第1章　初识ASP.NET AJAX

ASP.NET AJAX技术是一种实现异步（Asynchronous）网络应用的技术，它被整合在ASP.NET 2.0之中，是ASP.NET的一种扩展技术。通过ASP.NET AJAX技术，开发人员或程序员可以将Web服务器控件和客户端脚本结合起来，并在此基础上实现了Web页面的局部更新功能。因此，当浏览器与服务器交互时，ASP.NET AJAX技术可以将浏览器中的一部分内容呈现出来，从而避免了将浏览器整个内容提交到服务器。

1.1　ASP.NET AJAX概述

本小节将介绍ASP.NET AJAX的基本内容，如ASP.NET AJAX特点与原理、ASP.NET AJAX服务器端架构等。

1.1.1　ASP.NET和ASP.NET AJAX

AJAX是Asynchronous JavaScript and XML（异步JavaScript和XML）的缩写。它是一种创建交互式网页应用的网页开发技术。在AJAX中，XMLHttpRequest是其最核心的技术。它为页面中的Javascript脚本提供了一种通讯方式，从而使得页面通过这些脚本能够与服务器发生交互。页面内的Javascript脚本可以在不刷新页面的情况下与服务器发生交互，即页面可以从服务器获取数据，或者向服务器提交数据。AJAX技术和传统Web技术相比，存在以下3个主要区别。

- 能够更新页面中的部分内容，不需要刷新整个页面就能够与服务器通信。
- 页面和服务器直接的通讯可以使用异步操作，从而不需要打断用户的操作，使得页面具有更加快速的响应能力。
- 由于页面与服务器交互时，只需要页面的部分内容，因此减少了页面与服务器的通讯量，提高了应用程序的效率。

微软在ASP.NET框架基础之上，创建了一种被称为“ASP.NET AJAX”的技术，能够实现AJAX功能。ASP.NET AJAX技术最显著的功能就是：当浏览器与服务器交互时，它可以将浏览器中的一部分内容呈现出来（浏览器其他部分不需要与服务器交互），从而避免了将浏览器的整个内容提交到服务器。ASP.NET AJAX技术还提供了处理ECMAScript（Javascript）和动态HTML（DHTML）的脚本库，从而实现了服务器端控件或组件能够调用客户端脚本。和传统的ASP.NET Web应用程序相比，ASP.NET AJAX Web应用程序具有以下优点：

- Web窗体页能够实现局部更新的功能。
- 异步回传，能够将Web窗体页处理的逻辑与用户的操作进行异步处理。
- 正是因为Web窗体页局部更新的功能，所以减少了Web窗体页与服务器的通讯压力，提供了应用程序的性能。

ASP.NET AJAX包括服务器端部分和客户端部分。其中，ASP.NET AJAX服务器端部分提供了5个服务器端控件。通过这些控件，开发人员或程序员可以轻松实现异步网页和一个无刷新的Web环境。ASP.NET AJAX服务器端部分提供的5个服务器端控件说明如下：

- ScriptManager控件：管理页面的脚本。
- ScriptManagerProxy控件：管理页面的脚本。

- UpdatePanel控件：和ScriptManager控件共同提供了一个无刷新的Web环境。
- Timer控件：被称为定时器，它能够定时触发用户自定义的操作。
- UpdateProgress控件：显示整个或部分页面更新的过程。

1.1.2 ASP.NET AJAX服务器端架构

ASP.NET AJAX服务器端架构如图1.1所示。它基于ASP.NET框架之上，主要包括4个部分：ASP.NET AJAX服务器控件、ASP.NET AJAX服务器控件扩展、ASP.NET AJAX客户端脚本和ASP.NET AJAX Web服务。各个部分的具体说明如下：

- ASP.NET AJAX服务器控件：主要包括ASP.NET AJAX的基础控件，如ScriptManager、UpdatePanel、UpdateProgress、Timer、ScriptManagerProxy等控件。另外，ASP.NET AJAX Control Toolkit中也提供了部分控件，如NoBot、ReorderList等控件。
- ASP.NET AJAX服务器控件扩展：主要包括ASP.NET AJAX Control Toolkit中提供的扩展控件，如AutoCompleteExtener、CascadingDropDownExtener等控件。
- ASP.NET AJAX客户端脚本：包括处理ASP.NET AJAX客户端脚本的相关组件。例如，ASP.NET AJAX能够把服务器端的功能包装为客户端脚本的功能，从而使得服务器端的功能转换为客户端的功能。
- ASP.NET AJAX Web服务：包括处理ASP.NET AJAX Web的相关组件，如一种被称为ASP.NET AJAX Web服务桥（Bridge）机制。

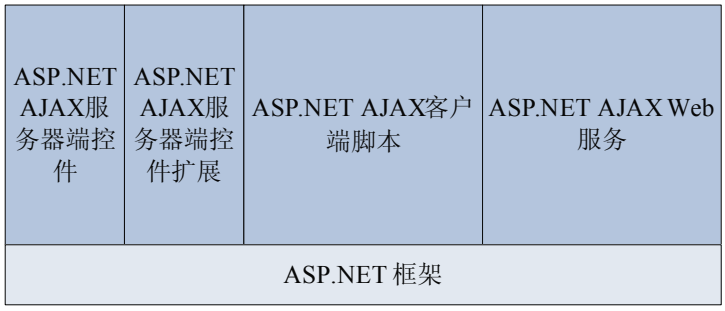

图1.1 ASP.NET AJAX服务器端架构

1.1.3 ASP.NET AJAX客户端架构

ASP.NET AJAX架构包括服务器端架构和客户端架构。由于本书重点介绍ASP.NET AJAX服务器端架构，因此下面仅仅对ASP.NET AJAX客户端架构做简单介绍。ASP.NET AJAX客户端架构的简单结构如图1.2所示。各个部分的具体说明如下：

- 客户端组件或控件：主要包括处理和封装客户端控件的类或组件，使得开发人员能够更加轻松和方便地使用这些客户端控件。
- ASP.NET AJAX基础框架/网络基础：主要包括ASP.NET AJAX中与客户端相关的基础功能，如实现用户界面（UI）的基础类库、网络访问层、框架基础类库、浏览器兼容功能等。

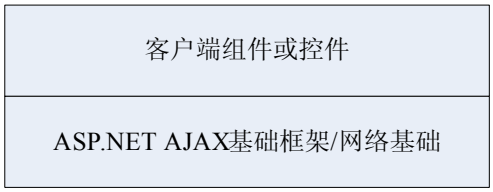

图1.2　ASP.NET AJAX客户端架构

1.2　搭建ASP.NET AJAX开发环境

ASP.NET AJAX是免费的，它可以从微软ASP.NET的官方网站（http://ajax.asp.net）下载。下载ASP.NET AJAX的具体网址为http://www.asp.net/ajax/downloads/，如图1.3所示。单击【Download ASP.NET Extensions v1.0】按钮，就可以下载ASP.NET Extensions v1.0。在此，笔者下载了ASP.NET Extensions v1.0。

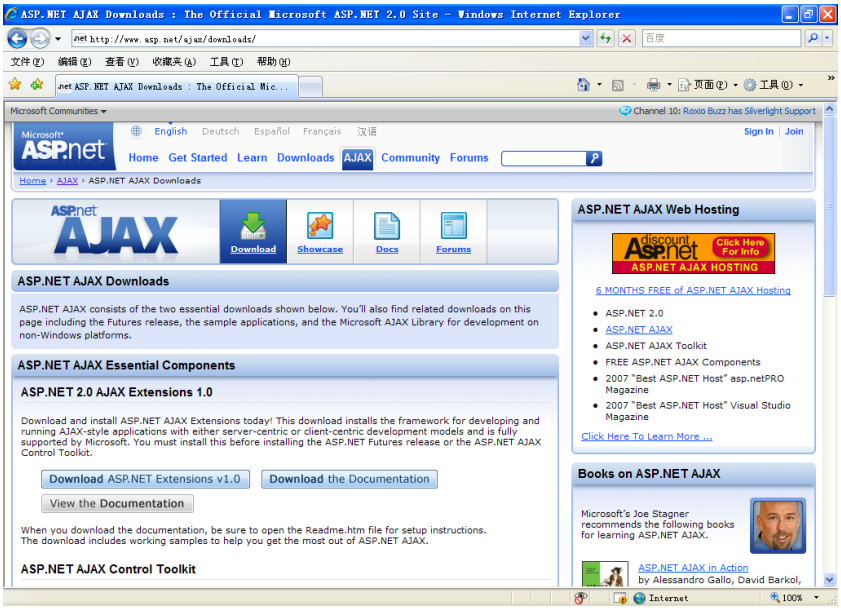

图1.3　下载ASP.NET AJAX的网页

在安装ASP.NET AJAX之前，必须事先检查系统是否已经满足安装ASP.NET AJAX的必备条件。必备条件的具体说明如下：

- 操作系统可以为Windows家族（如Windows Server 2003、Windows XP Home、Windows XP Professional、Windows Vista等）。
- 安装.NET Framework2.0或更高版本。
- IE5.01或更高版本。

如果机器已经具备了上述全部条件，就可以开始安装ASP.NET AJAX。下面介绍在笔者机器（配置为Windows 2003 Server和.NET Framework 3.0）上安装ASP.NET AJAX v1.0的具体步骤如下。

（1）双击ASP.NET AJAX v1.0的安装程序，弹出【Microsoft ASP.NET 2.0 AJAX

Extensions 1.0 Setup】对话框，如图1.4所示。

（2）单击【Next】按钮，弹出【Microsoft ASP.NET 2.0 AJAX Extensions 1.0 Setup】对话框，并选择【I accept the terms in the License Agreement】复选框，如图1.5所示。该对话框显示用户使用ASP.NET AJAX所遵循的协议。

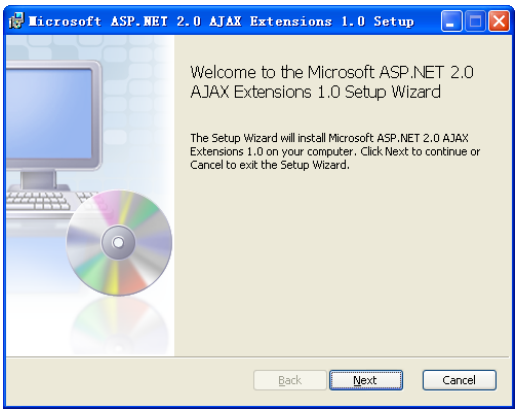

图1.4 【Microsoft ASP.NET 2.0 AJAX …】对话框

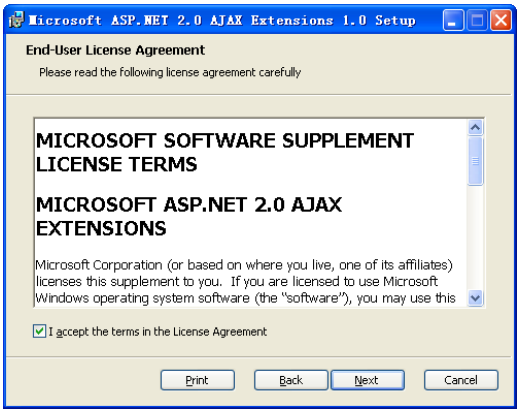

图1.5 【End-User License Agreement】对话框

（3）单击【Next】按钮，弹出【Microsoft ASP.NET 2.0 AJAX Extensions 1.0 Setup】对话框，如图1.6所示。该对话框显示安装ASP.NET AJAX之前的准备信息。

（4）单击【Install】按钮，将开始安装ASP.NET AJAX v1.0。安装完成之后，弹出【Microsoft ASP.NET 2.0 AJAX Extensions 1.0 Setup】对话框，如图1.7所示。单击【Finish】按钮，可以完成安装。

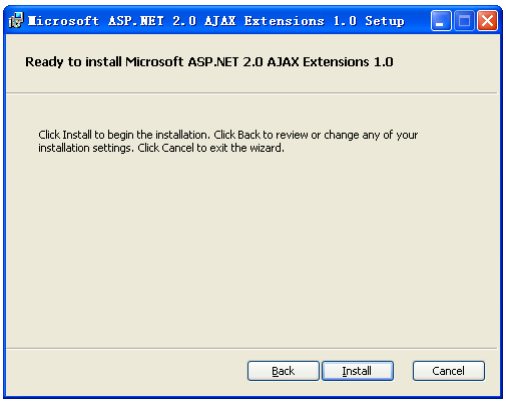

图1.6 【Ready to install Microsoft…】对话框

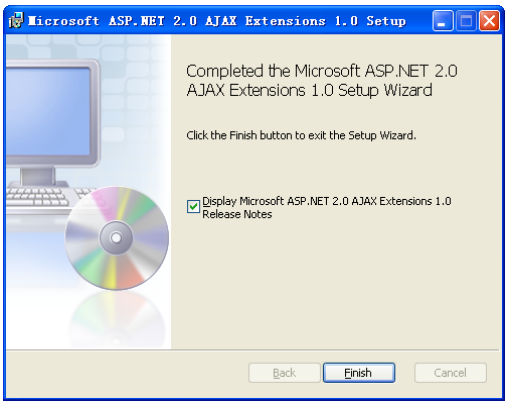

图1.7 【Completed the Microsoft…】对话框

安装ASP.NET AJAX v1.0成功之后，Visual Studio2005集成开发环境的【工具箱】面板中将出现【AJAX Extensions】选项卡。该选项卡显示了ASP.NET AJAX包含的Web服务器控件，如图1.8所示。

图1.8　【AJAX Extensions】选项卡

ASP.NET AJAX默认安装在“C:\Program Files\Microsoft ASP.NET”位置。依次打开该位置下的“ASP.NET 2.0 AJAX Extensions\v1.0.61025”目录，如图1.9所示。用户可以在【v1.0.61025】目录中查看ASP.NET AJAX包含的程序集，如System.Web.Extensions.dll、System.Web.Extensions.Design.dll等。其中，System.Web.Extensions.dll程序集是ASP.NET AJAX中最重要的程序集。

