第 16 章　ASP.NET 3.5 和 AJAX

现今，在 Web 开发领域最流行的就属 AJAX，AJAX 能够提升用户体验，更加方便的与 Web 应用程序进行交互。在传统的 Web 开发中，对页面进行操作往往需要进行回发，从而导致页面刷新，而使用 AJAX 就无需产生回发从而实现无刷新效果。

16.1　AJAX 基础

在 C/S 应用程序的开发过程中，很容易做到无“刷新”样式控制，因为 C/S 应用程序往往是安装在本地的，所以 C/S 应用程序能够维持客户端状态，对于状态的改变能够及时捕捉。相比之下，Web 应用属于一种无状态的应用程序，在 Web 应用程序操作过程中，需要通过 POST 等方法进行页面参数传递，这样就不可避免的产生页面的刷新。

16.1.1　什么是 AJAX

在传统的 Web 开发过程中，浏览者浏览一个 Web 页面，并进行相应的页面填写时，就需要使用表单向服务器提交信息。当用户提交表单时，就不可避免的会向服务器发送一个请求，服务器接受该请求后并执行相应的操作后将生成一个页面返回给浏览者。

然而，在服务器处理表单并返回新的页面的同时，浏览者第一次浏览时的页面（这里可以当作是旧的页面）和服务器处理表单后返回的页面在形式上基本相同，当大量的用户进行表单提交操作时，无疑是增加了网络的带宽，因为处理前和处理后的页面基本相同。

在 C/S 应用程序开发中，C/S 应用程序往往安装在本地，这样响应用户事件的时间非常的短，而且 C/S 应用程序可以算的上是有状态的应用程序，能够及时捕捉和相应用户的操作。而在 Web 端，由于每次的交互都需要向服务器发送请求，服务器接受请求和返回请求的过程就依赖于服务器的响应时间，所以给用户造成感觉要比在本地慢的多。

为了解决这一问题，通过在用户浏览器和服务器之间设计一个中间层——即 AJAX 层能就能够解决这一问题，AJAX 改变了传统的 Web 中客户端和服务器的“请求——等待——请求——等待”的模式，通过使用 AJAX 应用向服务器发送和接收需要的数据，从而不会产生页面的刷新。

AJAX 应用通过使用 SOAP 或其他一些基于 XML 的 web service 接口，并在客户端采用 JavaScript 处理来自服务器的响应，从而减少了服务器和浏览器之间的“请求——回发”操作，从而减少了带宽。当服务器和客户端之间的信息通信减少之后，浏览者就会感觉到 Web 应用中的操作就更快了。

AJAX 将一些应用的处理交付给客户端，让服务器端原本应该运行的操作和需要处理的事务分布给客户端，这样服务器端的处理时间也减少了。

相对于传统的 Web 开发，AJAX 提供了更好的用户体验，AJAX 也提供了较好的 Web 应用交互的解决方案，相对于传统的 Web 开发而言，AJAX 技术也减少了网络带宽。Ajax 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中就被引入了，它是一种支持异步请求的技术。简而言之，XmlHttpRequest 使您可以使用 JavaScript 向服务器提出请求并处理响应，而不会影响客户端的信息通信。

传统的 Web 应用和 AJAX 应用模型如图 16-1 所示。

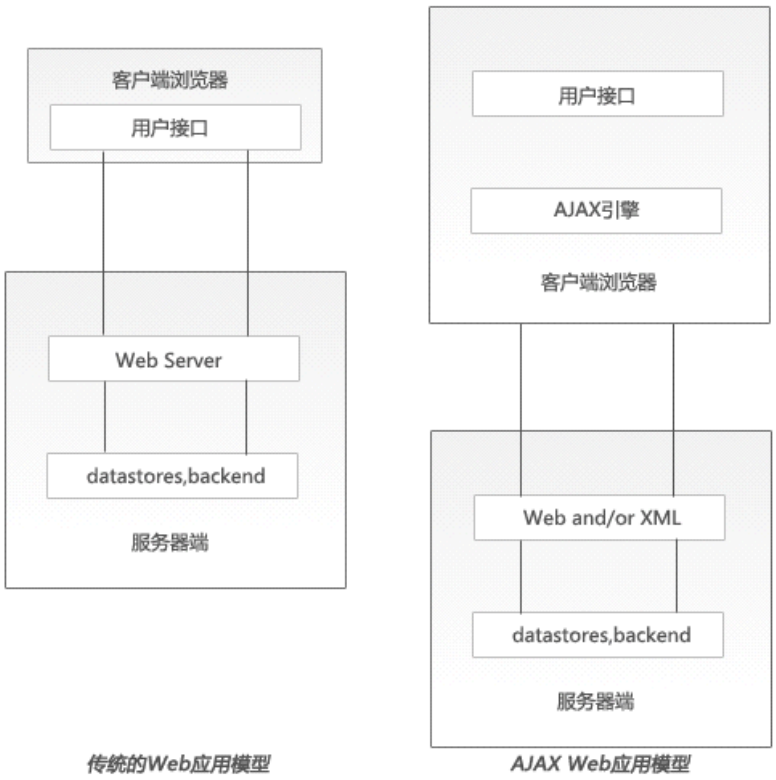

图 16-1　传统 Web 应用和 AJAX Web 应用模型

AJAX Web 应用模型的优点在于，无需进行整个页面的回发就能够进行局部的更新，这样能够使 Web 服务器能够尽快的响应用户的要求。而 AJAX Web 应用无需安装任何插件，也无需在 Web 服务器中安装应用程序，但是 AJAX 需要用户允许 JavaScript 在浏览器上执行，如图用户不允许 JavaScript 在浏览器上执行，则 AJAX 可能无法运行。但是随着 AJAX 的发展和客户端浏览器的发展，先进的所有的浏览器都能够支持 AJAX，包括最新的 IE8、Firefox 4 以及 Opera 等。

