[HEARTY] [int] NULL ,
[LIFE] [int] NULL ,
[IEPERIENCE] [int] NULL ,
[IHEARTY] [int] NULL ,
[ILIFE] [int] NULL ,
[password] [varchar] (6) NOT NULL ,
[email] [varchar] (50) NOT NULL ,
[likepage] [int] NULL ,
[privacy] [varchar] (2) NULL ,
[birthday] [datetime] NULL ,
[sex] [varchar] (2) NULL ,
[marriage] [varchar] (10) NULL ,
[city] [varchar] (10) NULL ,
[zipcode] [int] NULL ,
[address] [varchar] (50) NULL ,
[teloffice] [varchar] (30) NULL ,
[telhome] [varchar] (30) NULL ,
[telmobile] [varchar] (30) NULL ,
[bp] [varchar] (30) NULL ,
[fax] [varchar] (30) NULL ,
[homepage] [varchar] (50) NULL ,
[income] [money] NULL ,
```

```
    [educationlevelid] [varchar] (20) NULL ,
    [occupationid] [varchar] (50) NULL
)
GO
```

```
-----
CREATE TABLE [dbo].[FavoriteInfo] (
    [SerialNo] [int] NOT NULL ,
    [UserID] [int] NOT NULL ,
    [FavoriteURL] [varchar] (100) NULL ,
    [FavoriteName] [varchar] (500) NOT NULL ,
    [Sort] [tinyint] NULL ,
    [Status] [tinyint] NULL
)
GO
```

```
-----
CREATE TABLE [dbo].[FriendInfo] (
    [UserID] [int] NOT NULL ,
    [FriendID] [int] NOT NULL ,
    [Status] [tinyint] NULL ,
    [FriendName] [varchar] (20) NULL
)
GO
```

```
-----
create procedure pro_bt
    @vi_FFID varchar(10)
as
select articleid,Subject,contents,faceid,username,submittime,visits
from articleinfo where status=0 and forumid=@vi_FFID
GO
```

```
SET QUOTED_IDENTIFIER OFF    SET ANSI_NULLS ON
GO
```

```
-----
create procedure pro_golistartinfo
    @artid varchar(20)
as
select
```

```
articlefatherid,articleid,Subject,contents,faceid,username,submittime,visits from articleinfo
    where status=0 and articleid=@artid and articlefatherid=(select
articlefatherid from articleinfo where articleid=@artid)

GO

SET QUOTED_IDENTIFIER OFF    SET ANSI_NULLS ON
GO
-----
create procedure pro_listartinfo
    @artid varchar(20)
as
select articleid,Subject,contents,faceid,username,submittime,visits
from articleinfo
    where status=0 and articlefatherid=(select articlefatherid from articleinfo
where articleid=@artid)

GO

SET QUOTED_IDENTIFIER OFF    SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER OFF    SET ANSI_NULLS ON
GO
-----
create procedure pro_userdetail
    @unn varchar(20)
as
select username as '用户名称',username as '用户昵称',oicq as '用户oicq号码',
PerFectWork as '最喜爱的工作',
    LikeMusic as '最喜欢的音乐',LikeChat as '最喜欢的聊天室',
    LikeBook as '最喜欢的书',LikeBall as '最喜欢的球队',LikeFilm as '最喜欢的电影',
    LikeMan as '最喜欢的人',
    LikeArt as '最喜欢的艺术',Inamorato as '梦中情人'
from bbsuserinfo where username=@unn

GO

SET QUOTED_IDENTIFIER OFF    SET ANSI_NULLS ON
GO
```

13.4.2 数据库连接

13.4.2.1 数据库连接 bbs/Com/BBSDBLink.cs :

```
using System;
using System.Data;
using System.Data.SqlClient;
namespace bbsDB{
    public class BBSDBLink{
        public SqlConnection bbsConn = new
SqlConnection("server=213.213.213.126;uid=bbs;pwd=;database=bbs");
    }
}
```

13.4.3 用户管理

13.4.3.1 用户组件 GetUser.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using bbsDB; //数据库的连接:
//using bbsDB; //引入名字空间bbsDB;可用完全控制的方式:bbsDB.bbsDB()

namespace memberDB{

    public class GetUser{
        BBSDBLink link = new BBSDBLink();
        //public String Message = "";

        //////////////////////////////////////
        public DataView userdetail(String unn){
            SqlConnection gConn = link.bbsConn;
            SqlDataAdapter lComm = new SqlDataAdapter("pro_userdetail",
gConn);
            //命令集位存储过程:
            lComm.SelectCommand.CommandType = CommandType.StoredProcedure;

            //传递一个论坛ID = @unn 数据:
            lComm.SelectCommand.Parameters.Add(new
SqlParameter("@unn", SqlDbType.VarChar, 20));
            lComm.SelectCommand.Parameters["@unn"].Value = unn;
```

```
//
DataSet ds = new DataSet();
lComm.FillDataSet(ds, "bbsuserinfo");

return ds.Tables["bbsuserinfo"].DefaultView;
}

////////////////////////////////////
public DataView checkuser(String unn,String pws)
{
    SqlConnection cConn = link.bbsConn;
    String SQLStr1 = "select UserID from bbsuserinfo where
USERNICKNAME='"+unn+"' and password='"+pws+"'";
    SQLDataSetCommand cComm = new SQLDataSetCommand(SQLStr1, cConn);

    DataSet ds = new DataSet();
    cComm.FillDataSet(ds, "bbsuserinfo");

    return ds.Tables["bbsuserinfo"].DefaultView;
}
////////////////////////////////////
////////////////////////////////////

/*
public void DataSet InsertUser(String UserName,String UserNickName,int
OICQ, String password,String email){
    SqlConnection gConn = link.bbsConn;

    //values()里面的数据必须与数据库里面的数据的顺序一样！
    String insertCmd = "insert into
BBSUserInfo(UserName,UserNickName,OICQ,password,email) values(@UserName,
@UserNickName,@OICQ, @password,@email)";

    try{
        //创建命令集
        SQLCommand gComm = new SQLCommand(insertCmd, gConn);

        gComm.Parameters.Add(new SqlParameter("@UserName",
SQLDataType.VarChar, 20));
        gComm.Parameters["@UserName"].Value = UserName;

        gComm.Parameters.Add(new SqlParameter("@UserNickName",
```



```
SQLDataType.VarChar, 20));
    gComm.Parameters["@UserNickName"].Value = UserNickName;

    gComm.Parameters.Add(new SqlParameter("@email",
SQLDataType.VarChar, 50));
    gComm.Parameters["@email"].Value = email;

    gComm.Parameters.Add(new SqlParameter("@OICQ", SQLDataType.VarChar,
10));
    gComm.Parameters["@OICQ"].Value = OICQ;

    gComm.Parameters.Add(new SqlParameter("@password",
SQLDataType.VarChar, 6));
    gComm.Parameters["@password"].Value = password;

    gComm.ActiveConnection.Open();
    gComm.ExecuteNonQuery();
    Message = "用户注册：" + insertCmd.ToString();
} catch (SQLException e) {
    if (e.Number == 2627)
        Message = "一个记录已经存在：";
    else
        Message = "不能插入数据：";
}
gComm.ActiveConnection.Close();
BindGrid();
}

//列出用户信息：
public DataView BindGrid(String unn){
    SqlConnection gConn = link.bbsConn;
    SQLDataSetCommand gComm = new SQLDataSetCommand("select BBSUserInfo
as '用户ID',UserName as '用户名称',UserNickName as '用户昵称', email as '电子邮件
',OICQ from testbbs", gConn);
    DataSet ds = new DataSet();
    gComm.FillDataSet(ds, "BBSUserInfo");
    return ds.Tables["BBSUserInfo"].DefaultView;
}
*/

//////////////////////////////////////类空间
}
```

```
////////////////////////////////////名字空间  
}
```