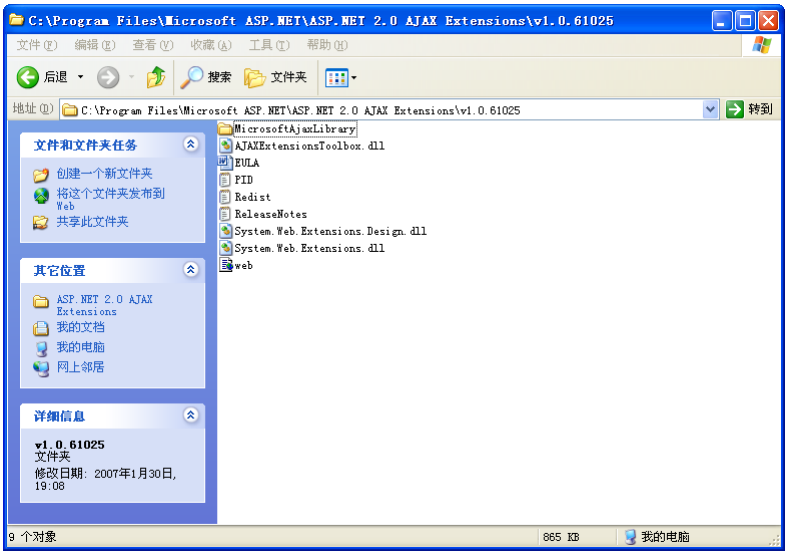

图1.9　查看ASP.NET AJAX包含的程序集

1.3　第一个ASP.NET AJAX Web应用程序AjaxStart

本小节介绍创建第一个ASP.NET AJAX Web应用程序，名称为AjaxStart。在Visual Studio 2005集成开发环境的【起始页】中，单击【创建】|【网站】链接，弹出【新建网站】对话框，选中【ASP.NET AJAX-Enabled Web Site】图标，同时设置保存网站的位置为“D:\ASP.NET AJAX\AjaxStart”，如图1.10所示。

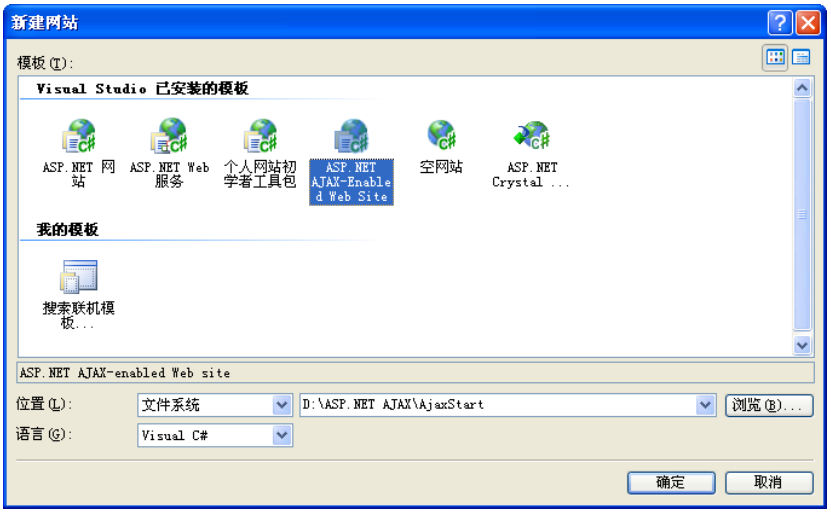

图1.10　【新建网站】对话框

单击【确定】按钮，就可以创建ASP.NET AJAX Web应用程序AjaxStart。在【解决方案资源管理器】面板中查看AjaxStart应用程序，如图1.11所示。该应用程序包含1个Web窗体页Default.aspx和1个配置文件Web.config。

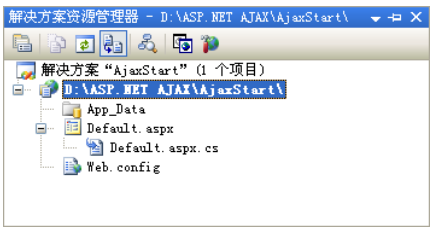

图1.11　查看应用程序AjaxStart

打开AjaxStart应用程序中的默认页面Default.aspx，并查看其HTML源代码。可以发现该页面已经包含一个ScriptManager控件。该控件是一个ASP.NET AJAX服务器端控件，专门用来管理页面上的脚本。Default.aspx页面的HTML代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
    Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <!-- ASP.NET AJAX服务器端控件ScriptManager -->
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <div>
        </div>
    </form>
```

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

1.4　配置ASP.NET AJAX Web应用程序

本小节将介绍配置ASP.NET AJAX Web应用程序的具体方法，如配置ASP.NET AJAX Web应用程序的程序集和配置文件Web.Config。

1.4.1　配置程序集

默认情况下，ASP.NET AJAX Web应用程序都将默认引用了System.Web.Extensions.dll程序集。另外，用户还可以手工配置ASP.NET AJAX Web应用程序的程序集。下面介绍配置AjaxStart应用程序的ASP.NET AJAX Web程序集的具体步骤。

（1）在AjaxStart应用程序【解决方案资源管理器】面板中，右击【D:\ASP.NET AJAX\AjaxStart】节点，弹出右键菜单，如图1.12所示。

（2）单击【添加引用】命令，弹出【添加引用】对话框。单击【浏览】选项卡，并找到ASP.NET AJAX的System.Web.Extensions.dll程序集，如图1.13所示。

图1.12　添加引用的操作

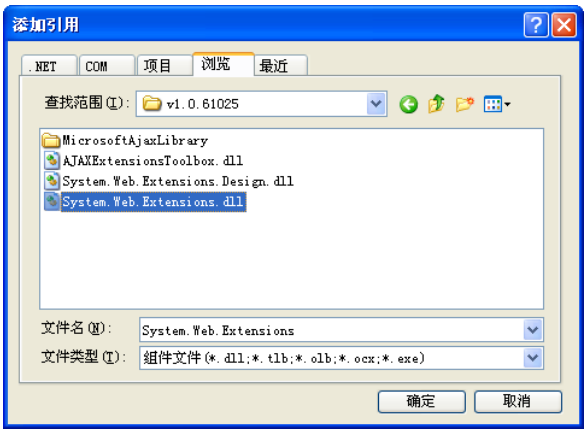

图1.13　【添加引用】对话框

（3）单击【确定】按钮，添加System.Web.Extensions.dll程序集。

注意：AjaxStart 应用程序并不会把 System.Web.Extensions.dll 程序集显式添加到其 Bin 目录下。

1.4.2　配置Web.Config文件

和普通的ASP.NET Web应用程序相比，ASP.NET AJAX Web应用程序的Web.Config配置文件是不相同的。下面以AjaxStart Web应用程序为例，介绍ASP.NET AJAX Web应用程序的Web.Config配置文件。

1. 配置<configSections>元素

ASP.NET AJAX Web网络应用程序的Web.Config配置文件的<configSections>元素新增加了<sectionGroup>子元素，并设置该子元素的name属性的值为“system.web.extensions”。程序代码如下：

```
<?xml version="1.0"?>
<configuration>
   <configSections>
      <sectionGroup name="system.web.extensions" type=
         "System.Web.Configuration.SystemWebExtensionsSectionGroup,
         System.Web.Extensions, Version=1.0.61025.0,
         Culture=neutral, PublicKeyToken=31bf3856ad364e35">
         ……
```

上述<sectionGroup>元素添加了与脚本相关的子元素<sectionGroup name="scripting">。如果希望ASP.NET Web应用程序在运行时能够对客户端脚本进行压缩或缓存，则需要在<sectionGroup name="scripting">元素下添加一个<section name="scriptResourceHandler">元素，具体程序代码如下：

```
            <sectionGroup name="scripting"
               type="System.Web.Configuration.ScriptingSectionGroup,
               System.Web.Extensions, Version=1.0.61025.0,
               Culture=neutral,PublicKeyToken=31bf3856ad364e35" >
               <section name="scriptResourceHandler" type=
                   "System.Web.Configuration.
                   ScriptingScriptResourceHandlerSection,
                   System.Web.Extensions, Version=1.0.61025.0,
                   Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                   requirePermission="false"
                   allowDefinition="MachineToApplication"/>
```

如果ASP.NET Web应用程序需要使用自定义的JOSN转换方式，则需要在<sectionGroup name="webServices">元素下添加一个<section name="jsonSerialization">元素，具体程序代码如下：

```
               <sectionGroup name="webServices" type="
                   System.Web.Configuration.
                      ScriptingWebServicesSectionGroup,
                   System.Web.Extensions, Version=1.0.61025.0,
                   Culture=neutral, PublicKeyToken=31bf3856ad364e35">
                   <section name="jsonSerialization" type="
                      System.Web.Configuration.
                         ScriptingJsonSerializationSection,
                      System.Web.Extensions, Version=1.0.61025.0,
                      Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                      requirePermission="false"
                      allowDefinition="Everywhere"/>
```

如果ASP.NET Web应用程序需要使用自定义个性化配置服务，则需要在<sectionGroup name="webServices">元素下添加一个<section name="profileService">元素，具体程序代码如下：

```
                   <section name="profileService" type="
                      System.Web.Configuration.
                         ScriptingProfileServiceSection,
                      System.Web.Extensions, Version=1.0.61025.0,
                      Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                      requirePermission="false"
```

```
                    allowDefinition="MachineToApplication" />
```

如果ASP.NET Web应用程序需要使用自定义的验证服务，则需要在<sectionGroup name="webServices">元素下添加一个<section name="authenticationService">元素，具体程序代码如下：

```
                    <section name="authenticationService" type="
                        System.Web.Configuration.
                            ScriptingAuthenticationServiceSection,
                        System.Web.Extensions, Version=1.0.61025.0,
                        Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                        requirePermission="false"
                        allowDefinition="MachineToApplication" />
                </sectionGroup>
            </sectionGroup>
        </sectionGroup>
    </configSections>
    ……
```

2. 配置<system.web>元素

ASP.NET AJAX Web网络应用程序的Web.Config配置文件在<system.web>元素下新增加了4个与ASP.NET AJAX相关的配置。

- 在<pages>元素的<controls>子元素下增加了与ASP.NET AJAX服务器控件相关的程序集。程序代码如下：

```
    <system.web>
        <pages>
            <controls>
                <add tagPrefix="asp" namespace="System.Web.UI"
                    assembly="System.Web.Extensions, Version=1.0.61025.0,
                    Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
            </controls>
        </pages>
```

- 在<compilation debug="false">元素的<assemblies>子元素下增加了与编译ASP.NET AJAX程序的相关程序集。程序代码如下：

```
        <compilation debug="false">
            <assemblies>
                <add assembly="System.Web.Extensions, Version=1.0.61025.0,
                    Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
            </assemblies>
        </compilation>
```

- 在<httpHandlers>元素下增加了Web文件和与处理脚本相关的文件，如*.asmx、*_AppService.axd、ScriptResource.axd等。程序代码如下：

```
        <httpHandlers>
            <remove verb="*" path="*.asmx"/>
            <add verb="*" path="*.asmx" validate="false"
                type="System.Web.Script.Services.ScriptHandlerFactory,
                System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
                PublicKeyToken=31bf3856ad364e35"/>
            <add verb="*" path="*_AppService.axd" validate="false"
```

```
            type="System.Web.Script.Services.ScriptHandlerFactory,
            System.Web.Extensions, Version=1.0.61025.0,Culture=neutral,
            PublicKeyToken=31bf3856ad364e35"/>
        <add verb="GET,HEAD" path="ScriptResource.axd"
            type="System.Web.Handlers.ScriptResourceHandler,
            System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
            PublicKeyToken=31bf3856ad364e35" validate="false"/>
    </httpHandlers>
```

- 在<httpModules>元素下增加了处理脚本模块（ScriptModule）相关的内容。程序代码如下：

```
    <httpModules>
        <add name="ScriptModule" type="System.Web.Handlers.ScriptModule,
            System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
            PublicKeyToken=31bf3856ad364e35"/>
    </httpModules>
</system.web>
```

3. 配置<system.webServer>元素

ASP.NET AJAX Web网络应用程序的Web.Config配置文件在<system.webServer>元素下新增加了3个与ASP.NET AJAX相关的配置。

- 设置<validation>元素的validateIntegratedModeConfiguration属性的值为false。程序代码如下：

```
<system.webServer>
    <validation validateIntegratedModeConfiguration="false"/>
```

- 在<modules>元素下增加了处理脚本模块（ScriptModule）相关的内容。程序代码如下：

```
    <modules>
        <add name="ScriptModule" preCondition="integratedMode"
            type="System.Web.Handlers.ScriptModule, System.Web.Extensions,
            Version=1.0.61025.0, Culture=neutral,
            PublicKeyToken=31bf3856ad364e35"/>
    </modules>
```

- 在<handlers>元素下增加了处理脚本相关的工厂（Factory）内容，如ScriptHandlerFactory、ScriptHandlerFactoryAppServices、ScriptResource等。程序代码如下：

```
    <handlers>
        <remove name="WebServiceHandlerFactory-Integrated" />
        <add name="ScriptHandlerFactory" verb="*" path="*.asmx"
            preCondition="integratedMode"
            type="System.Web.Script.Services.ScriptHandlerFactory,
            System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
            PublicKeyToken=31bf3856ad364e35"/>
        <add name="ScriptHandlerFactoryAppServices" verb="*"
            path="*_AppService.axd" preCondition="integratedMode"
            type="System.Web.Script.Services.ScriptHandlerFactory,
            System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
            PublicKeyToken=31bf3856ad364e35"/>
```