AJAX 包含诸多优点，同样也包含缺点。AJAX 无法维持刚刚的“历史”状态，当用户在一个页面进行操作后，AJAX 将破坏浏览器的功能中的“后退”功能，当用户执行了 AJAX 操作之后，当单击浏览器的后退按钮时，则不会返回到 AJAX 操作前的页面形式，因为浏览器仅仅能够记录静态页面的状态，而使用 AJAX 进行页面操作后，并不能改变本身页面的状态，所以单击后退按钮并不能返回操作前的页面状态。

在使用 AJAX 进行 Web 应用开发的过程中，另一个缺点就是容易造成用户体验变差。虽然 AJAX 能够极大的方便用户体验，但是当服务器需求变大时，当用户进行一个操作而 AJAX 无法及时相应时，可能会造成相反的用户体验。

例如用户阅读一个新闻时，当用户进行评论时，页面并没有刷新，但是评论这个操作已经在客户端和浏览器之间发生了，用户可能很难理解为什么页面没有显式也没有刷新，这样容易让用户变得急躁和不安，使得用户可能产生非法操作从而降低用户体验。为了解决这个问题，可以在页面明显的位置提示用户已经操作或提示请等待等操作，让用户知道页面正在运行。

相比于传统的 Web 应用，AJAX 的另一个缺点就是对移动设备的支持不够好。在当今 IPhone 和 GPhone 等智能移动设备逐渐普及的同时，AJAX 并不能很好的支持这些设备，这也需要等待 AJAX 技

术的进一步发展。

16.1.2 ASP.NET AJAX 入门

AJAX 技术看似非常的复杂，其实 AJAX 并不是新技术，AJAX 只是一些老技术的混合体，AJAX 通过将这些技术进行一定的修改、整合和发扬，就形成了 AJAX 技术。这些老技术包括有：

- XHTML：基于 XHTML1.0 规范的 XHTML 技术。
- CSS：基于 CSS2.0 的 CSS 布局的 CSS 编程技术。
- DOM： HTML DOM，XML DOM 等 DON 技术。
- JavaScript：JavaScript 编程技术。
- XML：XML DOM、XSLT、XPath 等 XML 编程技术。

上面的这些技术并不是最新的技术，这些技术已经在现在的开发当中被普遍使用，包括 XHTML、CSS 和 DOM，开发人员能够使用 JavaScript 进行 Web 应用中 Web 编程和客户端状态维护，而通过使用 XML 技术能够进行数据保存和交换。

除了上面的一些老技术，AJAX 还包含另一个技术，这个技术就是 XMLHttpRequest。在 AJAX 中，最重要的就是 XMLHttpRequest 对象，XMLHttpRequest 对象是 JavaScript 对象，正式 XMLHttpRequest 对象实现了 AJAX 可以在服务器和浏览器之间通过 JavaScript 创建一个中间层，从而实现了异步通信。如图 16-2 所示。

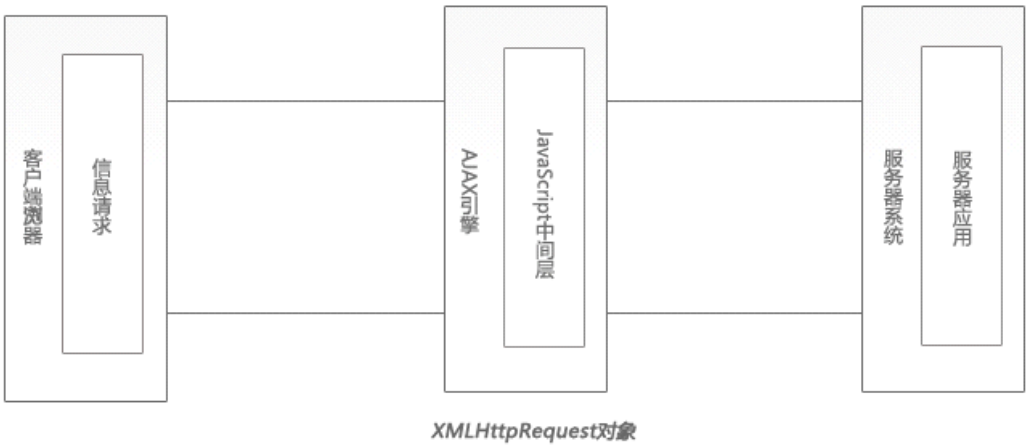

图 16-2　XMLHttpRequest 对象实现过程

AJAX 通过使用 XMLHttpRequest 对象实现异步通信，使用 AJAX 技术后，例如当用户填写一个表单，数据并不是直接从客户端发送到服务器，而是通过客户端发送到一个中间层，这个中间层可以被称为 AJAX 引擎。