13.4.3.2 注册页面 (bbs/User/login.aspx):

```
<%@ page language="c#" %>  
<%@ Import Namespace="System" %>  
<%@ Import Namespace="System.Web" %>  
<%@ Import Namespace="System.Web.UI" %>  
<%@ Import Namespace="System.Text" %>  
  
<html>  
<head>  
<title>用户注册!</title>  
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">  
<link rel="stylesheet" href="../images/bbs2.css">  
</head>  
<body bgcolor="#F0F0FF">  
<br>  
    <table width="45%" cellpadding="0" cellspacing="0" height="25"  
align="center">  
    <tr valign="top">  
        <td colspan="2" height="30"><font color="#FFFFFF"><b><font  
color="#000000" size="3">用户注册  
  
</font></b></font></td>  
    </tr>  
</table>  
<form method="post" action="deallogin.aspx" runat=server>  
<table width="45%" cellpadding="0" cellspacing="0" align="center">  
<tr valign="top">  
<td height="45" colspan="2">  
    <input type="hidden" id="step" value=1>  
    <br>  
</td>  
</tr>  
<tr valign="top">  
<td height="72" colspan="2">  
    <div align="center">  
    <!-----用户名开始----->  
    <input type="text" id="username" size="25" maxlength=20 columns=20  
runat=server>
```



```
<td>声明：<br>
<br>
<br>
```

在参加天极讨论区之前请您阅读并同意下列条款：

<p> 1. 遵守中国国家信息产业部发布的《互联网电子公告服务管理规定》里的一切规定以及中华人民共和国的各项有关法律法规:

不得发布危害国家稳定和安全的言论；

不得发布人身攻击和不负责任的信息；

不得发布黄色及内容不健康的信息；

不得做出危害本网站系统安全的行为；

2. 承担一切因您的行为而直接或间接导致的民事或刑事法律责任；

3. 天极网各讨论区的版主有权保留或删除其管辖范围内的任意内容；

4. 您在天极 FAQ 发表的作品，天极网有权在网站内转载或引用</p>

<p> 进入讨论区发言即表明您已经阅读并接受上述条款。</p>

<p> 一个浏览者如果要发表信息，则必须首先登记用户名。用户名登记以后，其他用户将不能再使用相同的用户登记了，而且只有用户进行了登记，才能根据自己的需要进行显示风格的修改等。

天极网对注册名的限制是

1. 不能使用任何不良的文字，如：色情、反动、宗教迷信、伪科学、有害于民族感情、正常邦交以及世界和平等文字。

2. 不能注册含有：国家领导人、海内外知名人士、文化遗产、重要设施以及国家政府机关等文字的用户名。

3. 不能注册蓄意攻击他人的注册名。

4. 注册名内不能包含有空格以及特殊符号

我们将定期审核用户名，将定期对电子公告牌的注册用户名进行检查，一旦发现问题就会立即取消该用户资格并删除该用户名，并对该用户的登记注册资料另行备份。

在登记中，您必须提供一个您的用户名，密码，长期有效的邮件地址(私有邮件地址)等信息。</p>

<p> 私有邮件地址是系统使用的，在您要求将您发布的信息有回应信息时发送给您通知，或您的密码忘记系统自动将密码发送给您等，都使用这个私有邮件地址。而公开邮件地址是供其他用户浏览您的信息时显示的，您可以不输入，但这样其他用户如果希望给您发送邮件将不能得到您的邮件地址。


```
</p>
</td>
</tr>
</table>
```

```
<br>
<table width="50%" cellspacing="0" cellpadding="0" align="center"
height="2">
  <tr bgcolor="#006666">
    <td height="1"></td>
  </tr>
</table>
<table width="50%" cellspacing="0" cellpadding="0" align="center"
height="1">
  <tr>
    <td height="1"></td>
  </tr>
</table>
<table width="50%" cellspacing="0" cellpadding="0" align="center"
height="1">
  <tr bgcolor="#006666">
    <td height="1"></td>
  </tr>
</table>

<table width="%" cellspacing="0" cellpadding="0" align="center"
height="1">
  <tr>
    <td height="1"></td>
  </tr>
</table>
<table width="50%" cellspacing="0" cellpadding="0" align="center"
height="1">
  <tr bgcolor="#006666">
    <td height="1"></td>
  </tr>
</table>
<br>
<p>
</body>
</html>
```

13.4.3.3 注册页面 2 (bbs/user/deallogin.aspx):

```
<%@ page language="c#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SQL" %>
<%@ Import Namespace="System.Web" %>
<%@ Import Namespace="System.Web.UI" %>
<%@ Import Namespace="System.Text" %>
<%@ Import Namespace="bbsDB"%>

<html>
<head>
<title>天极论坛系统.NET</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link rel="stylesheet" href="../images/bbs2.css">
</head>
<script language="c#" runat=server>

    protected void Page_Load(Object Src, EventArgs E){
        gConn = oConn.bbsConn;
        String UName = Request.QueryString["UserName.Value"];
        String sqlt="select * from BBSUserInfo where username='" UName "'";
        SqlCommand lcom = new SqlCommand(sqlt,gConn);

        lcom.ActiveConnection.Open();
        try {
            lcom.ExecuteNonQuery();
        }

        Catch(){

        }
        lcom.ActiveConnection.Close();
    }

</script>

<body bgcolor="#ffffff">
<!-- #include virtual="../IncludeFiles/head.inc" -->
<br>
<table width="80%" cellpadding="0" cellspacing="0" align="center">
```

```

<tr>
  <td height="24" bgcolor="#aaaadd">
    <font>请输入你的个人信息，带 的选项是必填选项:</font>
  </td>
</tr>
</table>

<form action="slogin.aspx" method="post" runat=server>

  <table width="80%" cellspacing="0" cellpadding="0" border="1"
align="center">
  <tr>
    <td> 用户名称 :</td>
    <td><input type="text" id="UserName" value="<%= UName%>"
size="25" runat=server      readonly>
    </td>
  </tr>
  <tr>
    <td> 用户昵称 :</td>
    <td><input type="text" id="UserNName" size="25" runat=server>
<asp:requiredfieldvalidator id=Requiredfieldvalidatorn
display=Dynamic
      controltovalidate="UserNName" runat=Server>
      不能为空 !
    </asp:requiredfieldvalidator>
    </td>
  </tr>
  <tr>
    <td> 输入密码 :</td>
    <td><input type="Password" id="UserPassword" size="25"
runat=server>
    <font color=red>*</font> 6个字符
    <asp:requiredfieldvalidator id=Requiredfieldvalidator1
display=Dynamic
      controltovalidate="UserPassword" runat=Server>
      不能为空 !
    </asp:requiredfieldvalidator>
    <asp:regularexpressionvalidator
id=Regularexpressionvalidator1 display=Dynamic
      controltovalidate="UserPassword" runat=Server
validationexpression="^[^]{4,10}">
      密码非法 !
  </td>
  </tr>
</table>

```

```
        </asp:regularexpressionvalidator>
    </td>
</tr>
<tr>
    <td> 确认密码: </td>
    <td><input type="Password" id="PasswordSure" size="25"
runat=server>
        <font color=red>*</font>
        <asp:comparevalidator id="comPassword" display=Dynamic
        controltocompare="UserPassword"
controltovalidate="PasswordSure"                runat=Server>
            两次录入的密码不同!
        </asp:comparevalidator>
    </td>
</tr>
<tr>
    <td> 电子邮件: </td>
    <td><input type="text" id="Email" size="25" runat=server>
        <asp:requiredfieldvalidator id=Requiredfieldvalidatoremail
display=Dynamic
        controltovalidate="Email" runat=Server>
            不能为空!
        </asp:requiredfieldvalidator>
        <asp:regularexpressionvalidator id="regEmail" display=Dynamic
        controltovalidate="Email"
validationexpression="[^\']*" runat=Server>
            非法字符
        </asp:regularexpressionvalidator>
    </td>
</tr>
<tr>
    <td> 提示问题: <br>
    <td><input type="text" id="HintQuestion" size="25" runat=server>
        <asp:requiredfieldvalidator id=Requiredfieldvalidatorihi
display=Dynamic
        controltovalidate="HintQuestion" runat=Server>
            不能为空!
        </asp:requiredfieldvalidator>
    </td>
</tr>
<tr>
    <td> 问题答案: </td>
```



```

        </form>

        <br>
        <!-- #include virtual="../IncludeFiles/buttom.inc" -->
    </body>
</html>