```
            <add name="ScriptResource" preCondition="integratedMode"
                verb="GET,HEAD" path="ScriptResource.axd"
                type="System.Web.Handlers.ScriptResourceHandler,
                System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
                PublicKeyToken=31bf3856ad364e35" />
        </handlers>
    </system.webServer></configuration>
```

1.5　脚本管理控件ScriptManager

本小节将介绍与ScriptManager控件相关的内容，如ScriptManager控件的属性、方法、引入脚本资源、引入Web服务资源、处理AJAX中的异常等。

1.5.1　ScriptManager控件概述

ScriptManager控件又称为脚本管理控件。它能够管理Web窗体页（或内容页、母版页等）上的脚本，以及基于ASP.NET AJAX的服务器端和客户端控件。它和UpdatePanel控件（1.6小节将详细介绍）能够共同实现无刷新的Web环境。ScriptManager控件不但能够动态创建与Web服务器相关的脚本，而且这些脚本还能够支持Web窗体页的局部更新功能。通过使用ScriptManager和UpdatePanel控件，Web窗体页把其需要更新的内容提交到服务器，而不需要更新的内容则不提交到服务器，从而实现了Web窗体页的局部更新功能。

ScriptManager控件能够放置在Web窗体页、内容页或母版页等页面上的任何位置。但是它必须放置在依赖于ASP.NET AJAX的控件或脚本的前面，否则页面可能发生脚本错误。因此，ScriptManager控件一般放置在Web窗体页的Form元素之后。

注意：如果一个 Web 窗体页上使用了 ScriptManager 控件，那么该页有且只能放置一个 ScriptManager 控件。

下述程序代码实例显示了一个ASP.NET AJAX Web窗体页的HTML设计代码。可以看到，ScriptManager控件（ID属性的值为ScriptManager1）放置在Web窗体页的Form元素之后。

```
    <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
        Inherits="_Default" %>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title>Untitled Page</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <!-- ASP.NET AJAX服务器端控件ScriptManager -->
            <asp:ScriptManager ID="ScriptManager1" runat="server" />
            <!-- 页面中的其他控件或元素等，如ASP.NET服务器端控件、ASP.NET AJAX服务器端
控件等 -->
        </form>
    </body>
    </html>
```

1.5.2 ScriptManager控件的属性

ScriptManager控件提供了多个与Web窗体页脚本相关的属性。这些属性包括指定是否支持局部更新的EnablePartialRendering、页面发生错误时显示错误信息的AsyncPostBackErrorMessage、动态引入的脚本集合的Scripts、动态引入的服务集合的Services等属性。它们的具体说明如表1-1所示。

表1-1 ScriptManager控件常用的属性及其说明

名称	说明
EnablePartialRendering	是否支持局部更新，默认值为True。
IsInAsyncPostBack	是否为异步处理模式。
ScriptMode	设置发送脚本的模式。
AllowCustomErrorsRedirect	是否使用配置文件Web.Config中的customErrors元素指定的网页来处理异常。
AsyncPostBackErrorMessage	当页面发生错误时，显示的错误信息。
ScriptLoadTimeout	载入脚本的超时限制。单位为秒。
ScriptPath	脚本资源的根目录。
Scripts	动态引入的脚本集合。
Services	动态引入的Web服务集合。
AuthenticationService	与当前ScriptManager实例相关的AuthenticationServiceManager对象。
ProfileService	与当前ScriptManager实例相关的ProfileServiceManager对象。
AsyncPostBackSourceElementID	引起异步回发的控件的ID属性的值。
AsyncPostBackTimeout	异步回传时超时限制。默认值为90，单位为秒。
EnableScriptGlobalization	是否启用脚本的国际化功能。
EnableScriptLocalization	是否启用脚本的本地化功能。
IsDebuggingEnabled	是否启用脚本的debug模式。

1.5.3 ScriptManager控件的方法

ScriptManager控件提供了8个向Web窗体页中注册脚本的静态方法，如表1-2所示。这些方法不但可以注册脚本代码（如javascript），还可以注册脚本使用的资源和文件等。

表1-2 ScriptManager控件的静态方法及其说明

名称	说明
RegisterArrayDeclaration()	注册数组。
RegisterClientScriptBlock()	注册脚本块。
RegisterClientScriptInclude()	注册脚本文件。
RegisterClientScriptResource()	注册脚本资源。
RegisterExpandoAttribute()	注册属性，即一个名称/键对。
RegisterHiddenField()	注册隐藏的域。
RegisterOnSubmitStatement()	注册在提交页面时执行的代码。
RegisterStartupScript()	注册在页面开始运行时执行的代码。
GetCurrent()	获取Web窗体页的ScriptManager实例。

1.5.4 引入脚本资源

ScriptManager控件可以以声明的方式引入脚本资源，如Javascript文件（文件后缀为.js）等。ScriptManager控件以声明方式引入脚本资源的语法类似如下：

```
<asp:ScriptManager ID="sm" runat="server">
   <Scripts>
     <asp:ScriptReference Path="AjaxScript.js" ScriptMode="Auto" />
   </Scripts>
</asp:ScriptManager>
```

ScriptManager控件引入的脚本资源放置在其<Scripts>元素下，每一个资源需要使用一个<asp:ScriptReference>元素。该元素与ScriptReference类相对应，它包含7个常用属性，具体说明如表1-3所示。

表1-3　ScriptReference类常用的属性及其说明

属性	说明
Assembly	引入脚本资源被包含的程序集的名称。
IgnoreScriptPath	是否在载入脚本资源时包含脚本的路径。
Name	引入脚本资源的文件名称。
NotifyScriptLoaded	是否在加载脚本资源完成之后发出一个通知。
Path	引入脚本资源的路径，一般为相对路径。
ResourceUICultures	脚本资源包含的本地化信息。
ScriptMode	加载脚本资源的模式，可以为debug或者release模式。默认值为Auto。

在下述代码实例中，AjaxScript.js脚本文件定义了一个脚本函数StringJoin(tbLeftValue, tbRightValue, tbResultValue)。该函数把tbLeftValue和tbRightValue控件的值（为字符串类型）拼接起来，然后设置为tbResultValue控件的值。AjaxScript.js脚本文件作为脚本资源，将被Scripts.aspx页面引用。

```
//AjaxScript.js脚本文件
function StringJoin(tbLeftValue,tbRightValue,tbResultValue)
{   ///拼接两个字符串
   tbResultValue.value = tbLeftValue.value + tbRightValue.value;
}
```

在下述代码实例中，Scripts.aspx页面在ScriptManager控件中引入了AjaxScript.js脚本文件，并设置引入脚本的模式为Auto（即ScriptMode属性的值）。Scripts.aspx页面还声明了3个输入框类型的HTML控件：tbLeftValue、tbRightValue和tbResultValue。其中，tbLeftValue和tbRightValue控件都是用来输入被拼接的两个值，tbResultValue控件显示tbLeftValue和tbRightValue控件的值拼接起来的结果。

另外，Scripts.aspx页面还声明了一个HTML按钮：btnComputer。该按钮调用AjaxScript.js脚本文件中的StringJoin(tbLeftValue, tbRightValue, tbResultValue)函数执行字符串拼接操作，并把拼接的结果显示在tbResultValue控件中。

```
<!-- AjaxStart/Scripts.aspx页面 -->
<%@ Page Language="C#" %>
<head runat="server"><title>引入脚本资源</title></head>
<asp:ScriptManager ID="sm" runat="server">
   <Scripts>
     <asp:ScriptReference Path="AjaxScript.js" ScriptMode="Auto" />
   </Scripts>
</asp:ScriptManager>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
   左值：<input id="tbLeftValue" type="text" /><br />
   右值：<input id="tbRightValue" type="text" /><br />
   结果：<input id="tbResultValue" type="text" />
```

```
        <input id="btnComputer" type="button" value="拼接两个字符串"
            onclick="StringJoin(tbLeftValue,tbRightValue,tbResultValue)"  />
    </ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，分别在【左值】和【右值】输入框中输入“123”和“456”，然后单击【拼接两个字符串】按钮拼接上述两个值，结果如图1.14所示。

图1.14　演示ScriptManager控件引入脚本资源的功能

1.5.5　引入Web服务资源

ScriptManager控件可以以声明的方式引入Web服务资源，如Web服务文件（文件后缀为.asmx）等。ScriptManager控件以声明方式引入Web服务资源的语法类似如下：

```
    <asp:ScriptManager ID="sm" runat="server">
        <Services>
            <asp:ServiceReference Path="AjaxService.asmx" />
        </Services>
    </asp:ScriptManager>
```

ScriptManager控件引入的Web服务资源放置在其<Services>元素下，每一个资源需要使用一个<asp:ServiceReference>元素。该元素与ServiceReference类相对应，它包含2个常用属性，具体说明如表1-4所示。

表1-4　ScriptReference类常用的属性及其说明

属性	说明
InlineScript	是否把引入的Web服务资源嵌入到页面的HTML中。默认值为false。如果该属性的值为True，则把引入的Web服务资源直接嵌入到页面的HTML中，并发送到客户端。
Path	引入Web服务资源的路径，一般为相对路径。

在下述代码实例中，AjaxService.asmx Web服务定义了一个静态方法Computer(int leftValue, int rightValue)。该方法计算tbLeftValue和tbRightValue参数的和，然后以字符串形式返回计算结果。AjaxService.asmx Web服务作为Web服务资源，将被Services.aspx页面引用。

```
    using System;
    using System.Web;
    using System.Web.Services;
    using System.Web.Services.Protocols;
    ///引入脚本命名空间
    using System.Web.Script.Services;
```

```
using System.Web.Script;
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService()]
public class AjaxService:System.Web.Services.WebService
{
    /// <summary>
    /// 计算两个数的和
    /// </summary>
    /// <param name="leftValue">左值</param>
    /// <param name="rightValue">右值</param>
    /// <returns>返回两个数的和</returns>
    [WebMethod]
    [System.Web.Script.Services.ScriptMethod]
    public static string Computer(int leftValue,int rightValue)
    {
        return (leftValue + rightValue).ToString();
    }
}
```

在下述代码实例中，Services.aspx页面在ScriptManager控件中引入了AjaxService.asmx Web服务。Services.aspx页面还声明了3个TextBox控件：tbLeftValue、tbRightValue和tbResultValue。其中，tbLeftValue和tbRightValue控件分别用来输入被计算和的两个值（左值和右值），tbResultValue控件显示tbLeftValue和tbRightValue控件的值之和。

另外，Services.aspx页面还声明了一个Button按钮：btnComputer。该按钮还定义了Click事件：btnComputer_Click(object sender,EventArgs e)。该事件调用AjaxService.asmx Web服务中的Computer(int leftValue,int rightValue)方法把leftValue和rightValue值相加，并把相加之后的结果显示在tbResultValue控件中。

```
<!-- AjaxStart/Services.aspx页面 -->
<%@ Page Language="C#" %>
<script runat="server">
protected void btnComputer_Click(object sender,EventArgs e)
{   ///调用Web服务的方法，计算两个数的和
    tbResultValue.Text = AjaxService.Computer(
        Int32.Parse(tbLeftValue.Text),Int32.Parse(tbRightValue.Text));
}
</script>
<head runat="server"><title>引入Web服务资源</title></head>
<asp:ScriptManager ID="sm" runat="server">
    <Services><asp:ServiceReference Path="AjaxService.asmx" /></Services>
</asp:ScriptManager>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
    左值：<asp:TextBox ID="tbLeftValue" runat="server"></asp:TextBox><br />
    右值：<asp:TextBox ID="tbRightValue" runat="server"></asp:TextBox><br />
    结果：<asp:TextBox ID="tbResultValue" runat="server"></asp:TextBox>
    <asp:Button ID="btnComputer" runat="server" Text="计算两个数的和"
        OnClick="btnComputer_Click" />
</ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，分别在【左值】和【右值】输入框中输入“123”和“456”，然后单击【计算两个数的和】按钮计算上述两个值的和，结果如图1.15所示。

图1.15　演示ScriptManager控件引入Web服务资源的功能

1.5.6　处理AJAX中的异常

ASP.NET包含多种处理异常的方法。通常情况下，如果不处理应用程序抛出的异常，那么网页将直接显示异常信息。如图1.16所示，AjaxStart应用程序抛出异常信息“这是一个ASP.NET AJAX环境中的异常”。

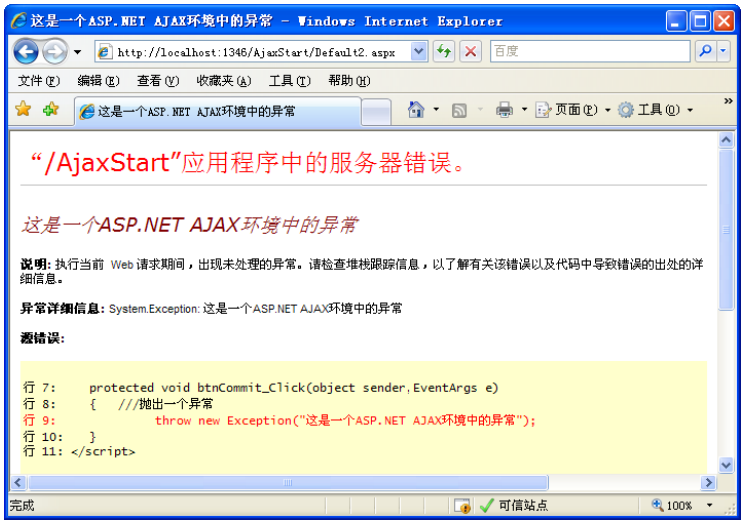

图1.16　AjaxStart应用程序抛出异常信息

然而，上述抛出的异常信息是非常不友好的，有时甚至还会泄漏应用程序的一些敏感信息。值得幸运的是，ASP.NET AJAX考虑了这些情况，它也提供了处理AJAX环境中的异常机制。ScriptManager控件包含了2个与异常相关的重要属性：AllowCustomErrorsRedirect和AsyncPostBackErrorMessage。其中，AllowCustomErrorsRedirect属性指定是否允许使用用户自定义页面来处理异常。AsyncPostBackErrorMessage属性指定当应用程序发生异常时显示的消息。

在下述代码实例中，AjaxException.aspx页面声明了1个Button控件，ID属性的值为

btnCommit。单击该控件将触发其Click事件：btnCommit_Click(object sender,EventArgs e)。该事件将抛出一个异常（为了演示ASP.NET AJX处理异常机制的功能，在此特意抛出该异常）。

为了处理AJAX环境中的异常，特意定义了ScriptManager控件的OnAsyncPostBackError事件：sm_AsyncPostBackError(object sender,AsyncPostBackErrorEventArgs e)。该事件把ScriptManager控件的AsyncPostBackErrorMessage属性的值设置e参数中的异常信息。即当AjaxException.aspx页面发生异常时，它将显示e参数中的异常信息。

注意：当 AjaxException.aspx 页面发生异常时，为了避免页面刷新，因此把 btnCommit 控件注册为支持异步回送的控件。该功能在 AjaxException.aspx 页面的 Page_Load(object sender,EventArgs e)事件中实现。Page_Load(object sender,EventArgs e)事件调用 ScriptManager 控件的 RegisterAsyncPostBackControl()方法把 btnCommit 控件注册为支持异步回送的控件。