开发人员无需知道 AJAX 引擎是如何将数据发送到服务器的。当 AJAX 引擎将数据发送到服务器时，服务器同样也不会直接将数据返回给浏览器，而是通过 JavaScript 中间层将数据返回给客户端浏览器。XMLHttpRequest 对象使用 JavaScript 代码可以自行与服务器进行交互。简而言之，AJAX 技术是通过使用 XHTML、CSS、DOM 等实现的，具体实现如下所示。

- 使用 XHTML+CSS 进行页面表示表示。
- 使用 DOM 进行动态显示和交互。
- 使用 XML 和 XSLT 进行数据交换。

- 使用 XMLHttpRequest 进行异步数据查询、检索。
- 使用 JavaScript 进行页面绑定。

16.1.3 ASP.NET 2.0 AJAX

在 ASP.NET 3.5 之前，ASP.NET 并不是原生的支持 AJAX 应用，所以在 ASP.NET 3.5 之前使用 ASP.NET AJAX 并不是一件容易的事情。同时由于国内服务器和主机的限制，导致很多应用无法直接基于 ASP.NET 3.5 而进行创建和开发。

ASP.NET 2.0 是现在国内最为成熟的.NET 技术平台，在 ASP.NET 2.0 中同样可以使用 AJAX 技术实现无页面刷新效果。在这之前，首先需要下载一个 ASP.NET 2.0 AJAX 安装程序，通过此安装程序能够在应用程序中安装 AJAX 环境，包括 AJAX 技术以及 AJAX 控件。

技巧：ASP.NET 2.0 AJAX 安装程序可以通过访问 ASP.NET AJAX 中文站点进行下载，（http://www.ajaxasp.net.cn/Download/Files/ASPAJAXExtSetup.msi）

下载 ASP.NET 2.0 AJAX 安装程序后，双击安装程序即可安装，如图 16-3 所示。安装完成后会在 C:\Program Files\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions 目录下生成文件 System.Web.Extensions.dll、System.Web.Extensions.Design.dll 以及 AJAXExtensionsToolbox.dll 等文件。这些文件都是在 ASP.NET 2.0 中使用 AJAX 的必要文件，开发人员能够将这些文件复制到 Web 应用的根目录下再添加引用即可，如图 16-4 所示。

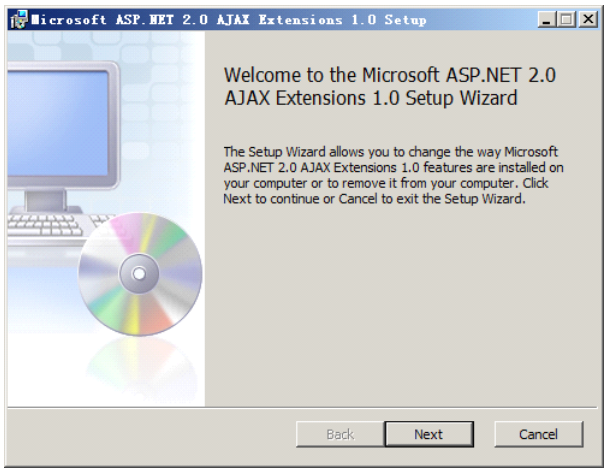

图 16-3 安装 ASP.NET 2.0 AJAX

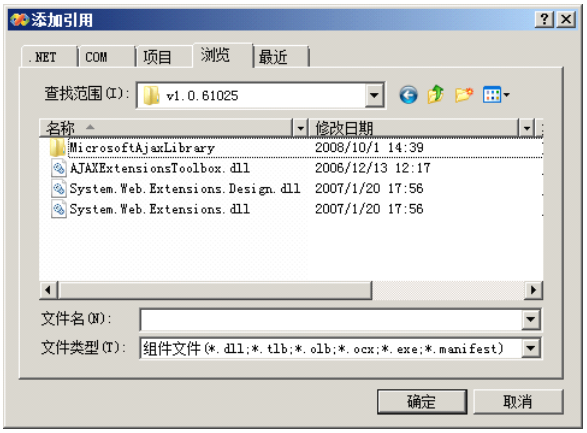

图 16-4 添加 ASP.NET 2.0 AJAX 引用

将这三个 DLL 文件添加引用到应用程序后，Web.config 文件将会更改，示例代码如下所示。

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

添加引用后，可以通过添加工具栏将 AJAX 控件安装到默认工具栏中。右击工具栏空白区域，在下拉菜单中单击【选择项】选项，则会弹出【选择工具箱项】窗口，如图 16-5 所示。单击【浏览】选项，可以选择相应的 DLL 文件以添加工具箱，如图 16-6 所示。