```

13.4.3.4 成功注册页面 (bbs/User/slogin.aspx):

```

<%@ page language="c#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SQL" %>
<%@ Import Namespace="System.Web" %>
<%@ Import Namespace="System.Web.UI" %>
<%@ Import Namespace="System.Text" %>
<%@ Import Namespace="bbsDB"%>

<html>
<head>
<title>天极论坛系统.NET</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link rel="stylesheet" href="../images/bbs2.css">
</head>
<script language="C#" runat="server">
    SqlConnection gConn;
    BBSDBLink oConn = new BBSDBLink();

    //页面传送:
    protected void Page_Load(Object Src, EventArgs E){
        gConn = oConn.bbsConn;
        String UName = Request.QueryString["UserName.Value"];
    }

    //响应鼠标事件:
    void SubmitBtn_Click(object Source, EventArgs e){
        //创建SQL语句
        String SQLStr = "insert into BBSUserInfo(UserName,UserNickName,
password,"
+ "email,Sign1,Sign2,likepage,privacy,birthday,sex,marriage,city,zipcode,"
+ "address,teloffice,telhome,telmobile,bp,fax,homepage,educationlevelid,o

```

```
ccupationid)"
        +"
values(@UserName,@UserNName,@UserPassword,@Email,@HintQuestion,@HintAnswer,"

        + "@CountPage,@privacy,@Birth,@Sex,@Marriage,@City,@ZipCode,@Address,@Tel
Office,"

        + "@TelHome,@TelMobile,@BP,@Fax,@HomePage,@EducationLevelID,@OccupationID
)";

        //创建命令集 :
        SQLCommand gComm = new SQLCommand(SQLStr, gConn);

        //23个数据字段的插入 :
        //@username
        gComm.Parameters.Add(new SqlParameter("@UserName", SqlDbType
.VarChar, 20));
        gComm.Parameters["@UserName"].Value = Request.QueryString["User
Name.Value"];

        //@UserNName
        gComm.Parameters.Add(new SqlParameter("@UserNName", SqlDbType
.VarChar, 20));
        gComm.Parameters["@UserNName"].Value = Request.QueryString["User
NName.Value"];

        //@UserPassword
        gComm.Parameters.Add(new SqlParameter("@UserPassword", SQLData
Type.VarChar, 6));
        gComm.Parameters["@UserPassword"].Value = Request.QueryString
["UserPassword.Value"];

        //@Email
        gComm.Parameters.Add(new SqlParameter("@Email", SqlDbType.Var
Char, 50));
        gComm.Parameters["@Email"].Value = Request.QueryString["Email
.Value"];

        //@HintQuestion
        gComm.Parameters.Add(new SqlParameter("@HintQuestion", SQLData
Type.VarChar, 100));
        gComm.Parameters["@HintQuestion"].Value = Request.QueryString
```

```
["HintQuestion.Value"];

        //@HintAnswer
        gComm.Parameters.Add(new SqlParameter("@HintAnswer", SqlDbType
.VarChar, 100));
        gComm.Parameters["@HintAnswer"].Value = Request.QueryString["Hint
Answer.Value"];

        //@CountPage
        gComm.Parameters.Add(new SqlParameter("@CountPage", SqlDbType
.VarChar, 4));
        gComm.Parameters["@CountPage"].Value = Request.QueryString["Count
Page.Value"];

        //@privacy
        gComm.Parameters.Add(new SqlParameter("@privacy", SqlDbType.Var
Char, 2));
        gComm.Parameters["@privacy"].Value = Request.QueryString["privacy
.Value"];

        //@Birth
        gComm.Parameters.Add(new SqlParameter("@Birth", SqlDbType.Var
Char, 20));
        gComm.Parameters["@Birth"].Value = Request.QueryString["Birth
.Value"];

        //@Sex
        gComm.Parameters.Add(new SqlParameter("@Sex", SqlDbType.VarChar,
2));
        gComm.Parameters["@Sex"].Value = Request.QueryString["Sex
.Value"];

        //@Marriage
        gComm.Parameters.Add(new SqlParameter("@Marriage", SqlDbType
.VarChar, 10));
        gComm.Parameters["@Marriage"].Value = Request.QueryString
["Marriage.Value"];

        //@City
        gComm.Parameters.Add(new SqlParameter("@City", SqlDbType.Var
Char, 10));
        gComm.Parameters["@City"].Value = Request.QueryString["City.
```

```
Value"];

        //@ZipCode
        gComm.Parameters.Add(new SqlParameter("@ZipCode", SqlDbType
.VarChar, 4));
        gComm.Parameters["@ZipCode"].Value = Request.QueryString["Zip
Code.Value"];

        //@Address
        gComm.Parameters.Add(new SqlParameter("@Address", SqlDbType
.VarChar, 50));
        gComm.Parameters["@Address"].Value = Request.QueryString
["Address.Value"];

        //@TelOffice
        gComm.Parameters.Add(new SqlParameter("@TelOffice", SqlDbType
.VarChar, 30));
        gComm.Parameters["@TelOffice"].Value = Request.QueryString["Tel
Office.Value"];

        //@TelHome,
        gComm.Parameters.Add(new SqlParameter("@TelHome", SqlDbType.Var
Char, 30));
        gComm.Parameters["@TelHome"].Value = Request.QueryString["TelHome
.Value"];

        //@TelMobile
        gComm.Parameters.Add(new SqlParameter("@TelMobile", SqlDbType
.VarChar, 30));
        gComm.Parameters["@TelMobile"].Value = Request.QueryString["Tel
Mobile.Value"];

        //@BP
        gComm.Parameters.Add(new SqlParameter("@BP", SqlDbType.VarChar,
30));
        gComm.Parameters["@BP"].Value = Request.QueryString["BP.Value"];

        //@Fax
        gComm.Parameters.Add(new SqlParameter("@Fax", SqlDbType.VarChar,
30));
        gComm.Parameters["@Fax"].Value = Request.QueryString["Fax
```



```
.Value"];

        //@HomePage
        gComm.Parameters.Add(new SqlParameter("@HomePage", SqlDbType
.VarChar, 50));
        gComm.Parameters["@HomePage"].Value = Request.QueryString["Home
Page.Value"];

        //@Income
        //gComm.Parameters.Add(new SqlParameter("@Income", SqlDbType
.Money));
        //gComm.Parameters["@Income"].Value = Request.QueryString["Income
.Value"];

        //@EducationLevelID
        gComm.Parameters.Add(new SqlParameter("@EducationLevelID", SQL
DataType.VarChar, 20));
        gComm.Parameters["@EducationLevelID"].Value = Request.Query
String["EducationLevelID.Value"];

        //@OccupationID
        gComm.Parameters.Add(new SqlParameter("@OccupationID", SQLData
Type.VarChar, 50));
        gComm.Parameters["@OccupationID"].Value = Request.QueryString
["OccupationID.Value"];
        //
        gComm.ActiveConnection.Open();
    try{
        gComm.ExecuteNonQuery();
        Message.InnerHtml = "<b>Record Added</b><br>" + "注册成功!!!";
    }catch (SQLException se){
        if (se.Number == 2627)
            Message.InnerHtml = "一个记录已经存在!";
        else
            Message.InnerHtml = "不能插入数据!";
        Message.Style["color"] = "red";
    }
    gComm.ActiveConnection.Close();
}

</script>
```

```
<body bgcolor="#ffffff">
<!-- #include virtual="../IncludeFiles/head.inc" -->
<br><br>
<center>
    注册成功! <a href="..">
</center>
<br><br>
    <span id="Message" MaintainState="false" style="font: arial 11pt;"
runat="server"/>
    <span id="Span1" style="color:red" runat=server></span>

<br>
<!-- #include virtual="../IncludeFiles/buttom.inc" -->
</body>
</html>
```