```
<!-- AjaxStart/AjaxException.aspx页面 -->
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender,EventArgs e)
{   ///把btnCommit控件注册为异步回送事件
    sm.RegisterAsyncPostBackControl(btnCommit);
}
protected void btnCommit_Click(object sender,EventArgs e)
{   ///抛出一个异常
    throw new Exception("这是一个ASP.NET AJAX环境中的异常");
}
protected void sm_AsyncPostBackError(object sender,
    AsyncPostBackErrorEventArgs e)
{   ///设置异常显示的消息
    sm.AsyncPostBackErrorMessage = e.Exception.Message;
}
</script>
<head runat="server"><title>处理AJAX中的异常</title></head>
<asp:ScriptManager ID="sm" runat="server"
    AllowCustomErrorsRedirect="false"
    OnAsyncPostBackError="sm_AsyncPostBackError"></asp:ScriptManager>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
    <asp:Button ID="btnCommit" runat="server" Text="单击我，抛出一个异常"
       OnClick="btnCommit_Click" />
</ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，单击【单击我，抛出一个异常】按钮，AjaxException.aspx页面将抛出一个异常，并在对话框中显示异常信息“这是一个ASP.NET AJAX环境中的异常”，如图1.17所示。

图1.17　演示ScriptManager控件处理AJAX中的异常的功能

1.6　局部更新控件UpdatePanel

本小节将介绍与UpdatePanel控件相关的内容，如UpdatePanel控件的属性、方法、局部更新、整页回送、多个UpdatePanel控件的更新方式等。

1.6.1　UpdatePanel控件概述

UpdatePanel控件是一个面板，它往往充当一个容器，能够实现局部更新功能。若Web窗体页只需要更新其部分内容时，用户可以把更新的内容事先放置在UpdatePanel控件中。UpdatePanel控件能够实现只更新其区域中的内容，而不需要刷新整个Web窗体页，即不需要把整个Web窗体页提交到服务器。上述功能其实就是一种“局部更新功能”。

在ASP.NET AJAX Web应用程序中，ScriptManager和UpdatePanel控件共同实现无刷新的Web窗体页环境。若Web窗体页只需要更新UpdatePanel控件中的内容，该控件能够自动或在给定条件下更新其区域中的数据，并且把这种更新操作隐藏在Web窗体页的背后。用户将不会感觉到这一更新操作，即用户不会感觉到被更新的Web窗体页的刷新操作。

在下述代码实例中，UpdatePanelCtrl.aspx页面使用ScriptManager和UpdatePanel控件共同实现无刷新的Web窗体页。UpdatePanelCtrl.aspx页面在UpdatePanel控件up中放置了1个GridView控件和1个SqlDataSource控件，它们的ID属性的值分别为gvUser和sqlDSUser。其中，sqlDSUser控件是一个数据源控件，它从AjaxInstantMessagingDB数据库（第15章的数据库）的User表中获取数据。另外，sqlDSUser控件获取数据时连接数据库的连接字符串存放在Web.Config配置文件中，程序代码如下：

```
    <add name="SQLCONNECTIONSTRING"
       connectionString="Data Source=localhost;
           Initial Catalog=AjaxInstantMessagingDB;
           Persist Security Info=True;
           User ID=sa;Password=123456"
       providerName="System.Data.SqlClient" />
    </connectionStrings>
```

gvUser控件以列表形式显示sqlDSUser控件中的数据。它包含了8个BoundField，分别显示User表中的ID、Username、Aliasname、UserIdentity、Signing、PictureUrl、Email和CreateDate列的值。另外，gvUser控件还启用了分页（每一页的数据量为10）和排序功能。

```
<!-- AjaxStart/UpdatePanelCtrl.aspx页面 -->
<%@ Page Language="C#" StylesheetTheme="ASPNETAjaxWeb" %>
<head runat="server"><title>无刷新Web环境</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
   <asp:GridView ID="gvUser" runat="server" AllowPaging="True"
      RowStyle-HorizontalAlign="Center" AllowSorting="True"
      AutoGenerateColumns="False" DataKeyNames="ID"
      DataSourceID="sqlDSUser" SkinID="gvSkin">
      <Columns>
         <asp:BoundField DataField="ID" HeaderText="ID"
            InsertVisible="False" ReadOnly="True" SortExpression="ID" />
         <asp:BoundField DataField="Username" HeaderText="Username"
            SortExpression="Username" />
         <asp:BoundField DataField="Aliasname" HeaderText="Aliasname"
            SortExpression="Aliasname" />
         <asp:BoundField DataField="UserIdentity"
            HeaderText="UserIdentity" SortExpression="UserIdentity" />
         <asp:BoundField DataField="Signing" HeaderText="Signing"
            SortExpression="Signing" />
         <asp:BoundField DataField="PictureUrl" HeaderText="PictureUrl"
            SortExpression="PictureUrl" />
         <asp:BoundField DataField="Email" HeaderText="Email"
            SortExpression="Email" />
         <asp:BoundField DataField="CreateDate" HeaderText="CreateDate"
            SortExpression="CreateDate" HtmlEncode="false"
            DataFormatString="{0:d}" />
      </Columns>
   </asp:GridView>
   <asp:SqlDataSource ID="sqlDSUser" runat="server" ConnectionString="<%$
      ConnectionStrings:SQLCONNECTIONSTRING %>"
      SelectCommand="SELECT [ID], [Username], [Aliasname],
      [UserIdentity], [Signing], [PictureUrl], [Email], [CreateDate]
      FROM [User]">
   </asp:SqlDataSource>
</ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，如图1.18所示。单击分页栏中的【6】链接，UpdatePanelCtrl.aspx页面以无刷新方式显示第6页的数据，如图1.19所示。

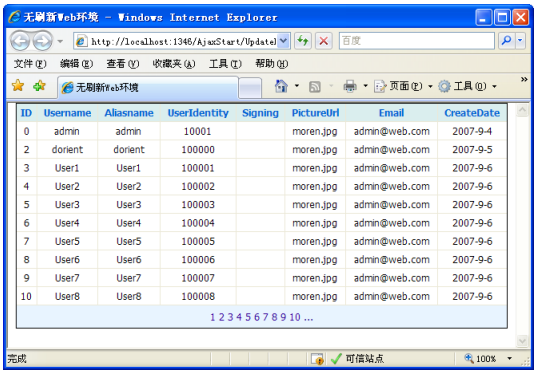

图1.18　UpdatePanelCtrl.aspx页面的初始界面

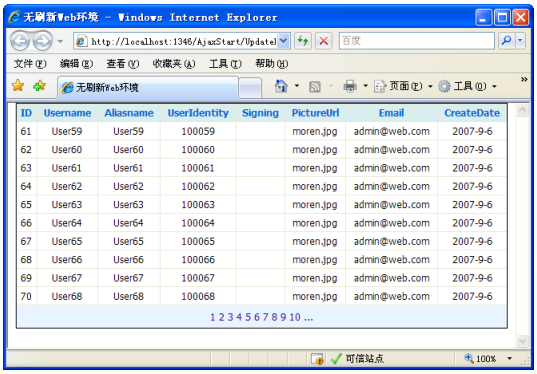

图1.19　UpdatePanelCtrl.aspx页面显示第6页数据

1.6.2　UpdatePanel控件的属性

UpdatePanel控件提供了多个重要属性，如UpdateMode、IsInPartialRendering、Triggers、ChildrenAsTriggers、RenderMode等，具体说明如表1-5所示。

表1-5　UpdatePanel控件的常用属性及其说明

名称	说明
ContentTemplate	内容模板。用户可以在模板中放置控件、HTML代码等内容。
UpdateMode	更新模式。
IsInPartialRendering	是否为部分更新模式。
ChildrenAsTriggers	当控件回发到服务器时，表示是否更新其内容。
Triggers	UpdatePanelTrigger对象的集合。
RequiresUpdate	控件的内容是否能够被更新。
RenderMode	呈现模式。
ContentTemplateContainer	内容模板的容器。

UpdateMode属性指定UpdatePanel控件的更新模式，它的值可以为Always或Conditional。其中，默认值为Always。默认情况下，在Web窗体页请求服务器之后，UpdatePanel控件总是更新其内容。如果用户希望手动控制UpdatePanel控件更新内容，则可以把UpdatePanel控件的UpdateMode属性的值设置为Conditional即可。

注意：IsInPartialRendering 属性是一个只读属性，它获取 UpdatePanel 控件的更新模式，即是否为部分更新默认。如果是，则返回 true，否则返回 false。

当一个Web窗体页存在多个UpdatePanel控件时，ChildrenAsTriggers属性可以指定某一个UpdatePanel控件是否作为其他UpdatePanel控件的Trigger。即可以指定某一个UpdatePanel控件是否跟随其他UpdatePanel控件同时更新。

Triggers属性可以指定一个集合。用户可以在向该集合中添加实现局部更新和整页回送的Trigger（一般由控件实现）。它的具体用法将在1.6.5小节中介绍。

1.6.3　UpdatePanel控件的方法

UpdatePanel控件只提供了一个方法：Update()。该方法可以以程序方式、动态地、实时地更新UpdatePanel控件中的内容。下述代码实例首先设置Label控件lbTime显示当前时间，然后调用了UpdatePanel控件up的Update()方法更新up控件的内容，并使得lbTime控件呈

现其被修改后的值（即当前时间）。

```
///UpdatePanel控件up中放置了一个Label控件lbTime显示当前时间。
lbTime.Text = DateTime.Now.ToString();
up.Update();
```

1.6.4　局部更新

UpdatePanel控件最重要的功能就是实现页面局部更新，而不需要更新整个页面。在下述代码实例中，PageUpdate.aspx页面演示了UpdatePanel控件的局部更新功能。该页面声明了1个Label控件和1个Button控件，它们的ID属性的值分别为lbTime和btnCommit。其中，lbTime控件显示当前时间，btnCommit控件定义了Click事件：btnCommit_Click(object sender, EventArgs e)。当用户单击【单击我，更新时间】按钮（btnCommit控件）时，将触发其Click事件，并在该事件中更新lbTime控件显示的时间。

```
<!-- AjaxStart/PageUpdate.aspx页面 -->
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender,EventArgs e)
{
    if(!Page.IsPostBack)
    {   ///显示当前时间
        lbTime.Text = "当前时间为：" + DateTime.Now.ToString();
    }
}
protected void btnCommit_Click(object sender,EventArgs e)
{   ///显示当前时间
    lbTime.Text = "当前时间为：" + DateTime.Now.ToString();
}
</script>
<head runat="server"><title>局部更新</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
    <asp:Label ID="lbTime" runat="server"></asp:Label>
    <asp:Button ID="btnCommit" runat="server" Text="单击我，更新时间"
       OnClick="btnCommit_Click" />
</ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，PageUpdate.aspx页面显示当前时间，如图1.20所示。当用户单击【单击我，更新时间】按钮时，PageUpdate.aspx页面将在lbTime控件中重新显示当前时间，如图1.21所示。值得注意的是，单击【单击我，更新时间】按钮时，PageUpdate.aspx页面不会被刷新。

图1.20　PageUpdate.aspx页面的初始界面

图1.21　PageUpdate.aspx页面无刷新更新时间

1.6.5　整页回送

在默认情况下，UpdatePanel控件实现页面局部更新的功能。然而有时候Web窗体页要求UpdatePanel控件中的控件（如按钮控件）能够把整个Web窗体页回送到服务器。要实现该功能，则需要使用UpdatePanel控件的PostBackTrigger对象。该对象可以指定执行回送到服务器操作的控件。因此，当用户单击PostBackTrigger对象指定的控件时，整个Web窗体页将被回送到服务器。

在下述代码实例中，PagePostBackUpdate.aspx页面演示了UpdatePanel控件实现整页回送的功能。该页面声明了1个Label控件和1个Button控件，它们的ID属性的值分别为lbTime和btnCommit。其中，lbTime控件显示当前时间，btnCommit控件定义了Click事件：btnCommit_Click(object sender, EventArgs e)。当用户单击【单击我，更新时间】按钮（btnCommit控件）时将触发其Click事件，并在该事件中更新lbTime控件显示的时间。

注意：btnCommit 控件放置在 UpdatePanel 控件 up 中，而 lbTime 控件没有放置在 UpdatePanel 控件 up 中。因此，当用户单击【单击我，更新时间】按钮（btnCommit 控件）时，它只更新 UpdatePanel 控件 up 中的内容，而不会更新 lbTime 控件中的内容。

