第 1 章

认识 ASP.NET

本章主要内容

1. 什么是.NET。
2. 什么是 ASP.NET。
3. 什么是 Web 窗体。
4. 创建 Web 窗体页。
5. 将控件和静态 HTML 文本添加到页。
6. 将 HTML 元素转换为服务器控件。
7. 认识 HTML 服务器控件。
8. 认识 Web 窗体服务器控件。
9. 事件及事件处理程序。
10. 为控件创建事件处理程序。
11. 生成并运行 Web 窗体页。
12. 熟悉 Web 窗体结构。
13. 熟悉集成开发环境。
14. Web 窗体控件事件模型。
15. Web 窗体代码模型。

1.1 什么是.NET

.NET 是微软公司的新战略,它包含微软公司对未来的核心战略、规划和洞察力。所有微软公司的产品都将围绕这个战略开发。此战略的核心就是.NET Framework,该框架提供了全面支持.NET 的核心技术。.NET Framework 是一种新的计算平台,它简化了在高度分布式 Internet 环境中的应用程序开发。.NET Framework 具有两个主要组件:公共语言运行库和.NET Framework 类库。

.NET 用来解决编程人员面临的许多问题,例如:

① 负责处理在创建大型、可靠的应用程序时的大量艰辛工作。

② 允许程序员统一两种架构——在本地机器上运行的应用程序和通过 Web 访问的

应用程序。

③ 减少了与编程框架相关的传统开销——不再需要高性能编程语言来编写复杂的代码以获取高性能的.NET 程序。

④ 允许不同语言的程序员在一个应用程序中协同工作。

⑤ 开始兼容各种最终用户工具。包括桌面、PDA 和手机。使开发人员能够创建出摆脱设备硬件束缚的,能够在各种操作系统上运行的应用程序。能够轻松实现互联网的连接。

总之,.NET 提供了一种更简单、更快捷、更廉价的方式,来获得高效的程序。

在某些方法上,.NET 很像 Java。实际上,Java 的口号是“一旦编写出来,就能在任何地方运行”,很符合.NET 的原则。不过.NET 并不是 Java 的克隆,微软的方法与之并不相同。

Java 是“一种语言,多个平台”,而.NET 是“多种语言,一个平台……(就现在而言)”,微软想扫除进入.NET 的障碍,为此就要让使用任何语言的人都可对它进行访问。.NET 有两种语言,即 C# 和 Visual Basic.NET,而 Visual Studio.NET 提供了这两种语言。

微软的策略更像是一场军事战役。它运用对 Windows 平台的理解,首先攻占 Windows 平台。当它从 Windows 平台中脱颖而出后,就可以“侵袭”别的平台(例如 Linux),证明.NET 应用程序可以从一个平台移植到另一个平台上。“侵袭并征服了 Linux”后,它又可转向另外的平台,例如 Solaris。

简而言之,.NET 将操作系统平台割裂开来。无论是哪种平台,如 Windows、Linux 等,都可以分成两个层次:程序设计层和执行层。.NET 开发人员是为程序设计层而不是执行层编写软件,将来不管占统治地位的是 Windows 平台还是 Linux 平台,或者是其他出乎意料的系统,就无关紧要了。

可以将.NET Framework 的讨论分解成以下几部分。

① MS 中间语言(MS Intermediate Language):在执行用户编写的所有程序代码前,应将其编译成更抽象、精简的形式。程序员可以使用任何.NET 语言编写代码,包括 Visual Basic、C#、JScript 和其他大约 20 种语言。这些代码都会编译为 MSIL 这种.NET 的通用语言。.NET 在这个层次的操作无需用户干涉,因此本书不讨论它。

② 公共语言运行时(Common Language Runtime,CLR):这是一个复杂的系统,用于在计算机上执行 MSIL 代码。公共语言运行时负责执行与 Windows 和 IIS 交互时所涉及的全部实质性任务。这个层次也是在后台进行的,本书也不涉及。

③ .NET Framework 类库(.NET Framework Class Libraries):实现大量重要功能的代码库。用户可以非常方便地在应用程序中调用库函数,使复杂任务的程序代码更为简洁。本书将介绍这些功能。