13.4.3.5 用户详细资料 userdetail.aspx :

```
<%@ page language="c#" %>
<%@ Import Namespace="bbsDB" %>
<%@ Import Namespace="memberDB" %>

<html>
<script language="C#" runat="server">
    GetUser gu = new GetUser();
    public String unn;

    protected void Page_Load(Object Sender, EventArgs e)
    {
        unn = Request.QueryString["username"];
        listarticle.DataSource=gu.userdetail(unn);
        listarticle.DataBind();
    }
</script>

<body>
<!-- #include virtual="../IncludeFiles/head.inc" -->
<br>
    <div align="center">
        <hr width="80%">
        <a href="postarticle.aspx">我要发言! </a>
        <hr width="80%">
```

```
<h3><font face="Verdana">通过存储过程的得调用:</font></h3>

<table width="80%">
  <tr>
    <td colspan="3" style="padding-top:20">

      <ASP:DataGrid id="listarticle" runat="server"
      Width="700"
      BackColor="#ccccff"
      BorderColor="black"
      ShowFooter="false"
      CellPadding=3
      CellSpacing="0"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      MaintainState="false"
      />

    </td>
  </tr>
</table>
</div>
<p>
<!-- #include virtual="../IncludeFiles/buttom.inc" -->
</body>
</html>
```

13.4.4 文章信息

13.4.4.1 列出文章组件 bbs/Com/listarticle.cs :

```
using System;
using System.Data;
using System.Data.SqlClient;
using bbsDB;
namespace article{
  public class listarticle{
    public BBSDBLink link = new BBSDBLink();
    //文章的显示:
    public DataView listart(String vi_FFID){
```

```
        SqlConnection gConn = link.bbsConn;
        SQLDataSetCommand lComm = new SQLDataSetCommand("pro_bt", gConn);

        //命令集位存储过程：
        lComm.SelectCommand.CommandType = CommandType.StoredProcedure;

        //传递一个论坛ID = @vi_FFID 数据：
        lComm.SelectCommand.Parameters.Add(new
SQLParameter("@vi_FFID", SqlDbType.VarChar, 10));
        lComm.SelectCommand.Parameters["@vi_FFID"].Value = vi_FFID;

        //
        DataSet ds = new DataSet();
        lComm.FillDataSet(ds, "articleinfo");

        return ds.Tables["articleinfo"].DefaultView;
    }

    //根据文章id选择出文章信息：
    public DataView listartinfo(String artid){
        SqlConnection gConn = link.bbsConn;
        SQLDataSetCommand lComm = new
SQLDataSetCommand("pro_listartinfo", gConn);

        //命令集位存储过程：
        lComm.SelectCommand.CommandType = CommandType.StoredProcedure;

        //传递一个论坛ID = @vi_FFID 数据：
        lComm.SelectCommand.Parameters.Add(new
SQLParameter("@artid", SqlDbType.VarChar, 10));
        lComm.SelectCommand.Parameters["@artid"].Value = artid;

        //
        DataSet ds = new DataSet();
        lComm.FillDataSet(ds, "articleinfo");

        return ds.Tables["articleinfo"].DefaultView;
    }

    //根据文章id选择出文章信息(当从论坛首页进入发表文章页面的时候调用这个函数)：
    public DataView golistartinfo(String artid){
```

```

        SqlConnection gConn = link.bbsConn;
        SQLDataSetCommand lComm = new
SQLDataSetCommand("pro_golistartinfo", gConn);
        //命令集位存储过程：
        lComm.SelectCommand.CommandType = CommandType.StoredProcedure;

        //传递一个论坛ID = @vi_FFID 数据：
        lComm.SelectCommand.Parameters.Add(new
SQLParameter("@artid", SqlDbType.VarChar, 10));
        lComm.SelectCommand.Parameters["@artid"].Value = artid;

        //
        DataSet ds = new DataSet();
        lComm.FillDataSet(ds, "articleinfo");

        return ds.Tables["articleinfo"].DefaultView;
    }

    ///类何名字空间
}

}

```

13.4.4.2 列出文章 bbs/article/listarticle.aspx :

```

<%@ page language="c#" %>
<%@ Import Namespace="bbsDB" %>
<%@ Import Namespace="article" %>

<html>
<script language="C#" runat="server">
    listarticle la = new listarticle();
    public String vi_FFID;

    protected void Page_Load(Object Sender, EventArgs e)
    {
        vi_FFID = Request.QueryString["forumid"];
        //int fid = (int)n;
        listarticle.DataSource=la.listart(vi_FFID);
        //String face;
        //face = Request.QueryString("faceid");
        listarticle.DataBind();
    }

```

```
}
/*
////////////////////////////////////

//public DataView listart(String vi_FFID){
    SqlConnection gConn = link.bbsConn;
    SQLDataSetCommand lComm = new SQLDataSetCommand("BuildTree", gConn);
    //命令集位存储过程：
    lComm.SelectCommand.CommandType = CommandType.StoredProcedure;

    //传递一个论坛ID = @vi_FFID 数据：
    lComm.SelectCommand.Parameters.Add(new
SQLParameter("@vi_FFID", SqlDbType.VarChar, 10));
    lComm.SelectCommand.Parameters["@vi_FFID"].Value = vi_FFID;

    //
    DataSet ds = new DataSet();
    lComm.FillDataSet(ds, "articleinfo");

    MyDataGrid.DataSource=ds.Tables["articleinfo"].DefaultView;
    MyDataGrid.DataBind();

    //return ds.Tables["articleinfo"].DefaultView;
}

////////////////////////////////////
*/
</script>

<body>
<!-- #include virtual="../IncludeFiles/head.inc" -->
<br>
<div align="center">
    <hr width="80%">
    <a href="postarticle.aspx?articleid=-1">
    我要发言！</a>
    <hr width="80%">
    <h3><font face="Verdana">我的论坛列表:</font></h3>

    <table width="80%">
    <tr>
```

```

        <td colspan="3" style="padding-top:20">
        <ASP:Repeater id="listarticle" runat="server">

        <template name="itemtemplate">

        <tr style="background-color:ffffff">

        <td>
            .gif">
                &nbsp;&nbsp;&nbsp;<a href="postarticle.aspx?forumid=<#=
vi_FFID%>&articleid=<# DataBinder.Eval(Container.DataItem, "articleid") %>">
                    <# DataBinder.Eval(Container.DataItem, "Subject"
%></a>&nbsp;&nbsp;&nbsp;<a href="../ManageUser/userdetail.aspx?username=<#
DataBinder.Eval(Container.DataItem, "username") %>">
                        <# DataBinder.Eval(Container.DataItem, "username")
%></a>&nbsp;&nbsp;&nbsp;<# DataBinder.Eval(Container.DataItem, "submittime") %>&nbsp;&nbsp;&
                        <# DataBinder.Eval(Container.DataItem, "visits") %>
                    </td>
                </tr>

        </template>

        </ASP:Repeater>

        </td>
        </tr>
    </table>
</div>
<p>
<!-- #include virtual="../IncludeFiles/buttom.inc" -->
</body>
</html>

```

13.4.4.3 发表文章(bbs/aarticle/postarticle.aspx) :

```

<%@ page language="c#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="bbsDB" %>