UpdatePanel控件为了实现整页回送的功能，它把btnCommit控件添加到PostBackTrigger对象中。因此，当用户单击【单击我，更新时间】按钮（btnCommit控件）时，PagePostBackUpdate.aspx页面的所有内容都将被回送到服务器，此时，它将更新lbTime控件中的内容。值得注意的是，在此更新过程中，PagePostBackUpdate.aspx页面被刷新了一次。

```
<!-- AjaxStart/PagePostBackUpdate.aspx页面 -->
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender,EventArgs e)
{
    if(!Page.IsPostBack)
    {   ///显示当前时间
        lbTime.Text = "当前时间为：" + DateTime.Now.ToString();
    }
}
protected void btnCommit_Click(object sender,EventArgs e)
{   ///显示当前时间
    lbTime.Text = "当前时间为：" + DateTime.Now.ToString();
}
```

```
    </script>
    <head id="Head1" runat="server"><title>整页回送</title></head>
    <asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
        <asp:Button ID="btnCommit" runat="server" Text="单击我，更新时间"
            OnClick="btnCommit_Click" />
        </ContentTemplate>
        <Triggers><asp:PostBackTrigger ControlID="btnCommit" /></Triggers>
    </asp:UpdatePanel>
    <asp:Label ID="lbTime" runat="server"></asp:Label>
```

上述代码实例的执行之后，PagePostBackUpdate.aspx页面显示当前时间，如图1.22所示。当用户单击【单击我，更新时间】按钮时，PagePostBackUpdate.aspx页面将在lbTime控件中重新显示当前时间，如图1.23所示。值得注意的是，单击【单击我，更新时间】按钮时，PagePostBackUpdate.aspx页面已经被刷新了一次。

图1.22　PagePostBackUpdate.aspx页面的初始界面

图1.23　显示回送后的时间

1.6.6　多个UpdatePanel控件的更新方式

若一个Web窗体页上放置多个UpdatePanel控件。默认情况下，单击任何一个UpdatePanel控件中的控件都会自动更新Web窗体页上所有UpdatePanel控件中的内容。因为UpdatePanel控件的UpdateMode属性的默认值为“Always”。

在下述代码实例中，MutilUpdatePanel.aspx页面演示了多个UpdatePanel控件的更新方式。其中，UpdatePanel控件的ID属性的值分别为up和upTime。该页面还声明了2个Label控件和1个Button控件，它们的ID属性的值分别为lbTime、lbupTime和btnCommit。其中，lbTime和lbupTime控件都显示当前时间，btnCommit控件定义了Click事件：btnCommit_Click(object sender, EventArgs e)。当用户单击【单击我，更新时间】按钮（btnCommit控件）时将触发其Click事件，并在该事件中更新lbTime和lbupTime控件显示的时间。

注意：lbTime 和 lbupTime 控件分别放置在不同的 UpdatePanel 控件中。其中，lbTime 控件放置在 up 控件中，lbupTime 控件放置在 upTime 控件中。

up控件的UpdateMode属性的默认值为“Always”，而upTime控件的UpdateMode属性的值设置为“Conditional”。因此，当用户单击【单击我，更新时间】按钮（btnCommit控件）时，upup控件中的内容被更新，而upTime控件中的内容不会被更新。

```
    <!-- AjaxStart/MutilUpdatePanel.aspx页面 -->
    <%@ Page Language="C#" %>
    <script runat="server">
```

```
    protected void Page_Load(object sender,EventArgs e)
    {
        if(!Page.IsPostBack)
        {   ///显示当前时间
            lbupTime.Text = "当前时间为(upTime控件)：" + DateTime.Now.ToString();
            lbTime.Text = "当前时间为(up控件)：" + DateTime.Now.ToString();
        }
    }
    protected void btnCommit_Click(object sender,EventArgs e)
    {   ///显示当前时间
        lbupTime.Text = "当前时间为(upTime控件)：" + DateTime.Now.ToString();
        lbTime.Text = "当前时间为(up控件)：" + DateTime.Now.ToString();
    }
    </script>
    <head id="Head1" runat="server">
        <title>多个UpdatePanel控件的更新方式</title>
    </head>
    <asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
        <asp:Button ID="btnCommit" runat="server" Text="单击我，更新时间"
            OnClick="btnCommit_Click" /><br />
        <asp:Label ID="lbTime" runat="server"></asp:Label>
    </ContentTemplate></asp:UpdatePanel>
    <asp:UpdatePanel ID="upTime" runat="server" UpdateMode="Conditional">
        <ContentTemplate>
            <asp:Label ID="lbupTime" runat="server"></asp:Label>
        </ContentTemplate>
    </asp:UpdatePanel>
```

上述代码实例的执行之后，MutilUpdatePanel.aspx页面在lbTime和lbupTime控件中都显示当前时间（“2007-10-2 13:52:06”），如图1.24所示。

当用户单击【单击我，更新时间】按钮时，MutilUpdatePanel.aspx页面将更新lbTime控件显示的时间(“2007-10-2 13:52:24”)。但是，lbupTime控件显示的时间不会更新(“2007-10-2 13:52:06”），如图1.25所示。

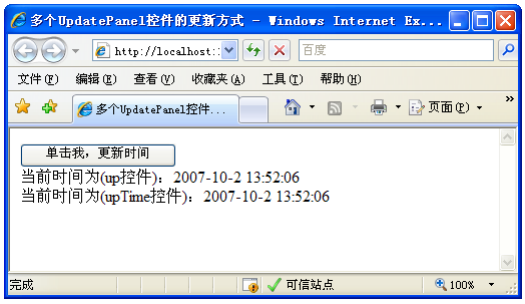

图1.24　初始界面（一）

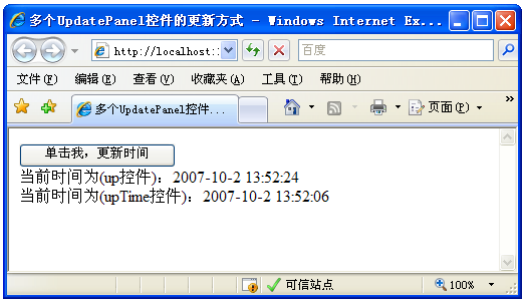

图1.25　显示部分更新后的时间

注意：在上述实例中，如果把 up 和 upTime 控件的 UpdateMode 属性的值都设置为默认值“Always”，那么 MutilUpdatePanel.aspx 页面将显示不同的运行结果。

在上述条件下，MutilUpdatePanel.aspx页面运行之后，MutilUpdatePanel.aspx页面在lbTime和lbupTime控件中都显示当前时间（“2007-10-2 13:52:56”），如图1.26所示。

当用户单击【单击我，更新时间】按钮时，MutilUpdatePanel.aspx页面将同时更新lbTime和lbupTime控件显示的时间（“2007-10-2 13:53:09”），如图1.27所示。

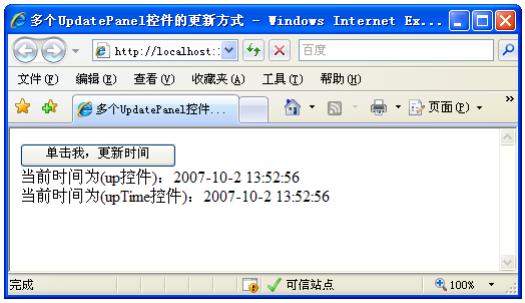

图1.26　初始界面（一）

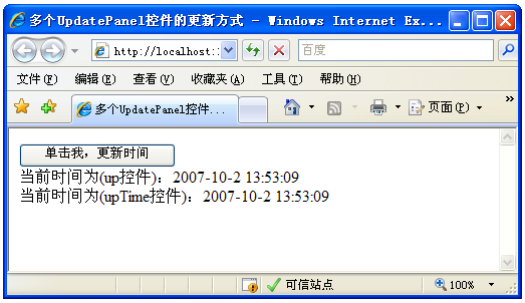

图1.27　显示全部更新后的时间

1.7　ASP.NET AJAX其他服务器控件

本小节介绍ASP.NET AJAX中除了ScriptManager和UpdatePanel控件之外的服务器端控件，如计时器控件Timer、更新进度条控件UpdateProgress、管理脚本的ScriptManagerProxy控件等。最后还介绍了在ASP.NET AJAX环境中弹出对话框的方法。

1.7.1　计时器控件Timer

Timer控件又称为计时器控件。它能够指定一个时间间隔和Tick事件，在每一次时间间隔到达之后，将触发其Tick事件。因此，该控件能够定时执行事先指定的一些操作。例如，使用该控件可以定时刷新Web窗体页或其部分内容。Timer控件的属性、方法和事件如表1-6所示。

表1-6　Timer控件的常用属性、方法、事件及其说明

属性、方法和事件	说明
Enabled	表示是否启用控件。
Interval	时间间隔，单位为毫秒。
GetScriptDescriptors()	获取与控件相关的组件、行为描述列表。
GetScriptReferences()	获取与控件相关的、引用的脚本资源。
RaisePostBackEvent()	当Web窗体页提交到服务器时，使控件产生一个Tick事件。
Tick（事件）	定时触发的事件。两次触发时间的间隔由Interval属性指定
OnTick（事件）	Tick事件。

Interval属性指定两次触发Tick事件所间隔的时间，它的单位为毫秒。在下述代码实例中，TimerCtrl.aspx页面声明了1个Label控件和1个Timer控件，它们的ID属性的值分别为lbTime和tTime。其中，lbTime控件用来显示当前时间。tTime控件是一个定时器控件，它定义了Tick事件：tTime_Tick(object sender,EventArgs e)，并设置Interval属性的值为1000。即tTime控件每隔1秒钟将触发tTime_Tick(object sender,EventArgs e)事件一次。

其中，tTime_Tick(object sender,EventArgs e)事件设置lbTime控件显示当前时间。由于该事件每隔1秒钟执行一次，因此，lbTime控件显示的时间也每隔1秒钟更新一次。

```
    <!-- AjaxStart/TimerCtrl.aspx页面 -->
```

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender,EventArgs e)
{
    if(!Page.IsPostBack)
    {   ///显示当前时间
        lbTime.Text = "当前时间为: " + DateTime.Now.ToString();
    }
}
protected void tTime_Tick(object sender,EventArgs e)
{   ///显示当前时间
    lbTime.Text = "当前时间为: " + DateTime.Now.ToString();
}
</script>
<head runat="server"><title>计时器控件Timer</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
    <asp:Label ID="lbTime" runat="server"></asp:Label>
    <asp:Timer ID="tTime" runat="server" Interval="1000"
        OnTick="tTime_Tick"></asp:Timer>
</ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，如图1.28所示。TimerCtrl.aspx页面每隔1秒钟显示当前时间，等待15秒之后，TimerCtrl.aspx页面如图1.29所示。

图1.28　TimerCtrl.aspx页面显示某一个时间

图1.29　TimerCtrl.aspx页面显示另外一个时间

1.7.2　更新进度条控件UpdateProgress

UpdateProgress控件又称为更新进程条控件。这是一个非常特殊的控件，它能够显示Web窗体页和服务器之间的交互过程，并且在该过程中能够以一种友好的方式向用户显示Web窗体页和服务器正在进行交互。UpdateProgress控件的属性、方法及其说明如表1-7所示。

表1-7　UpdateProgress控件的常用属性、方法及其说明

属性和方法	说明
AssociatedUpdatePanelID	与控件相关联的UpdatePanel控件的ID属性的值。
DisplayAfter	显示UpdateProgress控件之前等待的时间。
DynamicLayout	ProgressTemplate是否动态呈现。
ProgressTemplate	内容模板。
GetScriptDescriptors()	获取与控件相关的组件、行为描述列表。

GetScriptReferences()	获取与控件相关的、引用的脚本资源。

AssociatedUpdatePanelID属性指定与UpdateProgress控件相关联的UpdatePanel控件。该UpdateProgress控件将显示AssociatedUpdatePanelID属性指定的UpdatePanel控件与Web服务器交互的过程。当UpdatePanel控件与Web服务器没有发生交互时，UpdateProgress控件不显示任何内容。只有当UpdatePanel控件与Web服务器发生交互时，UpdateProgress控件显示这一个交互的过程。

注意：用户还可以在 UpdateProgress 控件中实现取消 Web 窗体页和服务器之间交互的功能。

从UpdatePanel控件与Web服务器发生交互动作时开始，等待DisplayAfter属性指定的时间之后，UpdateProgress控件才显示其内容。DisplayAfter属性的值单位为毫秒。

在下述代码实例中，ProgressCtrl.aspx页面声明了1个Button控件、1个UpdatePanel控件和1个UpdateProgress控件，它们的ID属性的值分别为btnCommit、up和upPanel。upPanel控件将显示up控件与Web服务器的交互过程。

用户单击【单击我，将开始加载Web窗体页的数据...】按钮（btnCommit控件），该按钮将触发其Click事件：btnCommit_Click(object sender,EventArgs e)，并在该事件中延时5秒钟。此过程中，up控件与Web服务器的交互过程，upPanel控件将显示这一交互过程。单击upPanel控件中的【取消】按钮将取消up控件与Web服务器的交互过程。

```
<!-- AjaxStart/ProgressCtrl.aspx页面 -->
<%@ Page Language="C#" %>
<script runat="server">
protected void btnCommit_Click(object sender,EventArgs e)
{   ///延时5秒钟
    System.Threading.Thread.Sleep(5000);
}
</script>
<head runat="server"><title>更新进度条控件UpdateProgress</title></head>
<asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
<script language="javascript" type="text/javascript">
<!--///获取当前页面的请求实例
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    function Cancel()
    {   ///判断是否为异步传送
        if(prm.get_isInAsyncPostBack() == true)
        {   ///取消当前的传送
            prm.abortPostBack();
        }
    }
//-->
</script>
<asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
    <asp:Button ID="btnCommit" runat="server"
        Text="单击我，将开始加载Web窗体页的数据..." OnClick="btnCommit_Click" />
</ContentTemplate></asp:UpdatePanel>
<asp:UpdateProgress ID="upPanel" runat="server" DisplayAfter="0"
    AssociatedUpdatePanelID="up">
    <ProgressTemplate>
```

```
            Web窗体页的正在加载数据，请等待...<br />
            <input id="btnCancel" type="button" value="取消"
                onclick="Cancel()" />
        </ProgressTemplate>
    </asp:UpdateProgress>
```