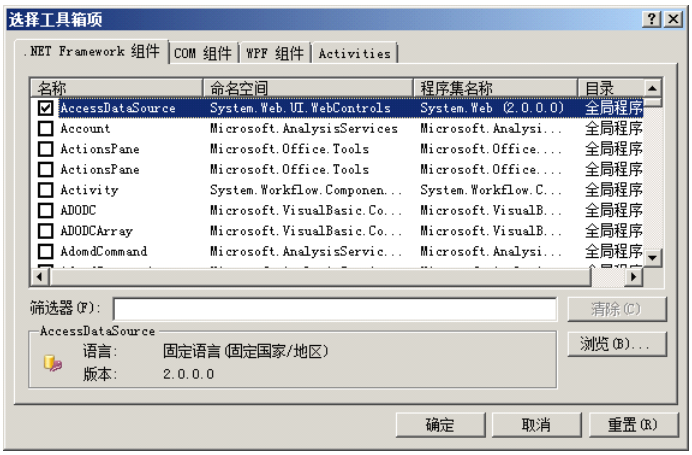

图 16-5　选择工具箱

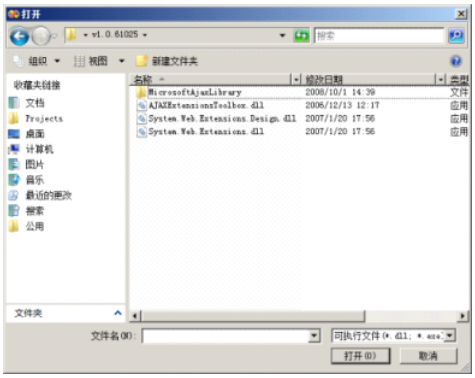

图 16-6　选择 DLL 文件

选择完成后，单击【确定】按钮就会发现在工具栏中已经包含了 AJAX 控件了。引用和工具栏添加完毕后，则需要修改 Web.config 文件从而实现 ASP.NET 2.0 对 AJAX 的支持。

注意：Web.config 文件已经保存在 ASP.NET 2.0 AJAX 应用程序下载安装包中，读者从上述连接或在官方网站下载 ASP.NET 2.0 AJAX 安装包后，其中就包含了 Web.config 文件。

16.1.4　ASP.NET 3.5 AJAX

在 ASP.NET 2.0 中，AJAX 需要下载和安装，开发人员还需要将相应的 DLL 文件分类存放并配置 Web.config 文件才能够实现 AJAX 功能。而在 ASP.NET 3.5 中，AJAX 已经成为.NET 框架的原生功能。创建 ASP.NET 3.5 Web 应用程序就能够直接使用 AJAX 功能，如图 16-7 所示。

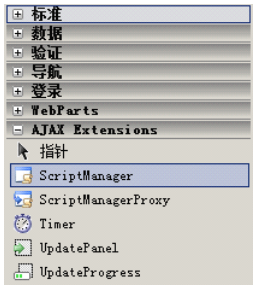

图 16-7　ASP.NET 3.5 AJAX

在 ASP.NET 3.5 中，可以直接拖动 AJAX 控件进行 AJAX 开发。AJAX 能够同普通控件一同使用，从而实现 ASP.NET 3.5 AJAX 中页面无刷新功能。在 ASP.NET 3.5 中，Web.config 文件已经被更改，并且声明了 AJAX 功能，示例代码如下所示。

```
<pages>
    <controls>
        <add tagPrefix="asp" namespace="System.Web.UI"
        assembly="System.Web.Extensions,
        Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
        <add
        tagPrefix="asp"
        namespace="System.Web.UI.WebControls"
        assembly="System.Web.Extensions,
```

```
            Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
        </controls>
    </pages>
```

在 ASP.NET 3.5 中，如果需要在 Internet 信息服务 7.0 中运行 ASP.NET AJAX 应用，则需要配置 System.webServer 配置节，示例代码如下所示。

```
<system.webServer>
    <validation validateIntegratedModeConfiguration="false"/>
    <modules>
        <remove name="ScriptModule"/>
        <add
        name="ScriptModule"
        preCondition="managedHandler"
        type="System.Web.Handlers.ScriptModule,
        System.Web.Extensions,
        Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    </modules>
    <handlers>
        <remove name="WebServiceHandlerFactory-Integrated"/>
        <remove name="ScriptHandlerFactory"/>
        <remove name="ScriptHandlerFactoryAppServices"/>
        <remove name="ScriptResource"/>
        <add name="ScriptHandlerFactory" verb="*" path="*.asmx" preCondition="integratedMode"
        type="System.Web.Script.Services.ScriptHandlerFactory,
        System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
        PublicKeyToken=31BF3856AD364E35"/>
        <add
            name="ScriptHandlerFactoryAppServices"
            verb="*" path="*_AppService.axd" preCondition="integratedMode"
            type="System.Web.Script.Services.ScriptHandlerFactory,
            System.Web.Extensions,
            Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
        <add name="ScriptResource"
            preCondition="integratedMode"
            verb="GET,HEAD" path="ScriptResource.axd"
            type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
            Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    </handlers>
</system.webServer>
```

注意：如果在 Internet 信息服务 7.0 下运行 ASP.NET AJAX 则必须使用 system.webServer 配置节。对早期版本的 IIS 来说则不需要配置此节。

16.1.5 AJAX 简单示例

虽然 AJAX 的原理听上去非常的复杂，但是 AJAX 的使用却是非常方便的。ASP.NET 3.5 提供了 AJAX 控件以便开发人员快速的进行 AJAX 应用程序开发。在进行 AJAX 页面开发时，首先需要使用脚本管理控件（ScriptManger），示例代码如下所示。

```
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
```