④ .NET 语言(.NET Language):.NET 语言是符合一些特殊结构要求(由公共语言规范定义)的编程语言,能够编译成 MSIL。可以用任一种语言进行开发,例如 C# 或 Visual Basic.NET,这没有任何限制,也可以用多种语言来开发程序。本书将用几章的篇幅讨论 Visual Basic 在 ASP.NET 中的应用,并在所有的代码示例中使用 Visual Basic。

⑤ ASP.NET:这个代码模块扩展了 Internet Information Server(IIS),使之能为

Web 页面执行.NET Framework。本书将汲及几乎所有的 ASP.NET 功能。

⑥ Web 服务：虽然 Web 服务不是严格意义上的.NET 的一部分，但.NET 明确地支持它。它们是可以通过 Web 访问的组件，可以是任何主题，例如新闻标题、天气预报、股票走势、病毒预防和操作系统更新等。第 7 章将详细讨论 Web 服务。

在详细介绍上述内容之前，首先介绍一些基本的代码概念与术语。

1.1.1　从用户代码到机器代码

计算机识别以二进制形式表示的任何事物（二进制位是表示指令和数据的 1 与 0 的序列）。因此，人们热衷于用“数字”一词来描述任何事情，即使它们与计算机的关系很小。通常把这种二进制位指令称为机器码。很明显，对大多数人来说，记住用于打印“Good Morning”时的 0 和 1 的代码序列是不可能的，更不用说要记住定义复杂 Web 应用程序的程序代码了。因此，人们发明了编程语言，通过类似英语的单词来编写代码。

一旦用高级语言编写了程序代码，就需要将其转换为机器码，这一转换过程称为编译，编译器软件就可以将人类可读指令编译成机器可读指令。编译过程包括将本机环境信息写入经编译的程序代码这一过程，因此编译后的程序代码可以最高效地利用计算机的所有可用资源。

多年来，有以下两种编译类型，它们的编译过程完全不同。

① 预编译型代码——在编码完成后并在执行前进行编译的代码。对于这种代码，由于编译器会花时间分析全部的代码和要运行它的计算机，因此代码能以很快的速度执行。不过，由于预编译的代码是针对某一计算机的，因此，除非仍在该计算机上使用编译过的代码，否则需要将另一台计算机设成与第一台计算机相同的系统与资源。

② 解释型代码——边执行（在用户请求页面时）边编译的代码。这种代码执行较慢，因为需要为每个请求编译代码，且系统没有机会全面优化所编写的代码。其优点是解释过程可以调整，以适应运行代码的计算机。

所以，开发人员在选择语言方面要做出取舍。既可以选择较慢的解释性代码，以适应运行代码的计算机，也可以选择较快的预编译代码，但不能充分利用计算机的优势。

1.1.2　两种中间语言介绍

.NET 在编译时采用两步来解决这个问题。编写在.NET Framework 上运行的程序时（通常用 Visual Basic.NET 或 C＃编写），需要在使用这些程序之前编译这些可读代码。.NET 编译器的设计方式意味着它并没有把我们带入会引起可移植性问题的二进制码。事实上，.NET 编译器将程序代码编译成称为 MS 中间语言（MSIL）的特殊格式。由于 MSIL 结构不需要像源代码那样易读，因此编译过程包括了一些优化操作。但是，并没有针对某台计算机进行优化。因此，MSIL 具有一般的优化性能，并可以移植到任何.NET 服务器上。

当用户执行 MSIL 代码时（即请求 ASP.NET 页面），将其传给 CLR，CLR 是.NET Framework 的另一个核心。CLR 使用 JIT（Just-In-Time）编译器将代码编译成真正的机器码，并对程序进行最后的且与计算机相匹配的优化，以使程序能在其所在的计算机上以尽可能快的速度运行。

MSIL 和 CLR 组合使用,具有两种编译代码的优点,即预编译码的结构优化和解释码的可移植性。

更为重要的是,MSIL 本身与计算机无关。因此,可以在装有 CLR 的任一计算机上运行。实际上,一旦编写出.NET 程序代码并将其编译,就可以将它复制到装有 CLR 的计算机上,并在该计算机上执行。虽然目前的 CLR 只有与 Windows 兼容的版本,但现在已着手建立基于其他操作系统的版本。在 Web 上搜索与 Mono Project 的信息,就可以找到更多的 CLR 版本。