上述代码实例的执行之后，如图1.30所示。单击【单击我，将开始加载Web窗体页的数据...】按钮，up控件与Web服务器的交互过程，upPanel控件将显示这一交互过程，如图1.31所示。

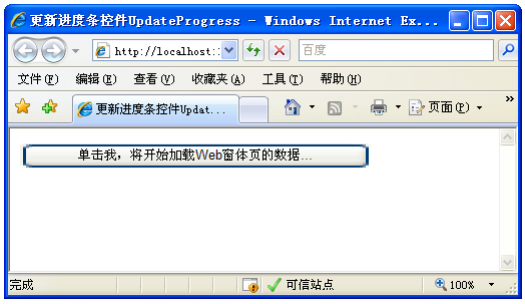

图1.30　ProgressCtrl.aspx页面初始界面

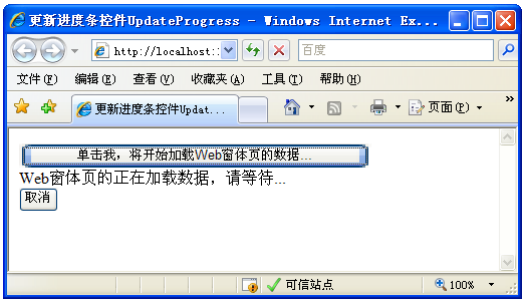

图1.31　upPanel控件显示交互过程

1.7.3　使用ScriptManagerProxy控件管理脚本

ScriptManagerProxy和ScriptManager控件非常相似，它们都能够管理Web窗体页上的脚本。当某一个Web窗体页或用户控件嵌套在另外一个Web窗体页或Master页内，并且在主Web窗体页或Master页中已经包含一个ScriptManager控件时，那么被嵌套的Web窗体页或用户控件将不能再包括ScriptManager控件（因为在一个Web窗体页中只能放置一个ScriptManager控件）。此时，被嵌套的Web窗体页或用户控件将只能使用ScriptManagerProxy控件来管理与AJAX相关的控件和脚本。ScriptManagerProxy控件的属性及其说明如表1-8所示。

表1-8　ScriptManagerProxy控件的常用属性及其说明

属性	说明
AuthenticationService	与当前ScriptManagerProxy实例相关的AuthenticationServiceManager对象。
ProfileService	与当前ScriptManagerProxy实例相关的ProfileServiceManager对象。
Scripts	引入的脚本集合。
Services	引入的Web服务集合。

1.7.4　弹出AJAX环境中对话框

在普通的ASP.NET环境中，要在Web窗体页上弹出一个对话框可以使用如下实例代码。

```
Response.Write("<script>alert('这个是一个对话框')</script>");
```

然而，在ASP.NET AJAX环境中，要在Web窗体页上弹出一个对话框，却不能使用上述代码实例。否则会弹出如图1.32所示的信息提示对话框。

图1.32　在AJAX环境使用Response.Write()弹出对话框的信息提示对话框

值得庆幸的是，ScriptManager类提供了在Web窗体页上弹出对话框的功能。该类使用RegisterClientScriptBlock()方法能够直接向Web窗体页上注册对话框的脚本，从而实现弹出对话框的功能。RegisterClientScriptBlock()方法的原型如下：

（1）public static void RegisterClientScriptBlock(Page page, Type type, string key, string script, bool addScriptTags)

（2）public static void RegisterClientScriptBlock(Control control, Type type, string key, string script, bool addScriptTags)

其中，（1）方法中的page参数为Page对象，（2）方法中的control参数为Control对象。type参数为对象的数据类型。key参数为脚本的标识符，script参数为脚本代码。addScriptTags参数表示是否向注册的脚本代码添加“<script></script>”标记。

注意：（1）方法可以为 Web 窗体页的 Page 对象注册脚本；（2）方法可以为 Web 窗体页的控件注册脚本。

在下述代码实例中，Dialog.aspx页面定义了如下2个私有方法：

- private void OpenDialogForButton(Button button,string message)，为Button控件实现弹出对话框的功能。
- private void OpenDialogForPage(Page page,string message)，为Page对象实现弹出对话框的功能。

```
<!-- AjaxStart/Dialog.aspx页面 -->
<%@ Page Language="C#" %>
<script runat="server">
/// <summary>
/// 在ASP.NET AJAX环境中，为Button控件弹出一个提示对话框
/// </summary>
/// <param name="button">Button控件</param>
/// <param name="message">对话框中的消息</param>
private void OpenDialogForButton(Button button,string message)
{
    ScriptManager.RegisterClientScriptBlock(
        button,
        typeof(Button),
        DateTime.Now.ToString().Replace(":"," "), ///使用当前时间作为标识
        "alert('" + message + "')",
        true);
}
/// <summary>
/// 在ASP.NET AJAX环境中，为Page对象弹出一个提示对话框
/// </summary>
/// <param name="page">Page对象</param>
```

```
    /// <param name="message">对话框中的消息</param>
    private void OpenDialogForPage(Page page,string message)
    {
        ScriptManager.RegisterClientScriptBlock(
            page,
            typeof(Page),
            DateTime.Now.ToString().Replace(":"," "),  ///使用当前时间作为标识
            "alert('" + message + "')",
            true);
    }
    ……
    </script>
```

在下述代码实例中，Dialog.aspx页面声明了1个Button控件，ID属性的值为btnCommit。该控件还定义了其Click事件：btnCommit_Click(object sender,EventArgs e)。该事件将调用OpenDialogForButton(Button button,string message)函数在Dialog.aspx页面上弹出一个对话框。另外，Dialog.aspx页面在第一次运行时，也弹出了一个对话框。

```
    <!-- AjaxStart/Dialog.aspx页面 -->
    <%@ Page Language="C#" %>
    <script runat="server">
    ……
    protected void Page_Load(object sender,EventArgs e)
    {
        if(!Page.IsPostBack)
        {   ///只有页面第一次启动时，才弹出该对话框
            OpenDialogForPage(Page,"这是Page对象弹出的提示对话框");
        }
    }
    protected void btnCommit_Click(object sender,EventArgs e)
    {   ///弹出按钮的对话框
        OpenDialogForButton((Button)sender,"这是Button按钮弹出的提示对话框");
    }
    </script>
    <head runat="server"><title>弹出AJAX环境中对话框</title></head>
    <asp:ScriptManager ID="sm" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="up" runat="server"><ContentTemplate>
        <asp:Button ID="btnCommit" runat="server" Text="单击我，弹出对话框"
            OnClick="btnCommit_Click" />
    </ContentTemplate></asp:UpdatePanel>
```

上述代码实例的执行之后，Dialog.aspx页面首先弹出一个对话框，如图1.33所示。单击【确定】按钮，Dialog.aspx页面的初始界面如图1.34所示。

图1.33　Page对象弹出的对话框

图1.34　Dialog.aspx页面的初始界面

在Dialog.aspx页面中，单击【单击我，弹出对话框】按钮，弹出一个对话框，如图1.35所示。

图1.35　按钮弹出的对话框

1.8　ASP.NET AJAX Control Toolkit

本小节主要介绍与ASP.NET AJAX Control Toolkit相关的内容，如安装ASP.NET AJAX Control Toolkit、导入ASP.NET AJAX Control Toolkit中的控件、引用ASP.NET AJAX Control Toolkit中的程序集、ASP.NET AJAX Control Toolkit中的Web演示站点和Web测试站点等。

1.8.1　ASP.NET AJAX Control Toolkit概述

ASP.NET AJAX Control Toolkit是在ASP.NET AJAX基础之上构建的，提供了数10种在ASP.NET AJAX环境中运行的控件。ASP.NET AJAX Control Toolkit是一个免费资源，任何开发人员或程序员都可以是该资源。它被微软ASP.NET的官方网站所支持，用户可以从该网站上下载。ASP.NET AJAX Control Toolkit主要包括以下4部分内容。

- 数10种基于ASP.NET AJAX的控件，并且还提供了这些控件的源代码。开发人员或程序员可以更加方便的、容易的使用这些控件实现网页中各种特效和功能。特别地，ASP.NET AJAX Control Toolkit提供了这些控件的源代码，开发人员或程序员还可以对这些控件进行优化，或者改进控件存在的缺点。
- Web演示站点，为学习或使用ASP.NET AJAX Control Toolkit中的控件的用户提供了不可多得的实例和指导。
- Web测试站点，演示了ASP.NET AJAX Control Toolkit中的控件的各个功能，为学习或使用这些控件的用户提供了测试实例。
- 自动化测试框架，ASP.NET AJAX Control Toolkit提供了一个自动测试框架。当用户开发自定义的ASP.NET AJAX控件时，就可以使用该框架测试开发的ASP.NET AJAX控件。

1.8.2 安装ASP.NET AJAX Control Toolkit

和ASP.NET AJAX一样，ASP.NET AJAX Control Toolkit也是免费的，它可以从微软ASP.NET的官方网站（http://ajax.asp.net）下载。下载ASP.NET AJAX的具体网址为http://www.asp.net/ajax/downloads/，如图1.36所示。

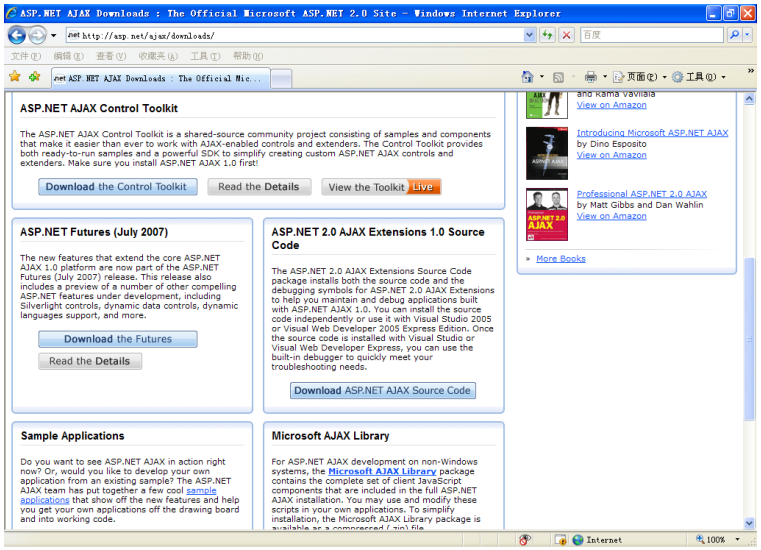

图1.36 下载ASP.NET AJAX Control Toolkit的ASP.NET的官方网站

单击【Download the Control Toolkit】按钮，可以重定向到下载ASP.NET AJAX Control Toolkit的页面（详细地址为http://www.codeplex.com/AtlasControlToolkit/Release/ProjectReleases.aspx?ReleaseId=4941），如图1.37所示。该页面存在4个与ASP.NET AJAX Control Toolkit相关的资源，具体说明如下：

- AjaxControlToolkit.zip：ASP.NET AJAX Control Toolkit控件，且包括其源代码，大小为3440K。
- AjaxControlToolkit-NoSource.zip：ASP.NET AJAX Control Toolkit控件，但是不包括其源代码，大小为2210K。
- AjaxControlToolkit-Framework3.5.zip：ASP.NET AJAX Control Toolkit控件，支持.NET Framework3.5，且包括其源代码，大小为2856K。
- AjaxControlToolkit-Framework3.5-NoSource.zip：ASP.NET AJAX Control Toolkit控件，支持.NET Framework3.5，但是不包括其源代码，大小为1633K。

在此，笔者下载的文件为“AjaxControlToolkit.zip”。本书中所有程序代码都基于该文件中的程序集。

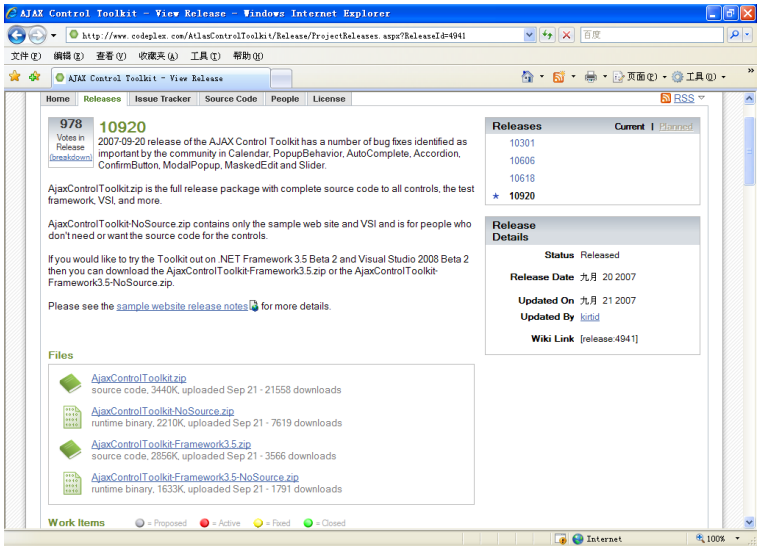

图1.37　下载ASP.NET AJAX Control Toolkit的网页

下载AjaxControlToolkit.zip文件之后，并解压缩到“D:\Project2005\AjaxControlToolkit”目录，可以查看AjaxControlToolkit.zip文件包含的内容，如图1.38所示。具体说明如下：