```

```
<%@ Import Namespace="article" %>
<%@ Import Namespace="memberDB" %>

<html>

<script language="C#" runat="server">
    public String artid;

    protected void Page_Load(Object Sender,EventArgs e){
        listarticle lar = new listarticle();
        String artid = Request.QueryString["articleid"];
        listart.DataSource = lar.listartinfo(artid);
        listart.DataBind();

        listcontent.DataSource = lar.golistartinfo(artid);
        listcontent.DataBind();

        //listarticle.DataSource =
gu.checkuser(username.Value,password.Value);
        //listarticle.DataBind();

        //input.DataSource = lar.golistartinfo(artid);
        //input.DataBind();

    }

    //提交:
    void AddAuthor_Click(object Source, EventArgs e) {
        BBSDBLink db = new BBSDBLink();
        String SQLStr = "insert into
articleinfo(FORUMID,articlefatherid,USERNICKNAME,SUBJECT,FACEID,CONTENTS)"
        + "
values('12',@articlefatherid,@username,@subject,@faceid,@contents)";
        SqlConnection pConn = db.bbsConn;
        SqlCommand pComm = new SqlCommand(SQLStr,pConn);
        //
        pComm.Parameters.Add(new SqlParameter("@username",
SQLDataType.VarChar, 20));
        pComm.Parameters["@username"].Value = username.Value;

        //
        //pComm.Parameters.Add(new
```



```
SqlParameter("@forumid", SqlDbType.VarChar, 4));
    //pComm.Parameters["@forumid"].Value = forumid.Value;

    //
    pComm.Parameters.Add(new
SqlParameter("@articlefatherid", SqlDbType.VarChar, 4));
    pComm.Parameters["@articlefatherid"].Value =
articlefatherid.Value;
    //pComm.Parameters["@articlefatherid"].Value =
Request.QueryString["articleid"];
    //
    pComm.Parameters.Add(new
SqlParameter("@subject", SqlDbType.VarChar, 200));
    pComm.Parameters["@subject"].Value = subject.Value;

    //
    pComm.Parameters.Add(new
SqlParameter("@faceid", SqlDbType.VarChar, 4));
    pComm.Parameters["@faceid"].Value = faceid.SelectedItem.Value;

    //
    pComm.Parameters.Add(new
SqlParameter("@contents", SqlDbType.Text, 16));
    pComm.Parameters["@contents"].Value = contents.Value;

    pComm.ActiveConnection.Open();
    try{
        GetUser gu = new GetUser();
        //DataSource = gu.checkuser(username.Value,password.Value);
        if ((gu.checkuser(username.Value,password.Value)) != null){
            pComm.ExecuteNonQuery();
            Message.InnerHtml = "发表文章成功!!!";
            //Response.Redirect("listarticle.aspx?forumid=forumid.Value");
        }
        else{
            Message.InnerHtml = "用户名何密码错误!!!!";
        }
    }catch(SQLException se){
        Message.InnerHtml = "不能插入数据!";
        Message.Style["color"] = "red";
    }
    pComm.ActiveConnection.Close();
```



```
        &nbsp;<a href="postarticle.aspx?articleid=<#  
DataBinder.Eval(Container.DataItem, "articleid") %>">  
        <# DataBinder.Eval(Container.DataItem, "Subject")  
%></a>&nbsp;<br/>(  
        <a href=" ../ManageUser/userdetail.aspx?username=<#  
DataBinder.Eval(Container.DataItem, "username") %>">  
        <# DataBinder.Eval(Container.DataItem, "username")  
%></a>&nbsp;<br/>(  
        <# DataBinder.Eval(Container.DataItem, "submittime") %>&nbsp;<br/><# DataBinder.Eval(Container.DataItem, "visits") %>  
        </td>  
    </tr>  
    </template>  
    </ASP:Repeater>  
</td>  
</tr>  
</table>  
</center>  
</div>  
  
<div align="center">  
    <center>  
  
    <form runat="server">  
        <h3><font face="Verdana">发表你的文章吧! </font></h3>  
        <table width="95%" >  
            <tr>  
                <td valign="top">  
  
                <table style="font: 8pt verdana">  
                    <tr>  
                        <td colspan="2" bgcolor="#aaaadd" style="font:10pt verdana">  
注意你的言论有没有违反国家法律! </td>  
                    </tr>  
  
                <input type="hidden" id="articlefatherid" value="12" runat="server">  
  
                <tr>  
                    <td nowrap>用户名称: </td>  
                    <td><input type="text" id="username" value="" runat="server">
```

不能为空！

```

        <asp:requiredfieldvalidator id=Requiredun display=Dynamic
            controtovalidate="username" runat=Server>
            不能为空！
        </asp:requiredfieldvalidator></td>
    </tr>

```

```

    <tr>
        <td nowrap>用户口令: </td>
        <td><input type="password" id="password" value=""
runat="server">不能为空！

```

```

        <asp:requiredfieldvalidator id=Requiredp display=Dynamic
            controtovalidate="password" runat=Server>
            不能为空！
        </asp:requiredfieldvalidator></td>
    </tr>

```

```

    <tr>
        <td nowrap>文章标题: </td>
        <td><input type="text" id="subject" value="" runat="server">
不能为空！

```

```

        <asp:requiredfieldvalidator id=Requiresd display=Dynamic
            controtovalidate="subject" runat=Server>
            不能为空！
        </asp:requiredfieldvalidator></td>
    </tr>

```

```

    <tr>
        <td>面部表情: </td>
        <td bgcolor="#ccccff">
<div align='center'>
        <asp:RadioButtonList id=faceid RepeatDirection = Horizontal
runat="server">
            <asp:ListItem value="20">
<img alt=大哭 height=15 src='../images/20.gif' width=15>
            </asp:ListItem>
            <asp:ListItem value="21">
                <img alt=恐惧 height=15 src='../images/21.gif' width=15>
            </asp:ListItem>

            <asp:ListItem value="22">
                <img alt=昏倒 height=15 src='../images/22.gif' width=15>
            </asp:ListItem>

```

```
<asp:ListItem value="23">
<img alt=玩酷 height=15 src='../images/23.gif' width=15>
</asp:ListItem>

<asp:ListItem value="24">
<img alt=有趣 height=15 src='../images/24.gif' width=15>
</asp:ListItem>

<asp:ListItem value="25">
<img alt=滑稽 height=15 src='../images/25.gif' width=15>
</asp:ListItem>

<asp:ListItem value="26">
<img alt=不宜 height=15 src='../images/26.gif' width=15>
</asp:ListItem>

<asp:ListItem value="17">
<img alt=大笑 height=15 src='../images/17.gif' width=15>
</asp:ListItem>

<asp:ListItem value="27">
<img alt=喜欢 height=15 src='../images/27.gif' width=15>
</asp:ListItem>

<asp:ListItem value="31">
<img alt=疑惑 height=15 src='../images/31.gif' width=14>
</asp:ListItem>

<asp:ListItem value="34">
<img alt=呐喊 height=17 src='../images/34.gif' width=17>
</asp:ListItem>

<asp:ListItem value="37">
<img alt=色相 src='../images/37.gif'>
</asp:ListItem>
</asp:RadioButtonList>
</div> </td><td>不能为空！
    <asp:requiredfieldvalidator id=Requiredfaceid display=Dynamic
        controltovalidate="faceid" runat=Server>
        不能为空！
    </asp:requiredfieldvalidator>
```

```

        </td>

        </tr>
        <tr>
            <td>言论内容: </td>
            <td><textarea cols="66" rows="10" name="contents"
id="contents" value="" runat="server"></textarea></td>
        </tr>
        <tr>
            <td ></td>
            <td align="center" style="padding-top:15">
                <input type="submit" value="提交"
OnServerClick="AddAuthor_Click" runat="server">
                <input type="reset" value="重写" runat="server">

                <span id="Message" MaintainState="false" style="font:
arial 11pt;" runat="server"/>

            </td>
        </tr>
    </table>
</td>
</tr>
</table>
<p>
    <span id="Span1" style="color:red" runat=server></span>
</form>
</center>
</div>
<!-- #include virtual="../../../IncludeFiles/buttom.inc" -->
</body>
</html>

```

在这里没有用到通常的 BBS 的树形结构，只侧重讲应用.NET 技术来构建一个 BBS。不过树形式很容易实现的，一个递归，再加上 HTML 语言的标识就可以实现了，在具体写程序的时候注意如.MoveNext 这样的语句啊，不要死循环，这里就不多说了。

13.4.5 运行结果

下面是 bbs 的运行结果，如图 13-5 所示。



图 13-5

加上递归算法，这就是一个网上论坛了，不过不仅仅是递归算法，其他的也可以实现的。

第 14 章 C#的 Windows 编程

C#不仅仅在 Internet 上应用，而且在 Windows 上也有出色的表现，下面我们来看看这个程序，这个程序的目的是对数据库进行浏览、增加、删除的操作。

14.1 表 结 构

四个表存放产品信息、用户信息等，键下面的结构，建立在微软的 Access 数据库上面，book.mdb，结构如图 14-1。

The figure displays four Access database table structures:

- Categories**:
 - CategoryID
 - CategoryName
- ProductDetails**:
 - ProductID
 - Name
 - Grans
 - [Percent]
- Customers**:
 - CustomerID
 - FirstName
 - LastName
 - Address
 - City
 - State
 - Zip
 - HomePhone
 - WorkPhone
 - Email
 - Reference
 - FamilyCount
 - PetCount
 - CardNumber
 - CardType
 - Expiration
 - CardName
- Products**:
 - ProductID
 - CategoryID
 - ProductName
 - ProductDescription
 - UnitPrice
 - ImagePath
 - ServingSize
 - Servings
 - Quantity
 - MinOnHand
 - MaxOnHand
 - Manufacturer

图 14-1

14.2 程 序

本程序中，要用到连接 Assecc 数据库的连接串字符，C#连接 Access 的字符串为：

```
string strCon="Provider=Microsoft.Jet.OLEDB.4.0 ;Data Source=book.mdb" ;  
ADOConnection myConn = new ADOConnection(strCon) ;
```

之后打开连接：

```
myConn.Open() ;
```

记得完了之后要关闭数据库连接啊：


```
myConn.Close() ;
```

对应于不同的操作，有相关的函数，下面是完整的代码文件：

```
(dataedit\dataedit.cs)
```

```
namespace SaurabhData {

    using System;
    using System.Drawing;
    using System.ComponentModel;
    using System.Windows.Forms;
    using System.Data.ADO;
    using System.Data;

    /// <summary>
    /// Class for viewing , editing and deleting data in a MsAccess 2000
database book.mdb.
    /// </summary>
    public class DataEdit : System.Windows.Forms.Form {

        /// <summary>
        /// Required by the Win Forms designer
        /// </summary>
        private System.ComponentModel.Container components;
        private System.Windows.Forms.Button delete;
        private System.Windows.Forms.Button update;
        private System.Windows.Forms.Button helpme;
        private System.Windows.Forms.Button lastrec;
        private System.Windows.Forms.Button nextrec;
        private System.Windows.Forms.Button previousrec;
        private System.Windows.Forms.Button firstrec;
        private System.Windows.Forms.TextBox t_bookstock;
        private System.Windows.Forms.TextBox t_bookprice;
        private System.Windows.Forms.TextBox t_bookauthor;
        private System.Windows.Forms.TextBox t_booktitle;
        private System.Windows.Forms.TextBox t_bookid;
        private System.Windows.Forms.Label l_bookstock;
        private System.Windows.Forms.Label l_bookprice;
        private System.Windows.Forms.Label l_bookauthor;
        private System.Windows.Forms.Label l_booktitle;
        private System.Windows.Forms.Label l_bookid;
        private System.Windows.Forms.Label labell1;
        private System.Windows.Forms.StatusBar statusBar;
```

```
private System.Data.DataSet myDataSet ;
private ListManager myListManager;
private bool isBound=false;

public DataEdit() {

    // Required for Win Form Designer support
    InitializeComponent();

    //Connect to the DataBase.
    GetConnected() ;

}

/// <summary>
///     Clean up any resources being used
/// </summary>
public override void Dispose() {
    base.Dispose();
    components.Dispose();
}

///<summary>
///     This method connects to the database and returns a Dataset
Object.
///</summary>
public void GetConnected()
{
    //ADOConnection myConn;
    try{
        statusBar.Text="正在连接数据库...";
        //建立一个ADO连接
        string strCon="Provider=Microsoft.Jet.OLEDB.4.0 ;Data
Source=book.mdb" ;
        ADOConnection myConn    =new ADOConnection(strCon) ;

        string strCom="SELECT * FROM bookstock" ;
        //创建一个DataSet
        myDataSet = new DataSet() ;

        myConn.Open() ;
```

```
        //运行SQL语句
        ADODataSetCommand myCommand = new
ADODataSetCommand(strCom,myConn) ;
        //填充
        myCommand.FillDataSet(myDataSet,"bookstock") ;
        //关闭数据库连接
        myConn.Close() ;

        //绑定
        if(!isBound)
        {
            t_bookid.Bindings.Add("Text",
myDataSet.Tables["bookstock"], "bookid");
            t_booktitle.Bindings.Add("Text",
myDataSet.Tables["bookstock"], "booktitle");
            t_bookauthor.Bindings.Add("Text",
myDataSet.Tables["bookstock"], "bookauthor");
            t_bookprice.Bindings.Add("Text",
myDataSet.Tables["bookstock"], "bookprice");
            t_bookstock.Bindings.Add("Text",
myDataSet.Tables["bookstock"], "bookstock");
            //call the method to DataBind
            GetListManager() ;
            isBound=true ;
        }
        statusBar.Text="已经连接，现在你可以操作数据库了。";
    }
    catch(Exception e)
    {
        MessageBox.Show("Error in connecting! "+e.ToString(),
"Error", MessageBoxButtons.IconExclamation);
    }
}

/// <summary>
///     The main entry point for the application.
/// </summary>
public static void Main(string[] args) {
    Application.Run(new DataEdit());
}

/// <summary>
```

```
/// Required method for Designer support - do not modify
/// the contents of this method with an editor
/// </summary>
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
    this.t_bookid = new System.Windows.Forms.TextBox();
    this.previousrec = new System.Windows.Forms.Button();
    this.l_bookauthor = new System.Windows.Forms.Label();
    this.delete = new System.Windows.Forms.Button();
    this.t_booktitle = new System.Windows.Forms.TextBox();
    this.t_bookauthor = new System.Windows.Forms.TextBox();
    this.t_bookprice = new System.Windows.Forms.TextBox();
    this.l_bookprice = new System.Windows.Forms.Label();
    this.t_bookstock = new System.Windows.Forms.TextBox();
    this.l_bookstock = new System.Windows.Forms.Label();
    this.helpme = new System.Windows.Forms.Button();
    this.statusBar = new System.Windows.Forms.StatusBar();
    this.l_booktitle = new System.Windows.Forms.Label();
    this.update = new System.Windows.Forms.Button();
    this.nextrec = new System.Windows.Forms.Button();
    this.lastrec = new System.Windows.Forms.Button();
    this.firstrec = new System.Windows.Forms.Button();
    this.labell = new System.Windows.Forms.Label();
    this.l_bookid = new System.Windows.Forms.Label();

    t_bookid.Location = new System.Drawing.Point(184, 56);
    t_bookid.ReadOnly = true;
    t_bookid.TabIndex = 27;
    t_bookid.Size = new System.Drawing.Size(80, 20);

    previousrec.Location = new System.Drawing.Point(96, 312);
    previousrec.ForeColor = System.Drawing.Color.White;
    previousrec.Size = new System.Drawing.Size(40, 24);
    previousrec.TabIndex = 5;
    previousrec.Font = new System.Drawing.Font("Microsoft Sans Serif",
8f, System.Drawing.FontStyle.Bold);
    previousrec.Text = "上一个";
    previousrec.Click += new System.EventHandler(GoPrevious);

    l_bookauthor.Location = new System.Drawing.Point(24, 160);
    l_bookauthor.Text = "Author.";
```

```
l_bookauthor.Size = new System.Drawing.Size(112, 20);
l_bookauthor.Font = new System.Drawing.Font("Verdana", 10f,
System.Drawing.FontStyle.Bold);
l_bookauthor.TabIndex = 22;
l_bookauthor.BackColor = System.Drawing.Color.DarkOrange;
l_bookauthor.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;

delete.Location = new System.Drawing.Point(184, 352);
delete.ForeColor = System.Drawing.Color.White;
delete.Size = new System.Drawing.Size(80, 24);
delete.TabIndex = 9;
delete.Font = new System.Drawing.Font("Microsoft Sans Serif", 8f,
System.Drawing.FontStyle.Bold);
delete.Text = "删除";
delete.Click += new System.EventHandler(GoDelete);

t_booktitle.Location = new System.Drawing.Point(184, 108);
t_booktitle.TabIndex = 0;
t_booktitle.Size = new System.Drawing.Size(176, 20);

t_bookauthor.Location = new System.Drawing.Point(184, 160);
t_bookauthor.TabIndex = 1;
t_bookauthor.Size = new System.Drawing.Size(128, 20);

t_bookprice.Location = new System.Drawing.Point(184, 212);
t_bookprice.TabIndex = 2;
t_bookprice.Size = new System.Drawing.Size(80, 20);

l_bookprice.Location = new System.Drawing.Point(24, 212);
l_bookprice.Text = "Price.";
l_bookprice.Size = new System.Drawing.Size(112, 20);
l_bookprice.Font = new System.Drawing.Font("Verdana", 10f,
System.Drawing.FontStyle.Bold);
l_bookprice.TabIndex = 23;
l_bookprice.BackColor = System.Drawing.Color.DarkOrange;
l_bookprice.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;

t_bookstock.Location = new System.Drawing.Point(184, 264);
t_bookstock.TabIndex = 3;
t_bookstock.Size = new System.Drawing.Size(80, 20);
```

```
l_bookstock.Location = new System.Drawing.Point(24, 264);
l_bookstock.Text = "Stock.";
l_bookstock.Size = new System.Drawing.Size(112, 20);
l_bookstock.Font = new System.Drawing.Font("Verdana", 10f,
System.Drawing.FontStyle.Bold);
l_bookstock.TabIndex = 24;