MSIL 也可以由任一遵循 CLS 的可读语言生成。Visual Basic .NET、C# 和 JScript.NET 是"与.NET 兼容的"三大语言。另外 MSIL 编译器还支持其他 20 多种语言。因此,可以在应用程序内部交替地使用这些兼容语言。一旦将一套文件编译成 MSIL,它们都将统一为一种语言。这种灵活性允许不同的小组在同一个 Web 站点上用不同的语言协同工作。

1.2 什么是 ASP.NET

ASP.NET 是一个基于.NET 的统一的 Web 开发平台,该 Web 开发平台使得 Web 开发人员可以使用任何.NET 编程语言(包括 Visual Basic .NET、C# 和 C++ 托管扩展)开发 Web 应用程序。ASP.NET 还是 .NET Framework 的一部分,所以它提供对该框架所有功能的访问。

1.3 什么是 Web 窗体

Web 窗体是一项 ASP.NET 功能,我们可以使用它为 Web 应用程序创建用户界面。Web 窗体页提供了一种强大而直接的编程模型,该模型使用快速应用程序开发(RAD)技术来生成基于 Web 的复杂用户界面。

在 Web 窗体页中,用户界面编程分为两个不同的部分:可视组件和逻辑。如果用户以前使用过类似于 Visual Basic 和 Visual C++ 的工具,将认同在窗体的可视部分和窗体后与之交互的代码之间存在这样一种划分。

视觉元素被称作 Web 窗体"页"(page)。Web 窗体页由一个包含静态 HTML 或 ASP.NET 服务器控件的文件组成。文件的扩展名为.aspx。

Web 窗体页用作要显示的静态文本和控件的容器。利用 Visual Studio Web 窗体设计器和 ASP.NET 服务器控件,我们可以按照在任何 Visual Studio 应用程序中的方式来设计窗体。

Web 窗体页的逻辑由代码组成,这些代码由用户创建以与窗体进行交互。编程逻辑位于与用户界面文件不同的文件中。该文件称作"代码隐藏"文件,扩展名为".aspx.cs"。在代码隐藏文件中编写的逻辑可以使用 Visual Basic 或 Visual C# 来编写。

1.4　创建第一个 Web 应用程序

实例 1-1　Hello,World

主要知识点：创建一个新的 Web 窗体页，将控件和静态 HTML 文本添加到页，将 HTML 元素转换为服务器控件；认识 HTML 服务器控件，认识 Web 窗体服务器控件，事件及事件处理程序，为控件创建事件处理程序，生成并运行 Web 窗体页，熟悉 Web 窗体结构，熟悉集成开发环境，Web 窗体控件事件模型，Web 窗体代码模型。

创建图 1.1 所示 Web 应用程序，当单击“显示”按钮时，在文本框中显示“Hello,World”，单击“清除”按钮时，将文本框中的内容清空。

图 1.1　第一个 Web 应用程序

1.4.1　创建项目和窗体

1）在“文件”菜单上指向“新建”，然后选择“项目”命令。

2）在“新建项目”对话框中（如图 1.2 所示），执行以下操作：

（1）在“项目类型”窗格中选择“Visual C＃项目”。

（2）在“模板”窗格中选择“ASP.NET Web 应用程序”。

（3）在“位置”文本框中，为应用程序输入完整的 URL（包含 http://、服务器名称和项目名称）。Web 服务器上必须安装 IIS 5（或更高版本）和 .NET Framework。如果计算机上已安装 IIS，可以为服务器指定 http://localhost。