开发人员无需对 ScriptManger 控件进行配置，只需保证 ScriptManger 控件在 UpdatePanel 控件之前即可。使用了 ScriptManger 控件之后，可以使用 UpdatePanel 用来确定需要进行局部更新的控件，示例代码如下所示。

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:TextBox ID="TextBox1" runat="server" AutoPostBack="True"
            ontextchanged="TextBox1_TextChanged"></asp:TextBox>
         <asp:Button ID="Button1" runat="server" onclick="Button1_Click1"
            Text="Button" />
        <br />
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </ContentTemplate>
</asp:UpdatePanel>
```

上述代码使用了 UpdatePanel 控件将服务器控件进行绑定，当浏览者操作 UpdatePanel 控件中的控件实现某种特定的功能时，页面只会针对 UpdatePanel 控件之间的控件进行刷新操作，而不会进行这个页面的刷新。为控件进行事件操作编写代码，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Label2.Text = DateTime.Now.ToString();                    //获取当前时间
}
protected void Button1_Click1(object sender, EventArgs e)
{
    TextBox1.Text = DateTime.Now.ToString();                  //获取当前时间
}
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    Label1.Text = TextBox1.Text.Length.ToString();            //统计 TextBox1 中的字符串个数
}
```

当用户单击按钮控件时，TextBox 控件将会获得当前时间并呈现到 TextBox 控件中，当 TextBox 控件失去焦点时，则能够统计 TextBox 控件中字符串的个数。在传统的 Web 开发中，无论是单击按钮还是使用 AutoPostBack 属性都需要向服务器发送请求，服务器接收请求后执行请求，请求执行完毕再生成一个 Web 页面呈现在客户端。

当 Web 页面再次呈现到客户端时，用户能够很明显的感觉到页面被刷新。使用 UpdatePanel 控件后，页面只会针对 UpdatePanel 控件内的内容进行更新，而不会影响 UpdatePanel 控件外的控件，运行后如图 16-8 和图 16-9 所示。

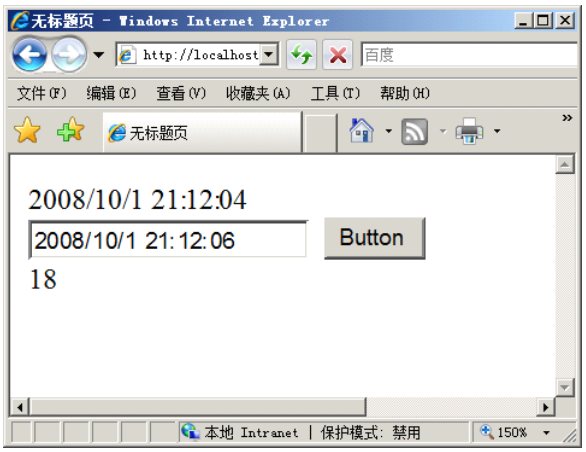

图 16-8　单击按钮获取时间

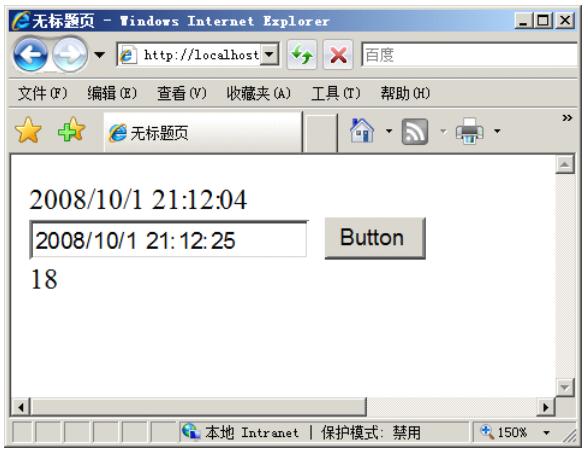

图 16-9　再次获取时间

当应用程序运行之后，单击按钮控件能够获取当前时间，当再次单击按钮控件之后，当前时间同样能够被获取并呈现在 TextBox 中，但是页面并没有再次被更新。在执行过程中，第一次获取的时间为 2008/10/1 21:12:04，当再次获取时间时，时间还是 2008/10/1 21:12:04，这说明 UpdatePanel 控件外的页面元素都没有再更新。

16.2　ASP.NET 3.5AJAX 控件

在 ASP.NET 3.5 当中，系统提供了 AJAX 控件以便开发人员能够在 ASP.NET 3.5 中进行 AJAX 应用程序开发，通过使用 AJAX 控件能够减少大量的代码开发，为开发人员提供了 AJAX 应用程序搭建和应用的绝佳环境。

16.2.1　脚本管理控件（ScriptManger）

脚本管理控件（ScriptManger）是 ASP.NET AJAX 中非常重要的控件，通过使用 ScriptManger 能够进行整个页面的局部更新的管理。ScriptManger 用来处理页面上局部更新，同时生成相关的代理脚本以便能够通过 JavaScript 访问 Web Service。