- AjaxControlToolkit文件夹包含ASP.NET AJAX Control Toolkit控件的源文件。
- Binaries文件夹包含ASP.NET AJAX Control Toolkit控件所需要的程序集。
- SampleWebSite文件夹包含ASP.NET AJAX Control Toolkit控件的Web演示站点的源文件。
- ToolkitTests文件夹包含ASP.NET AJAX Control Toolkit控件的Web测试站点的源文件。

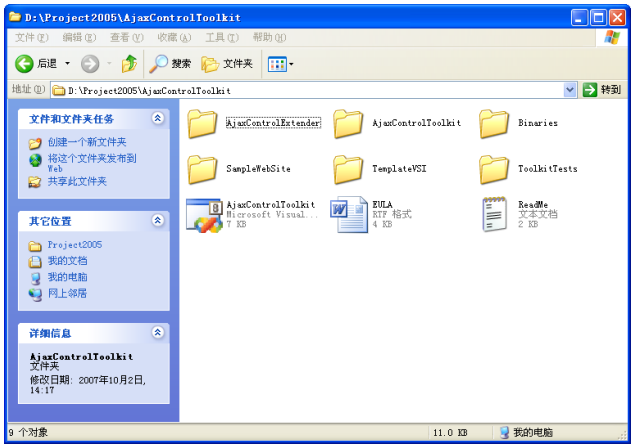

图1.38　AjaxControlToolkit.zip文件包含的文件

使用Visual Studio 2005集成开发环境打开图1.38中的AjaxControlToolkit解决方案，在【解决方案资源管理器】面板中查看AjaxControlToolkit解决方案的内容，如图1.39所示。该解决方案包括4个项目，具体说明如下：

- AjaxControlToolkit项目：创建ASP.NET AJAX Control Toolkit控件。

- SampleWebSite项目（或网站）：创建ASP.NET AJAX Control Toolkit中的Web演示站点。
- ToolkitTests项目（或网站）：创建ASP.NET AJAX Control Toolkit中的Web测试站点。
- TemplateVSI项目：ASP.NET AJAX Control Toolkit控件的辅助项目。

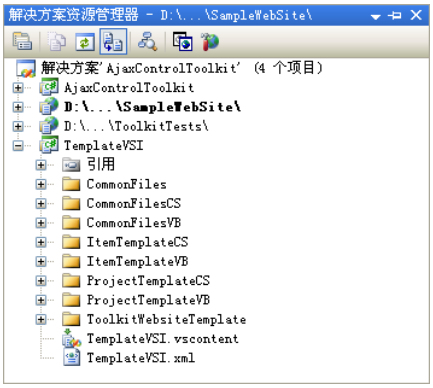

图1.39　【解决方案资源管理器】面板中查看AjaxControlToolkit解决方案

1.8.3　导入ASP.NET AJAX Control Toolkit中的控件

ASP.NET AJAX Control Toolkit的程序集名称为“AjaxControlToolkit.dll”。该程序集包含了ASP.NET AJAX Control Toolkit中的所有控件。为了在Visual Studio 2005集成开发环境中更加方便的使用ASP.NET AJAX Control Toolkit中的控件，可以把ASP.NET AJAX Control Toolkit中的控件导入到Visual Studio 2005集成开发环境的【工具箱】面板中，具体步骤如下：

（1）在【工具箱】面板中，新建一个选项卡，名称为“AJAX Control”，如图1.40所示。

（2）右键单击【AJAX Control】选项卡，并选中【选择项...】命令，弹出【选择工具箱】对话框，如图1.41所示。

图1.40　【工具箱】面板

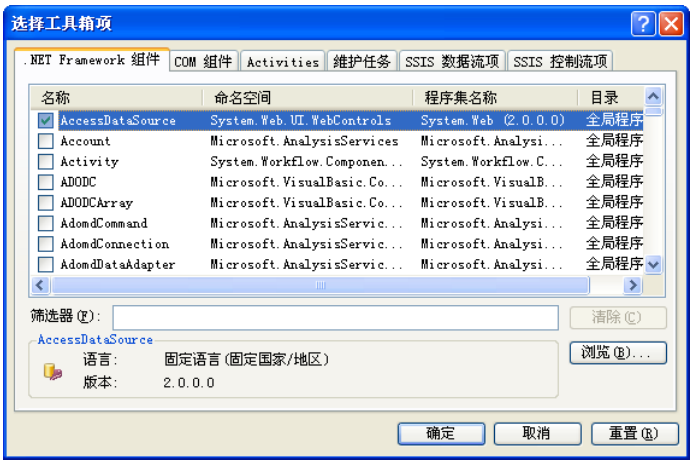

图1.41　【选择工具箱】对话框

（3）单击【浏览】按钮，并在【打开】对话框中选择“D:\Project2005\AjaxControlToolkit”目录的AjaxControlToolkit.dll程序集，如图1.42所示。在【选择工具箱】对话框中，即将被导入的控件将被选中，且背景颜色为蓝色。

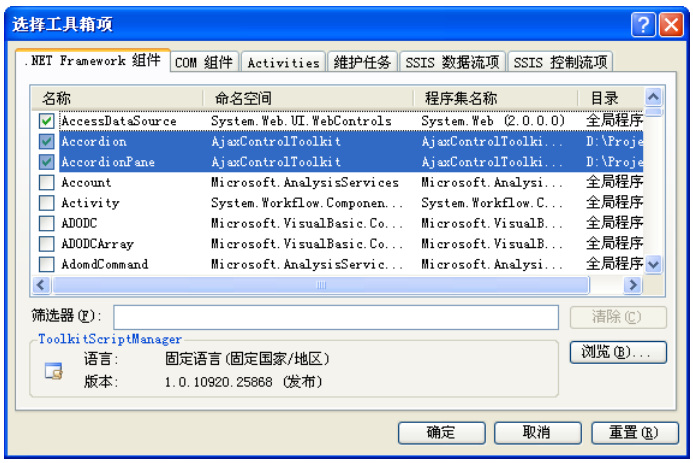

图1.42　选择AjaxControlToolkit.dll程序集

（4）单击【确定】按钮，就可以把AjaxControlToolkit.dll程序集中的控件导入到Visual Studio 2005集成开发环境的【工具箱】面板中。在【工具箱】面板中查看被导入的ASP.NET AJAX Control Toolkit中的控件，如图1.43和图1.44所示。

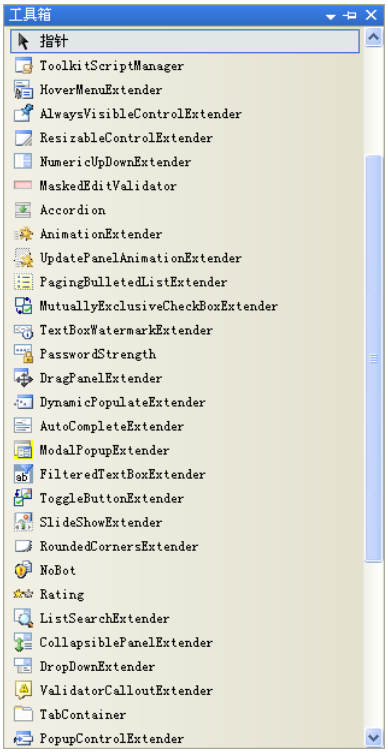

图1.43　【AJAX Control】选项卡的控件（一）

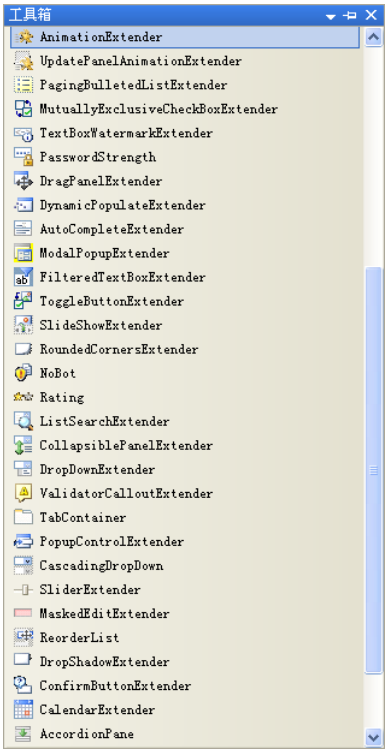

图1.44　【AJAX Control】选项卡的控件（二）

1.8.4 引用ASP.NET AJAX Control Toolkit中的程序集

如果开发人员或程序员在ASP.NET Web应用程序中使用了ASP.NET AJAX Control Toolkit中的控件，默认情况下，ASP.NET Web应用程序会自动把该程序集到引用应用程序中。然而，有时需要开发人员或程序员手工引用ASP.NET AJAX Control Toolkit中的程序集AjaxControlToolkit.dll到应用程序中。具体操作步骤如下：

（1）在Visual Studio 2005集成开发环境中选择要引用AjaxControlToolkit.dll程序集的项目或网站（本实例使用AjaxStart网站）。

（2）右键单击【AjaxStart】节点，单击【添加引用...】命令，弹出【添加引用】对话框，并选中【浏览】选项卡。在此，笔者选择了ASP.NET AJAX Control Toolkit中的程序集AjaxControlToolkit.dll，如图1.45所示。

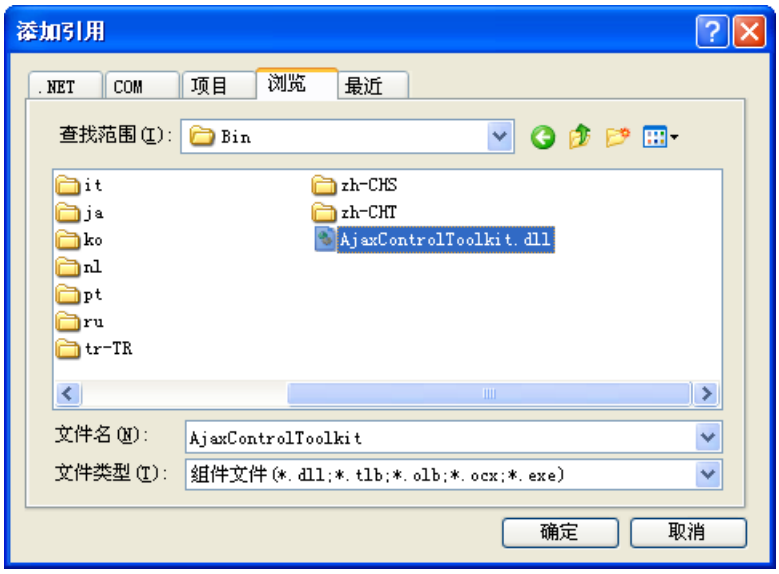

图1.45 选择AjaxControlToolkit.dll程序集

（3）单击【确定】按钮，就可以把ASP.NET AJAX Control Toolkit中的程序集AjaxControlToolkit.dll添加到引用的AjaxStart网站。此时，查看AjaxStart网站的【解决方案资源管理器】面板的“Bin”文件夹，如图1.46所示。

注意：AjaxControlToolkit.dll程序集会默认添加多个不同语言环境下的资源文件（这些资源文件被包含在不同的文件夹中）。

（4）AjaxStart网站在引用AjaxControlToolkit.dll程序集时，会自动引用它的不同语言环境下的资源文件。如果项目或网络使用的语言环境比较单一（如简体中文），那么可以将“Bin”文件夹中除了简体中文之外的资源文件夹全部删除。在此，AjaxStart网站只在简体中文环境中使用，因此笔者只保留了“zh-CHS”文件夹（对应于简体中文），其他的资源文件夹全部被删除，如图1.47所示。

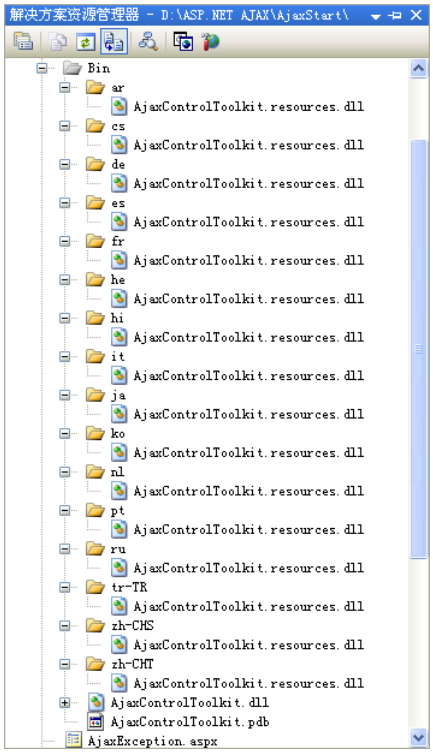

图1.46　AjaxStart网站的“Bin”文件夹

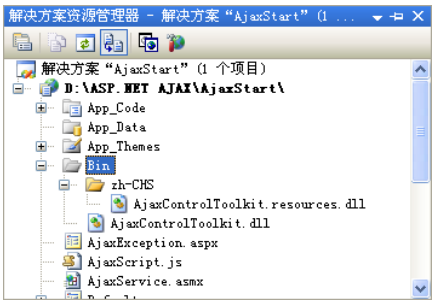

图1.47　只保留“zh-CHS”文件夹

1.8.5　ASP.NET AJAX Control Toolkit中的Web演示站点

使用Visual Studio 2005集成开发环境打开图1.38中的AjaxControlToolkit解决方案。该解决方案包括4个项目：AjaxControlToolkit、SampleWebSite、ToolkitTests和TemplateVSI。其中，SampleWebSite项目（或网站）创建ASP.NET AJAX Control Toolkit中的Web演示站点。在【解决方案资源管理器】面板中，查看SampleWebSite网站包括的具体内容，如图1.48所示。查看该站点内容的具体步骤如下：

（1）把AjaxControlToolkit解决方案中的SampleWebSite网站设置为启动项目。右键单击【解决方案资源管理器 - D:\...\SampleWebSite\】面板中的【D:\...\SampleWebSite\】节点，并选中【设为启动项目】命令，操作如图1.49所示。