注意：如果已经打开了一个解决方案，可选择“关闭解决方案”，从而使新的 Web 窗体项目成为它自己解决方案的一部分。

3）单击“确定”按钮后，将在指定的 Web 服务器的根处创建新的 Web 窗体项目。

此外，名为 WebForm1.aspx 的新 Web 窗体页将显示在“设计”视图中 Web 窗体设计器上（如图 1.3 所示）。

图 1.2 “新建项目”对话框

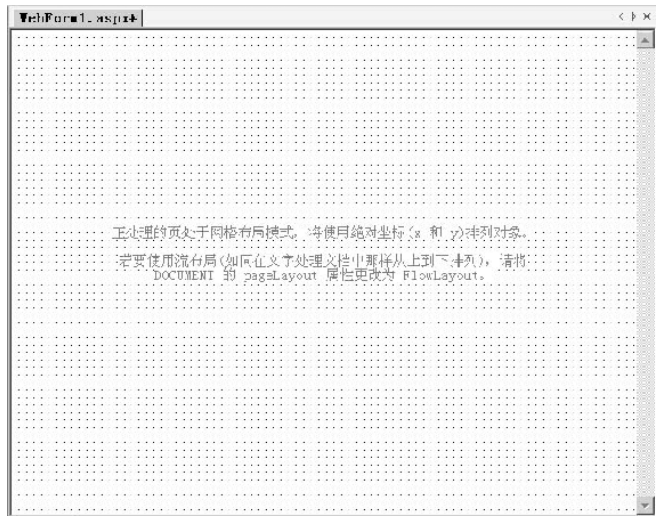

图 1.3 Web 窗体设计器“设计”视图

1.4.2 检查 Web 窗体结构

花一些时间查看 Web 窗体页的结构方式和 Web 窗体设计器的布局方式。通过一个名为 WebForm1.aspx 的文件打开 Web 窗体设计器。Web 窗体页由两个单独的文件组成:

① .aspx 文件包含组成页用户界面的 HTML 文本和控件。

② 一个单独的文件,名为 WebForm1.aspx.cs,包含页的代码,即它是该页的类文件。有时将其称为“代码隐藏”文件。默认情况下,解决方案资源管理器不显示页的类文件。

可用以下方法在解决方案资源管理器中查看页的类文件(如图 1.4 所示):单击解决

方案资源管理器工具栏中的“显示所有文件”按钮，然后展开 WebForm1.aspx 的节点。

在 Web 窗体设计器的底部有两个选项卡（设计和 HTML），这两个选项卡显示正在使用的.aspx 文件的不同视图：

① “设计”视图提供一个所见即所得（WYSIWYG）的视图，可以在其中拖动控件并使用“属性”窗口对它们进行配置。

② HTML 视图显示相同的信息（如图 1.5 所示），但以 HTML 文件的“原始”格式显示。

同在 HTML 文件中一样，Web 窗体设计器支持 HTML 视图中元素的智能感知。

图 1.4　解决方案资源管理器

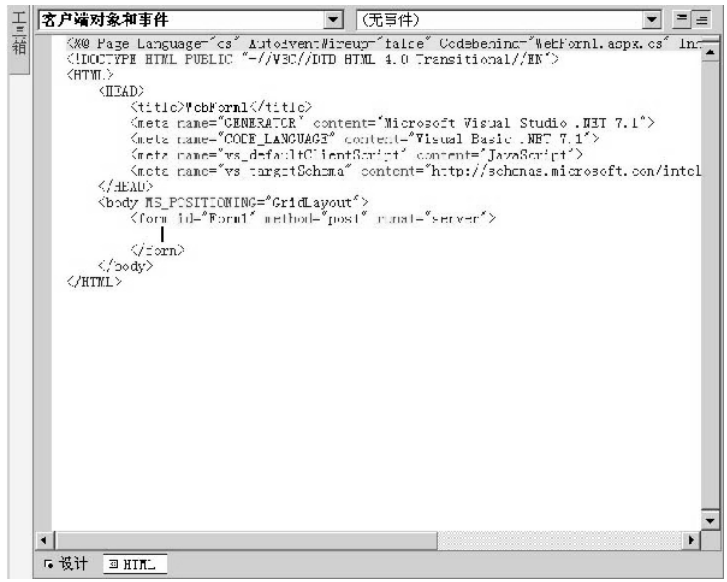

图 1.5　Web 窗体设计器 HTML 视图

可以使用两个视图中的任何一个。在它们之间进行切换时，每个视图都用在另一个视图中所做的更改进行更新。正如我们稍后将看到的，将使用代码编辑器为页的类文件（.aspx.cs）编写代码。