ScriptManger 只能在页面中被使用一次，这也就是说每个页面只能使用一个 ScriptManger 控件，ScriptManger 控件用来进行该页面的全局管理。创建一个 ScriptManger 控件后系统自动生成 HTML 代码，示例代码如下所示。

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

ScriptManger 控件用户整个页面的局部更新管理，ScriptManger 控件的常用属性如下所示：

- AllowCustomErrorRedirect：指明在异步回发过程中是否进行自定义错误重定向。
- AsyncPostBackTimeout：指定异步回发的超时事件，默认为 90 秒。
- EnablePageMethods：是否启用页面方法，默认值为 false。
- EnablePartialRendering：在支持的浏览器上为 UpdatePanel 控件启用异步回发。默认值为 True。
- LoadScriptsBeforeUI：指定在浏览器中呈现 UI 之前是否应加载脚本引用。
- ScriptMode：指定要在多个类型时可加载的脚本类型，默认为 Auto。

在 AJAX 应用中，ScriptManger 控件基本不需要配置就能够使用。因为 ScriptManger 控件通常需要同其他 AJAX 控件搭配使用，在 AJAX 应用程序中，ScriptManger 控件就相当于一个总指挥官，这个总指挥官只是进行指挥，而不进行实际的操作。

1．使用 ScriptManger

ScriptManger 控件在页面中相当于指挥的功能，如果需要使用 AJAX 的其他控件，就必须使用 ScriptManger 控件并且页面中只能包含一个 ScriptManger 控件。示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label1" runat="server" Text="这是一串字符" Font-Size="12px"></asp:Label>
                <br /><br />
                <asp:TextBox ID="TextBox1" runat="server" AutoPostBack="True"
                ontextchanged="TextBox1_TextChanged"></asp:TextBox>
                字符的大小(px)
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

上述代码创建了一个ScriptManger控件和一个UpdatePanel控件用于AJAX应用开发。在UpdatePanel控件中，包含一个 Label 标签控件和一个 TextBox 文本框控件，当文本框控件的内容被更改时，则会触发 TextBox1_TextChanged 事件。TextChanged 事件相应的 CS 代码如下所示。

```
    protected void TextBox1_TextChanged(object sender, EventArgs e)
    {
        try
        {
            Label1.Font.Size = FontUnit.Point(Convert.ToInt32(TextBox1.Text));     //改变字体
        }
        catch
        {
            Response.Write("错误");                                                 //抛出异常
        }
    }
```

上述代码通过文本框中的输入进行字体控制，当输入一个数字字符串并失去焦点时，则会触发改事件并执行相应的代码，运行后如图 16-10 和图 16-11 所示。

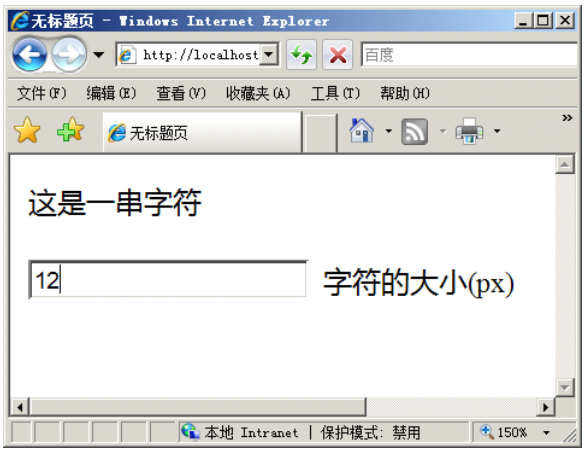

图 16-10　输入字符大小

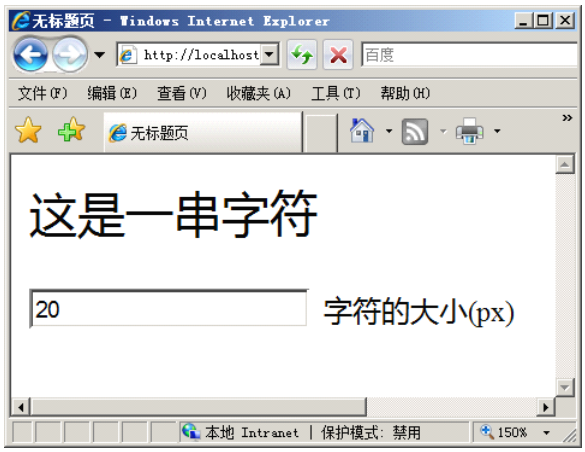

图 16-11　调整字体大小

2. 捕获异常

当页面回传发生异常时，则会触发 AsyncPostBackError 事件，示例代码如下所示。

```
protected void ScriptManager1_AsyncPostBackError(object sender, AsyncPostBackErrorEventArgs e)
{
    ScriptManager1.AsyncPostBackErrorMessage = "回传发生异常:" + e.Exception.Message;
}
```