l_bookstock.BackColor = System.Drawing.Color.DarkOrange;
l_bookstock.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;

helpme.Location = new System.Drawing.Point(320, 312);
helpme.ForeColor = System.Drawing.Color.White;
helpme.Size = new System.Drawing.Size(64, 24);
helpme.TabIndex = 10;
helpme.Font = new System.Drawing.Font("Microsoft Sans Serif", 8f,
System.Drawing.FontStyle.Bold);
helpme.Text = "帮助";
helpme.Click += new System.EventHandler(GoHelp);

statusBar.BackColor = System.Drawing.SystemColors.Control;
statusBar.Location = new System.Drawing.Point(0, 401);
statusBar.Size = new System.Drawing.Size(394, 24);
statusBar.TabIndex = 24;
statusBar.Text = "Ready";

l_booktitle.Location = new System.Drawing.Point(24, 108);
l_booktitle.Text = "Title.";
l_booktitle.Size = new System.Drawing.Size(112, 20);
l_booktitle.Font = new System.Drawing.Font("Verdana", 10f,
System.Drawing.FontStyle.Bold);
l_booktitle.TabIndex = 21;
l_booktitle.BackColor = System.Drawing.Color.DarkOrange;
l_booktitle.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;

update.Location = new System.Drawing.Point(40, 352);
update.ForeColor = System.Drawing.Color.White;
update.Size = new System.Drawing.Size(80, 24);
update.TabIndex = 8;
update.Font = new System.Drawing.Font("Microsoft Sans Serif", 8f,
System.Drawing.FontStyle.Bold);
update.Text = "更改";
update.Click += new System.EventHandler(GoUpdate);
```

```
nextrec.Location = new System.Drawing.Point(168, 312);
nextrec.ForeColor = System.Drawing.Color.White;
nextrec.Size = new System.Drawing.Size(40, 24);
nextrec.TabIndex = 6;
nextrec.Font = new System.Drawing.Font("Microsoft Sans Serif", 8f,
System.Drawing.FontStyle.Bold);
nextrec.Text = "下一个";
nextrec.Click += new System.EventHandler(GoNext);

lastrec.Location = new System.Drawing.Point(240, 312);
lastrec.ForeColor = System.Drawing.Color.White;
lastrec.Size = new System.Drawing.Size(40, 24);
lastrec.TabIndex = 7;
lastrec.Font = new System.Drawing.Font("Microsoft Sans Serif", 8f,
System.Drawing.FontStyle.Bold);
lastrec.Text = "最后面";
lastrec.Click += new System.EventHandler(GoLast);

firstrec.Location = new System.Drawing.Point(24, 312);
firstrec.ForeColor = System.Drawing.Color.White;
firstrec.Size = new System.Drawing.Size(40, 24);
firstrec.TabIndex = 4;
firstrec.Font = new System.Drawing.Font("Microsoft Sans Serif", 8f,
System.Drawing.FontStyle.Bold);
firstrec.Text = "最前面";
firstrec.Click += new System.EventHandler(GoFirst);

label1.Location = new System.Drawing.Point(49, 8);
label1.Text = "A Book Stock!";
label1.Size = new System.Drawing.Size(296, 24);
label1.ForeColor = System.Drawing.SystemColors.ControlLightLight;
label1.Font = new System.Drawing.Font("Verdana", 14f,
System.Drawing.FontStyle.Bold);
label1.TabIndex = 25;

l_bookid.Location = new System.Drawing.Point(24, 56);
l_bookid.Text = "ID.";
l_bookid.Size = new System.Drawing.Size(112, 20);
l_bookid.Font = new System.Drawing.Font("Verdana", 10f,
System.Drawing.FontStyle.Bold);
l_bookid.TabIndex = 20;
l_bookid.BackColor = System.Drawing.Color.DarkOrange;
```

```
l_bookid.TextAlign = System.Windows.HorizontalAlignment.Center;
this.Text = "Book Stock";
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.BorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
this.BackColor = System.Drawing.SystemColors.Desktop;
this.ClientSize = new System.Drawing.Size(394, 425);

this.Controls.Add(delete);
this.Controls.Add(update);
this.Controls.Add(helpme);
this.Controls.Add(lastrec);
this.Controls.Add(nextrec);
this.Controls.Add(previousrec);
this.Controls.Add(firstrec);
this.Controls.Add(t_bookstock);
this.Controls.Add(t_bookprice);
this.Controls.Add(t_bookauthor);
this.Controls.Add(t_booktitle);
this.Controls.Add(t_bookid);
this.Controls.Add(l_bookstock);
this.Controls.Add(l_bookprice);
this.Controls.Add(l_bookauthor);
this.Controls.Add(l_booktitle);
this.Controls.Add(l_bookid);
this.Controls.Add(label1);
this.Controls.Add(statusBar);

}

///
```



```
///</summary>

protected void GoDelete(object sender, System.EventArgs e)
{
    try{

        string strCon="Provider=Microsoft.Jet.OLEDB.4.0 ;Data
Source=book.mdb" ;
        ADOConnection myConn = new ADOConnection(strCon) ;
        myConn.Open() ;
        string strDele="SELECT * FROM bookstock" ;
        ADODataSetCommand myCommand = new
ADODataSetCommand(strDele,myConn);

        myDataSet.Tables["bookstock"].Rows[myListManager.Position].Delete() ;

        myCommand.Update(myDataSet,"bookstock");

        statusBar.Text="Record Deleted" ;

        myConn.Close() ;

    }
    catch(Exception ed)
    {
        MessageBox.Show("Error in Deleting! "+ed.ToString(),
"Error", MessageBoxButtons.IconExclamation);
    }
}

///<summary>
///    Update Button Clicked
///    <para>
///        This first connects to the database and updates the row
///        Then is calls the GetConneted()<see cref="GetConnected"/>
///        which again reinitilizes the DataSet.
///    </para>
///</summary>

protected void GoUpdate(object sender, System.EventArgs e)
{
```

```
        int i=myListManager.Position ;
        try{
            //connecting to the database
            string strCon="Provider=Microsoft.Jet.OLEDB.4.0 ;Data
Source=book.mdb" ;
            ADOConnection myConn = new ADOConnection(strCon) ;
            myConn.Open() ;

            //update the database
            string strUpdt = "UPDATE bookstock SET booktitle='"
                +t_booktitle.Text+"', bookauthor='"
                +t_bookauthor.Text+"', bookprice='"
                +t_bookprice.Text+"', bookstock='"
                +t_bookstock.Text+" WHERE bookid=
"+t_bookid.Text;

            ADOCommand myCommand = new ADOCommand(strUpdt,myConn);
            myCommand.ExecuteNonQuery();

            statusBar.Text="Record Updated" ;

            myConn.Close() ;

            myDataSet=null ;
            myListManager= null;

            if(isBound)
            {
                t_bookid.Bindings.Remove(0);
                t_booktitle.Bindings.Remove( 0);
                t_bookauthor.Bindings.Remove( 0);
                t_bookprice.Bindings.Remove( 0);
                t_bookstock.Bindings.Remove( 0);
                isBound=false;
            }

            GetConnected();
        }
        catch(Exception ed)
        {
            MessageBox.Show("Error in Updating! "+ed.ToString(),
```

```
"Error", MessageBoxButtons.IconExclamation);

        }
        myListManager.Position=i ;
    }

    ///<summary>
    ///     To navigate the ListManager, increment the Position property.
    ///</summary>
    private void MoveNext()
    {

        if (myListManager.Position == myListManager.Count -1)
            MessageBox.Show("End of records");
        else
            myListManager.Position += 1;
    }

    ///<summary>
    ///     To navigate the ListManager, increment the Position property.
    ///</summary>
    private void MovePrevious(){

        if (myListManager.Position == 0)
            MessageBox.Show("First record");
        else
            myListManager.Position -= 1;
    }

    ///<summary>
    ///     Move to position 0 in the list.
    ///</summary>
    private void MoveFirst(){

        myListManager.Position = 0;
    }

    ///<summary>
    ///     Move to the count -1 position.
    ///</summary>
    private void MoveLast(){

myListManager.Position = myListManager.Count - 1;
```

```
    }

    ///<summary>
    ///     Get Help
    ///</summary>

    protected void GoHelp(object sender, System.EventArgs e)
    {
        MessageBox.Show("查询1.0版", "About ...",
        MessageBox.IconInformation);

    }

    ///<summary>
    ///     Last Record Button Clicked
    ///</summary>

    protected void GoLast(object sender, System.EventArgs e)
    {
        MoveLast();
    }

    ///<summary>
    ///     Next Record Button Clicked
    ///</summary>

    protected void GoNext(object sender, System.EventArgs e)
    {
        MoveNext();
    }

    ///<summary>
    ///     Previous Record Button Clicked
    ///</summary>

    protected void GoPrevious(object sender, System.EventArgs e)
    {
        MovePrevious();
    }

    ///<summary>
    ///     First Record Button Clicked
```

```
///</summary>  
  
protected void GoFirst(object sender, System.EventArgs e)  
{  
    MoveFirst();  
}  
  
}  
}
```