1.4.3　添加控件和文本

现在可以将控件和文本添加到页。首先我们讲网格或流模式。

默认情况下，正在处理的 Web 窗体页处于网格布局模式。在此模式下，可以在页上拖动控件并使用绝对坐标（x 和 y）对它们进行定位。

也可以使用流布局模式，它类似于传统的 HTML 页（项是从上到下放置的）。每种视图都有优点：网格模式便于放置项目。而在流模式中，添加静态文本则更加容易。

对于本实例，我们将使用网格模式，学会如何在此模式下放置静态文本。

1. 将控件添加到 Web 窗体页

Web 窗体控件叫做“服务器控件”,这是因为当页运行时,这些控件将在服务器代码中作为页类的一部分进行实例化。当用户与这些控件交互时(例如当用户单击 Web 窗体按钮控件时),在发送页之后,与控件关联的代码将在服务器上运行。在服务器代码中,可以为服务器控件编写事件处理程序,设置它们的属性等。

并不是 Web 窗体页上的每个元素都是服务器控件。例如,默认情况下,静态 HTML 文本不是服务器控件,不能通过服务器代码对其进行控制。默认情况下,即使标准的 HTML 控件(例如,一个 HTML 提交按钮)也不是服务器控件。在服务器代码中,它们不会作为一级控件显示(就像在任何 HTML 页中一样,HTML 元素在客户端脚本中是可编程的)。

因此,要使用 Web 窗体页上的控件,应该将它们添加为服务器控件。服务器控件有以下两种类型。

① HTML 服务器控件:标记为或转换为可在服务器代码中编程的 HTML 元素,即为 HTML 服务器控件。通常,只有当由于某些原因要在服务器代码中对 HTML 元素进行编程时才将这些 HTML 元素转换为 HTML 服务器控件。

② Web 窗体服务器控件:这些是特定于某些 Web 窗体的控件,这些控件提供的功能比 HTML 服务器控件更多,并且不直接映射到 HTML 元素。

在实例中,我们将添加每种类型的控件各一个。

1) 将 HTML 服务器控件添加到 Web 窗体页

(1) 单击底部的“设计”选项卡切换到“设计”视图。

(2) 从工具箱的 HTML 选项卡中(如图 1.6 所示),将一个 Text Field 元素拖到页上。

切换到 HTML 视图,可看到添加了如下一个标记:

```
<INPUT type = "text">
```

(3) 切换到“设计”视图。

(4) 通过右击该元素并选择“作为服务器控件运行”,将该 HTML 文本元素转换为服务器控件。

控件的左上角出现一个标志符号(),指示该控件是一个服务器控件。转换 HTML 元素使它成为一个 HtmlInputText 服务器控件。

作为该转换的一部分,以下两个属性被添加到 HTML 元素:

① id 属性在代码中标识该控件。对于 HtmlInputText 服务器控件,默认将该属性设置为名称 Text1。

② runat 属性设置为 server。这将元素标记为服务器控件,并使它作为可编程的元素对 Web 窗体服务器代码可见。

2) 将 Web 服务器控件添加到 Web 窗体页

(1) 从工具箱的“Web 窗体”选项卡(如图 1.7 所示)中将一个 Button Web 服务器控件拖到页上。

图 1.6　工具箱 HTML 选项卡

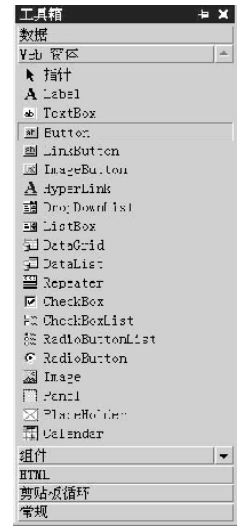

图 1.7　工具箱“Web 窗体”选项卡

注意：只有设计器处于“设计”视图时，“Web 窗体”选项卡才可用。

该步骤在设计器中创建一个 Web 服务器控件元素而不是一个 HTML 按钮。如果切换到“HTML”视图，将看到以下内容：

```
<asp:Button id = "Button1" runat = "server" Text = "Button"></asp:Button>
```

该元素不直接对应于 HTML 元素。而是当页运行时，创建并处理 Button Web 服务器控件的一个实例。在页处理过程中，控件将某些 HTML 元素输出到页（在本例中为一个 HTML<input type=submit>元素）。

（2）在“设计”视图中，选择 Button 控件，在属性窗口中，将 Text 属性设置为“显示”，将 ID 属性设置为“btnShow”（如图 1.8 所示）。

切换到“HTML”视图，将看到以下内容：

```
<asp:Button id = "btnShow" runat = "server" Text = "显示"></asp:Button>
```