AsyncPostBackError 事件的触发依赖于 AllowCustomErrorsRedirct 属性、AsyncPostBackErrorMessage 属性和 Web.config 中的<customErrors>配置节。其中，AllowCustomErrorsRedirct 属性指明在异步回发过程中是否进行自定义错误重定向，而 AsyncPostBackErrorMessage 属性指明当服务器上发生未处理异常时要发送到客户端的错误消息。示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    throw new ArgumentException();                                    //抛出异常
}
```

上述代码当单击按钮控件时，则会抛出一个异常，ScriptManger 控件能够捕获异常并输出异常，运行代码后系统会提示异常“回传发生异常:值不在预期范围内”。

16.2.2　脚本管理控件（ScriptMangerProxy）

ScriptManger 控件作为整个页面的管理者，ScriptManger 控件能够提供强大的功能以致开发人员无需关心 ScriptManger 控件是如何实现 AJAX 功能的，但是一个页面只能使用一个 ScriptManger 控件，如果在一个页面中使用多个 ScriptManger 控件则会出现异常。

在 Web 应用的开发过程中，常常需要使用到母版页。在前面的章节中提到，母版页和内容窗体一同组合成为一个新页面呈现在客户端浏览器，那么如果在母版页中使用了 ScriptManger 控件，而在内容窗体中也使用 ScriptManger 控件的话，整合在一起的页面就会出现错误。为了解决这个问题，就可以使用另一个脚本管理控件，ScriptMangerProxy 控件。ScriptMangerProxy 控件和 ScriptManger 控件十分相似，首先创建母版页，示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div style="width:300px; float:left; background:#f0f0f0; height:300px">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
```

```
        </asp:ContentPlaceHolder>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <asp:ScriptManager ID="ScriptManager1" runat="server">
                </asp:ScriptManager>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="获取当前时间" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    <div style="width:300px; float:left; background:gray;color:White;height:300px">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
```

上述代码创建了母版页，并且母版页中使用了 ScriptMangerProxy 控件为母版页中的控件进行 AJAX 应用支持，母版页中按钮控件的事件代码如下所示。

```
        protected void Button1_Click(object sender, EventArgs e)
        {
            TextBox1.Text = "母版页中的时间为" + DateTime.Now.ToString();                    //获取母版页时间
        }
```

在内容窗体中可以使用母版页进行样式控制和布局，内容窗体页面代码如下所示。

```
<%@ Page Language="C#"
MasterPageFile="~/Site1.Master"
AutoEventWireup="true"
CodeBehind="MyScriptMangerProxy.aspx.cs" Inherits="_16_2.MyScriptMangerProxy" Title="无标题页" %>
    <asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    </asp:Content>
    <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    </asp:Content>
    <asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">
        <asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
        </asp:ScriptManagerProxy>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="内容窗体时间" />
    </ContentTemplate>
    </asp:UpdatePanel>
        <br />
    </asp:Content>
```

上述代码为内容窗体代码，在内容窗体中，使用了 Site1.Master 母版页作为样式控制，并且通过使用 ScriptMangerProxy 控件进行内容窗体 AJAX 应用的支持。运行后如图 16-12 所示。

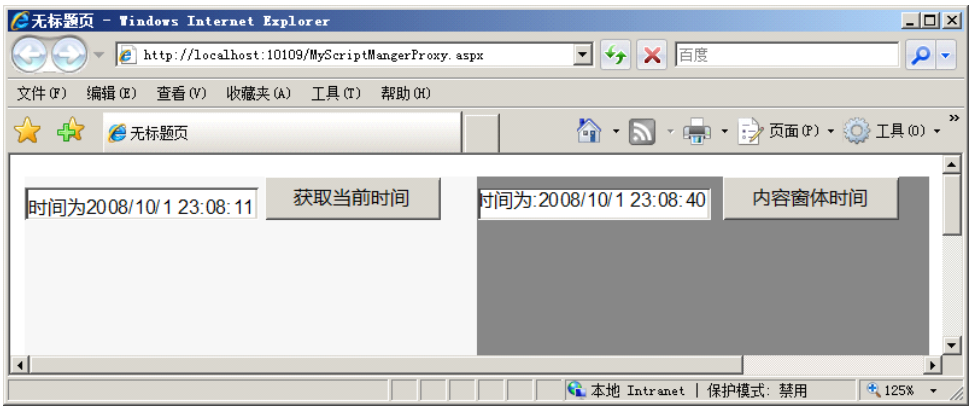

图 16-12　ScriptMangerProxy 控件

ScriptMangerProxy 控件与 ScriptManger 控件非常的相似，但是 ScriptManger 控件只允许在一个页面中使用一次。当 Web 应用需要使用母版页进行样式控制时，母版页和内容页都需要进行局部更新时 ScriptManger 控件就不能完成需求，使用 ScriptMangerProxy 控件就能够在母版页和内容页中都实现 AJAX 应用。

16.2.3　时间控件（Timer）

在 C/S 应用程序开发中，Timer 控件是最常用的控件，使用 Timer 控件能够进行时间控制。Timer 控件被广泛的应用在 Windows WinForm 应用程序开发中，Timer 控件能够在一定的时间内间隔的触发某个事件，例如每隔 5 秒就执行某个事件。