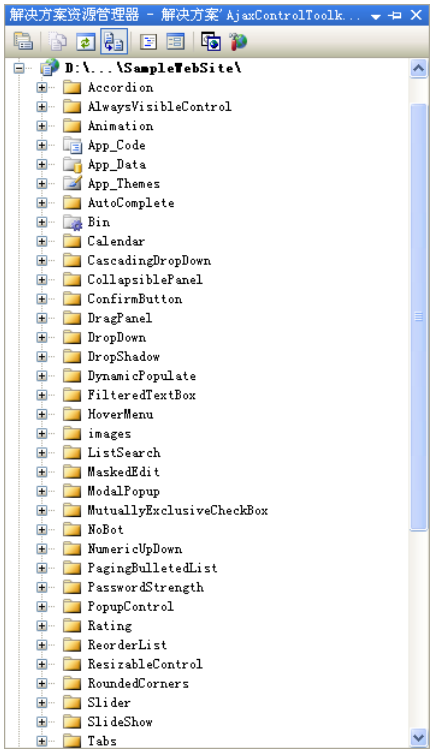

图1.48　SampleWebSite网站

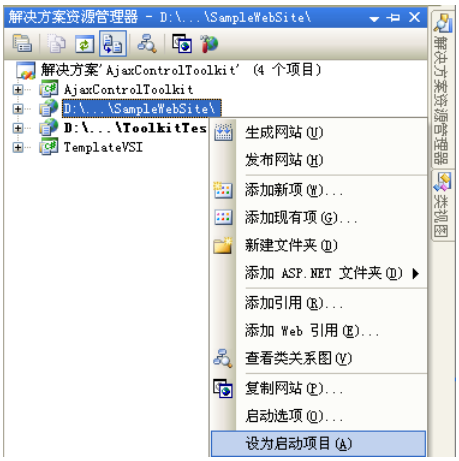

图1.49　把SampleWebSite网站设置为启动项目

（2）把Default.aspx页面设置为SampleWebSite网站的起始页面，并按F5运行该网站。在IE浏览器中查看Default.aspx页面，如图1.50所示。此时，Default.aspx页面显示了ASP.NET AJAX Control Toolkit中以下控件的演示链接，单击这些链接就可以查看其相对应控件的演示效果。

- Accordion，播放动画的控件。
- AlwaysVisibleControl，实现总是浮动可见的面板。
- Animation，播放动画的控件。
- AutoComplete，实现自动输入建议的功能。
- Calendar，日历控件。
- CascadingDropDown，实现级联下拉选择的控件。
- CollapsiblePanel，实现可折叠/扩展的面板。
- ConfirmButton，弹出确认对话框。
- DragPanel，实现可拖拉的面板。
- DropDown，实现下拉式菜单。
- DropShadow，实现显示阴影的面板。
- DynamicPopulate，实现动态弹出的面板。
- FilteredTextBox，实现过滤字符的输入。
- HoverMenu，盘旋式菜单。
- ListSearch，实现列表搜索功能。
- MaskedEdit，控制用户输入并指定格式。
- ModalPopup，弹出模态对话框。

- MutuallyExclusiveCheckBox，实现多复选框之间的互斥功能。
- NoBot，拒绝机器人登录。

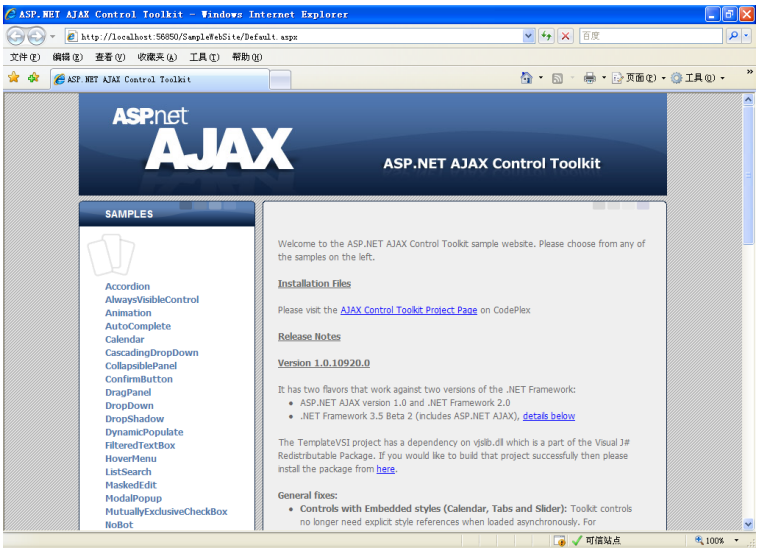

图1.50　SampleWebSite网站的起始页面Default.aspx（一）

（3）拖动Default.aspx页面中的滚动条，可以查看ASP.NET AJAX Control Toolkit中以下控件的演示链接，单击这些链接就可以查看其相对应控件的演示效果，如图1.51所示。

- NumericUpDown，实现自动增减的输入功能。
- PagingBulletedList，实现导航分页式列表。
- PasswordStrength，密码强度提示功能。
- PopupControl，实现弹出控件的功能。
- Rating，等级控件。
- ReorderList，动态排列数据的列表。
- ResizableControl，实现可变大小的控件。
- RoundedCorners，实现圆角的控件。
- Slider，滑动条控件。
- SlideShow，实现播放照片或图片的功能。
- Tabs，选项卡控件。
- TextBoxWatermark，为文本输入框添加水印。
- ToggleButton，用图片代替复选框。
- UpdatePanelAnimation，实现多样式更新面板。
- ValidatorCallout，实现多样式验证。

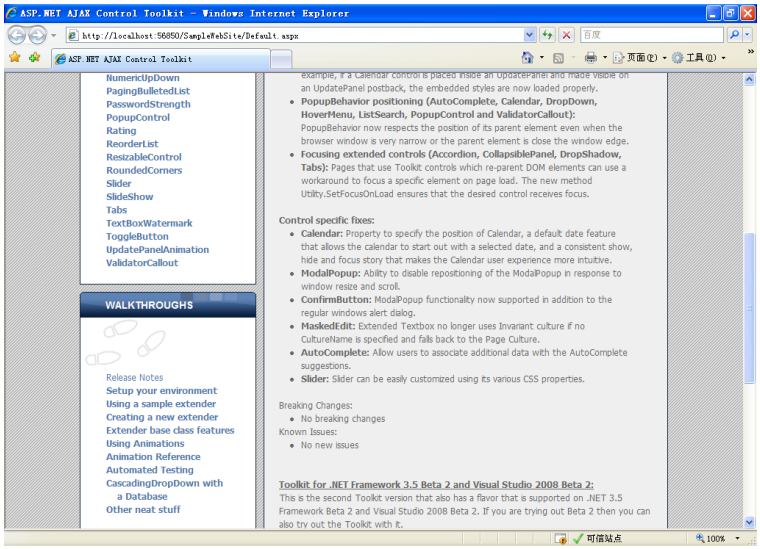

图1.51　SampleWebSite网站的起始页面Default.aspx（二）

1.8.6　ASP.NET AJAX Control Toolkit中的Web测试站点

使用Visual Studio 2005集成开发环境打开图1.38中的AjaxControlToolkit解决方案。该解决方案包括4个项目：AjaxControlToolkit、SampleWebSite、ToolkitTests和TemplateVSI。其中，ToolkitTests项目（或网站）创建ASP.NET AJAX Control Toolkit中的Web测试站点。在【解决方案资源管理器】面板中，查看ToolkitTests网站包括的具体内容，如图1.52所示。查看该站点内容的具体步骤如下：

（1）把AjaxControlToolkit解决方案中的ToolkitTests网站设置为启动项目。右键单击【解决方案资源管理器 - D:\...\ToolkitTests\】面板中的【D:\...\ToolkitTests\】节点，并选中【设为启动项目】命令，操作如图1.53所示。

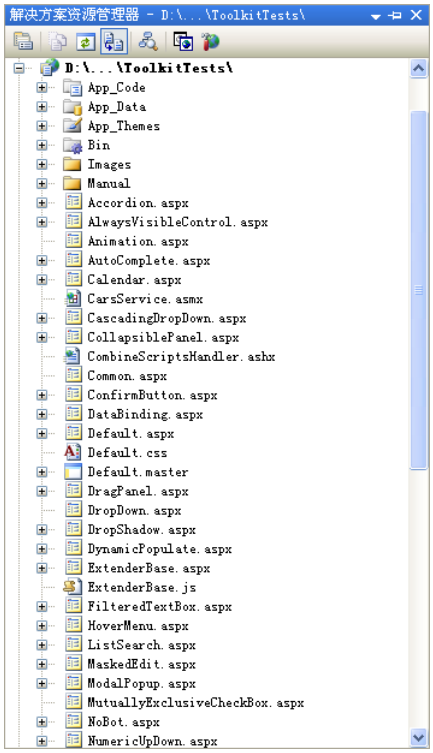

图1.52　ToolkitTests网站

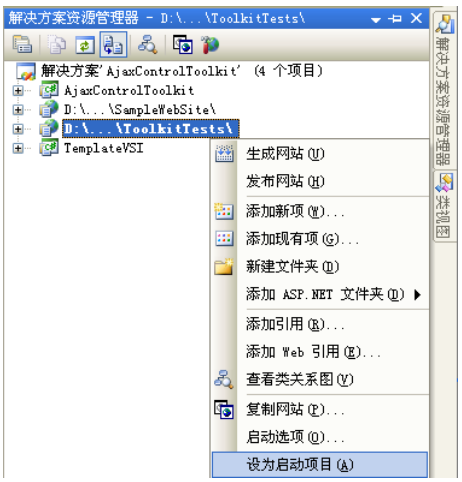

图1.53　把ToolkitTests网站设置为启动项目

（2）把Default.aspx页面设置为ToolkitTests网站的起始页面，并按F5运行该网站。在IE浏览器中查看Default.aspx页面，如图1.54所示。此时，Default.aspx页面显示了ASP.NET AJAX Control Toolkit中以下控件的测试，选中这些控件前面的复选项，并单击【Run Tests】按钮就可以测试选中的控件。

- TestHarnessTests。
- ExtenderBase，演示扩展控件的基础功能。
- Accordion，演示播放动画的控件。
- AlwaysVisibleControl，演示总是浮动可见的面板。
- Animation，演示播放动画的控件。
- AutoComplete，演示自动输入建议的功能。
- Calendar，演示日历控件。
- CascadingDropDown，演示级联下拉选择的控件。
- CollapsiblePanel，演示可折叠/扩展的面板。
- ConfirmButton，演示弹出确认对话框。
- DataBinding，演示数据绑定功能。
- DragPanel，演示可拖拉的面板。
- DropDown，演示下拉式菜单。
- DropShadow，演示显示阴影的面板。
- DynamicPopulate，演示动态弹出的面板。
- FilteredTextBox，演示过滤字符的输入。
- HoverMenu，演示盘旋式菜单。

- ListSearch，演示列表搜索的功能。
- MaskedEdit，演示控制用户输入并指定格式的功能。
- ModalPopup，演示弹出模态对话框。
- MutuallyExclusiveCheckBox，演示多复选框之间的互斥功能。
- NoBot，演示拒绝机器人登录。
- NumericUpDown，演示自动增减的输入功能。
- PagingBulletedList，演示导航分页式列表。
- PasswordStrength，演示密码强度提示功能。

（3）在Default.aspx页面中，选中【Animation】复选框，单击【Run Tests】按钮就可以测试Animation控件，测试结果如图1.55所示。

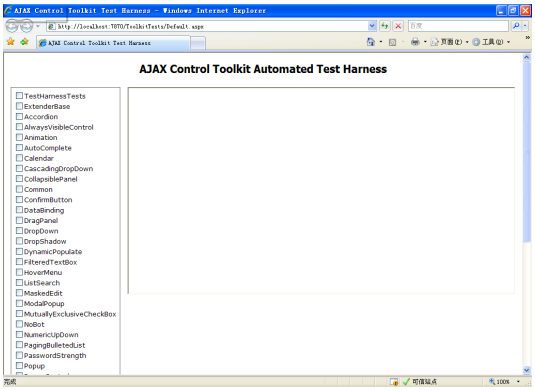

图1.54　ToolkitTests网站的起始页面Default.aspx

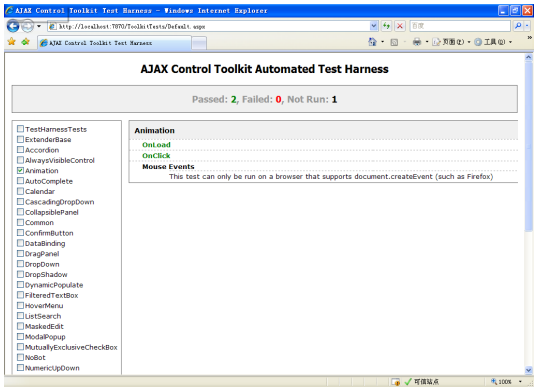

图1.55　测试Animation控件

（3）在Default.aspx页面中，选中【CascadingDropDown】复选框，单击【Run Tests】按钮就可以测试CascadingDropDown控件，测试结果如图1.56所示。

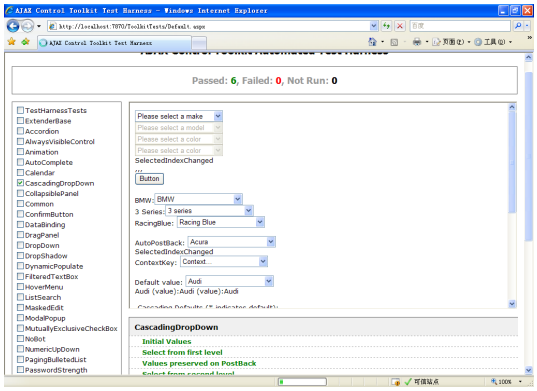

图1.56　测试CascadingDropDown控件