（3）切换到“设计”视图，再将一个 Button Web 服务器控件拖到页上，在属性窗口中将其 Text 属性设置为“清除”，将 ID 属性设置为“btnClear”。

2. 将 HTML 文本添加到页

在网格布局模式中，放置控件十分简单。但放置静态 HTML 文本的情况如何呢？因为页上的所有元素都是通过 x 和 y 坐标放置的，所以我们不能像在流布局模式中那样只是在页上想要文本出现的位置输入文本。

具体方法是，添加一个 HTML Label 控件，可以在其中添加静态文本。然后，可以将该标签（实质上是一个<DIV>元素）放置在页上的任何位置。在网格布局模式下添加静态文本的步骤如下：

（1）从工具箱的 HTML 选项卡中，将一个 Label 控件拖到页上。放置该控件并根据要输入的文本调整其大小。

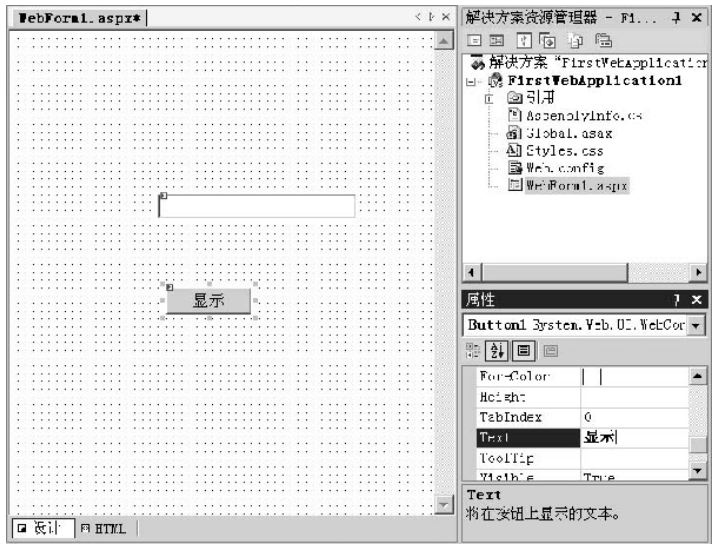

图 1.8 设置控件属性

(2) 单击该标签将其选中,然后再次单击它(缓慢地进行此操作,以确保不是双击该元素)。

该标签进入文本编辑模式,表现为带阴影的边框。

(3) 输入所需的静态文本。例如,输入"第一个 Web 应用程序"(如图 1.9 所示)。

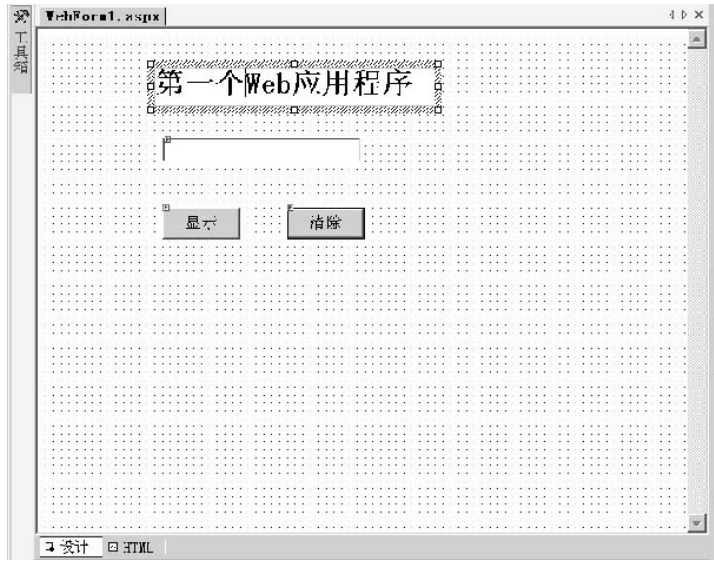

图 1.9 将 HTML 文本添加到页

(4) 选择该文本,然后使用"格式"工具栏上的工具设置文本的块格式、字体、大小等。

(5) 单击标签以外的部分离开文本编辑模式。