14.3 运行结果

下图是本程序的运行结果，如图 14-2 所示。



图 14-2

由上面我们可以看到，C Sharp 不仅仅可以编写基于 Web 上的程序，在 Window 上也不比其他的语言差。

将来的发展，C Sharp 一定会是 Window 上的程序语言的主流。

第 15 章 附 录

由于微软推出其 .NET 战略还不是很久，加上 .NET 技术还在发展当中，关于 .NET 学习的中文资料还不是太多，即使有一些大多数也只是翻译的英文原作，目前一个比较流行的说法是 2001 年底左右，.NET 技术将成为各网站采用的主流技术。但是在网上，特别是国外的一些网站对 .NET 架构推崇倍至，已经有人采用 .NET 架构建立了自己关于 .NET 学习的网站，上面有一些对 .NET 技术的讨论已经相当的深入，值得我们借鉴。

这里着重介绍的是关于 .NET 架构的权威网站：MSDN ONLINE(<http://msdn.microsoft.com/library/default.asp>)。这个网站相信大家都不陌生，它是所有微软程序开发者的大本营，是微软技术发展的源产地，它的权威性在全球不容置疑。它当中含有一个专题叫作“.net beta Document”(.net 架构 Beta 版的文档)，详细地对微软的 .NET 架构思想、开发工具、开发步骤、注意事项都一一作出了阐述。它目前含有七大部分，分别是：

1. “.NET Framework Developer’s Guide”，net 架构开发者指引，主要阐述 net 架构涉及的新的基本概念如“通用运行环境(CLS)”、“即时编译器(JIT)”、“应用名空间(Application Domain)”等等，如何开发自己的应用，调试优化自己的代码，配置和发布自己的应用。

2. “.NET Framework Reference”，net 架构引用手册，主要是对 net 架构提供的基本服务，各种通用类、结构、成员、方法的详尽说明。它是程序员在开发 .net 应用时的参考手册。

3. “.NET Framework Samples”，net 架构开发实例，主要介绍了应用 .net 技术可以实现的一些功能，在这里只有说明，演示程序在你安装了 .net Framework SDK 的机器上。

4. “.NET Framework Tutorials”，net 架构教程，一个相当简单的入门教程，适合希望快速了解 .NET 架构思想又不想深入开发的一般应用人员。

5. “.NET Framework Tools”，net 架构辅助工具，介绍了 .net Framework SDK 提供的一系列的辅助工具的用法，以帮助开发人员快速的进行程序开发。

6. “.NET Framework Developer Specifications”，net 架构开发者指导，主要对于一些应用设计时需要注意的地方进行了详细的阐述，指导开发者如何对一些应用模型进行开发。

7. “.NET Framework Language Specification”，c#语言指引，C# 语言是微软从 C 和 C++ 发展出来的一种新型的语言，它具有 Visual Basic 的高产量和 C++ 的强大控制能力，是微软力推的下一代语言。

15.1 .NET 学习网站

15Seconds

<http://www.15seconds.com/focus/.NET.htm>

GotDotNet <http://www.gotdotnet.com>

15.2 ASP.NET 的学习网站

ASP.NET <http://www.ASP.NET>
 123aspx.com <http://www.123aspx.com>
 ASP Next Gen <http://www.aspnextgen.com>
 ASPFree.com <http://www.aspfree.com/asp+/Default.aspx>
 ASP Today <http://www.asptoday.com>
 4GuyFromRolla <http://www.4guyfromrolla.com>
 ASPWorkshops.com <http://www.aspworkshops.com>
 411ASP.NET Directory <http://www.411ASP.NET>
 DevelopersDex.com <http://www.developersdex.com>
 ASP 101 <http://www.asp101.com/aspplus>
 ASP Wire <http://www.aspwire.com>
 ASP Watch <http://www.aspwatch.com>
 Dot Net Books <http://www.dotnetbooks.com>
 Dot Net Wire <http://www.dotnetwire.com>
 DevX.com <http://www.devx.com/dotnet>
 IbuySpy <http://www.ibuyspy.com>
 ASPNG.com <http://www.aspng.com/aspng>
 Askasp+ <http://askasp-plus.com/>
 Asp.net developer workshop <http://db101.terrashare.com/>
 硅谷动力 http://www.etechnic.com.cn/columns/asp_1.shtml
 豆腐技术站 <http://www.asp888.net>

15.3 VB.Net 学习网站

Microsoft
<http://msdn.microsoft.com/vstudio/nextgen/default.asp>
 GotDotNet <http://www.gotdotnet.com>
 Earthlink.net
<http://home.earthlink.net/~butlerbob/VBNet/index.htm>

<http://home.earthlink.net/~butlerbob/Port/PortVBNet.htm>

<http://home.earthlink.net/~butlerbob/New/NewVBNet.htm>
 AppleVB.com <http://www.applevb.com/vs7/index.htm>
 中国在线 <http://www.techng.com>
 VB Net <http://www.mvps.org/vbnet/>

15.4 C Sharp 学习网站

The HitMill
<http://www.hitmill.com/programming/dotNET/csharp.html>
C Sharp Help <http://www.csharp-help.com>
C# Station <http://www.csharp-station.com>
Csharpindex.com <http://www.csharp-index.com>
Oreilly.com <http://windows.oreilly.com>
CodeHound.com <http://www.codehound.com/csharp>
Csharp园地 <http://csharp.tongtu.net>
C# Help <http://www.psb.sz.js.cn/images/dir/3/index.html>
Mcpcentral.com <http://www.mcpcentral.com/CSharpTab.asp>
C# Corner <http://www.c-sharpcorner.com/>
The Code Project <http://www.codeproject.com/csharp/>
Andymcm.com <http://www.andymcm.com/csharpfaq.htm>
C# developer <http://developer.al-maqsood.org/>
Findtutorials.com <http://www.findtutorial.com>
Managedworld.com
<http://www.managedworld.com/articles/0002/article.aspx>
C#程序员指南 <http://csharp.myrice.com/>
C#学习 <http://griter.top263.net/display.html>
四海一家 <http://www.friendall.com/dotnet/>