但是在 Web 应用中，由于 Web 应用是无状态的，开发人员很难通过编程方法实现 Timer 控件，虽然 Timer 控件还是可以通过 JavaScript 实现，但是这样也是以复杂的编程的大量的性能要求为代价的，这样就造成了 Timer 控件的使用困难。在 ASP.NET AJAX 中，AJAX 提供了一个 Timer 控件，用于执行局部更新，使用 Timer 控件能够控制应用程序在一段时间内进行事件刷新。Timer 控件初始代码如下所示。

```
<asp:Timer ID="Timer1" runat="server">
</asp:Timer>
```

开发人员能够配置 Timer 控件的属性进行相应事件的触发，Timer 的属性如下所示。

- Enabled：是否启用 Tick 时间引发。
- Interval：设置 Tick 事件之间的连续时间，单位为毫秒。

通过配置 Timer 控件的 Interval 属性，能够指定 Time 控件在一定时间内进行事件刷新操作，示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
                <asp:Timer ID="Timer1" runat="server" Interval="1000" ontick="Timer1_Tick">
                </asp:Timer>
```

```
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

上述代码使用了一个 ScriptManage 控件进行页面全局管理，ScriptManage 控件是必须的。两外，在页面中使用了 UpdatePanel 控件，该控件用于控制页面的局部更新，而不会引发整个页面刷新。在 UpdatePanel 控件中，包括一个 Label 控件和一个 Timer 控件，Label 控件用于显示时间，Timer 控件用于每 1000 毫秒执行一次 Timer1_Tick 事件，Label1 和 Timer 控件的事件代码如下所示。

```
        protected void Page_Load(object sender, EventArgs e)                    //页面打开时执行
        {
            Label1.Text = DateTime.Now.ToString();                               //获取当前时间
        }
        protected void Timer1_Tick(object sender, EventArgs e)                   //Timer 控件计数
        {
            Label1.Text = DateTime.Now.ToString();                               //遍历获取时间
        }
```

上述代码在页面被呈现时，将当前时间传递并呈现到 Label 控件中，Timer 控件用于每隔一秒进行一次刷新并将当前时间传递并呈现在 Label 控件中，这样就形成了一个可以计数的时间，如图 16-13 和图 16-14 所示。

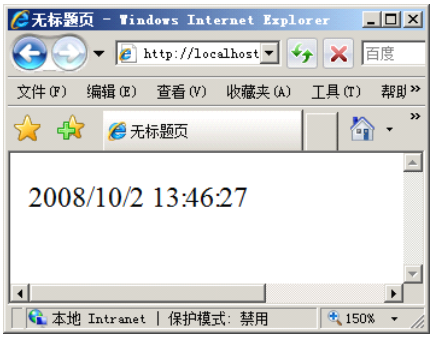

图 16-13　初始页面

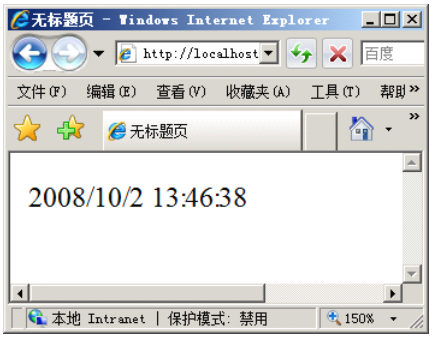

图 16-14　刷新操作

Timer 控件能够通过简单的方法让开发人员无需通过复杂的 JavaScript 实现 Timer 控制。但是从另一方面来讲，Timer 控件会占用大量的服务器资源，如果不停的进行客户端服务器的信息通信操作，很容易造成服务器当机。

16.2.4　更新区域控件（UpdatePanel）

更新区域控件（UpdatePanel）在 ASP.NET AJAX 是最常用的控件，在上面几节控件的讲解中，已经使用到 UpdatePanel 控件，这已经说明 UpdatePanel 控件是非常重要的 AJAX 控件。

UpdatePanel 控件使用的方法同 Panel 控件类似，只需要在 UpdatePanel 控件中放入需要刷新的控件就能够实现局部刷新。使用 UpdatePanel 控件，整个页面中只有 UpdatePanel 控件中的服务器控件或事件会进行刷新操作，而页面的其他地方都不会被刷新。UpdatePanel 控件 HTML 代码如下所示。

```
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        </asp:UpdatePanel>
```

UpdatePanel 控件可以用来创建局部更新，开发人员无需编写任何客户端脚本，直接使用 UpdatePanel

控件就能够进行局部更新，UpdatePanel 控件的属性如下所示。

- RenderMode：该属性指明 UpdatePanel 控件内呈现的标记应为 或。
- ChildrenAsTriggers：该属性指明来在 UpdatePanel 控件的子控件的回发是否导致 UpdatePanel 控件的更新，其默认值为 True。
- EnableViewState:：指明控件是否自动保存其往返过程。
- Triggers：指明可以导致 UpdatePanel 控件更新的触发器的集合。
- UpdateMode：指明 UpdatePanel 控件回发的属性，是在每次进行事件时进行更新还是使用 UpdatePanel 控件的 Update 方法再进行更新。
- Visible：UpdatePanel 控件的可见性。

UpdatePanel 控件要进行动态更新，必须依赖于 ScriptManage 控件。当 ScriptManage 控件允许局部更新时，它会以异步的方式发送到服务器，服务器接受请求后，执行操作并通过 DOM 对象来替换局部代码，其原理如图 16-15 所示。

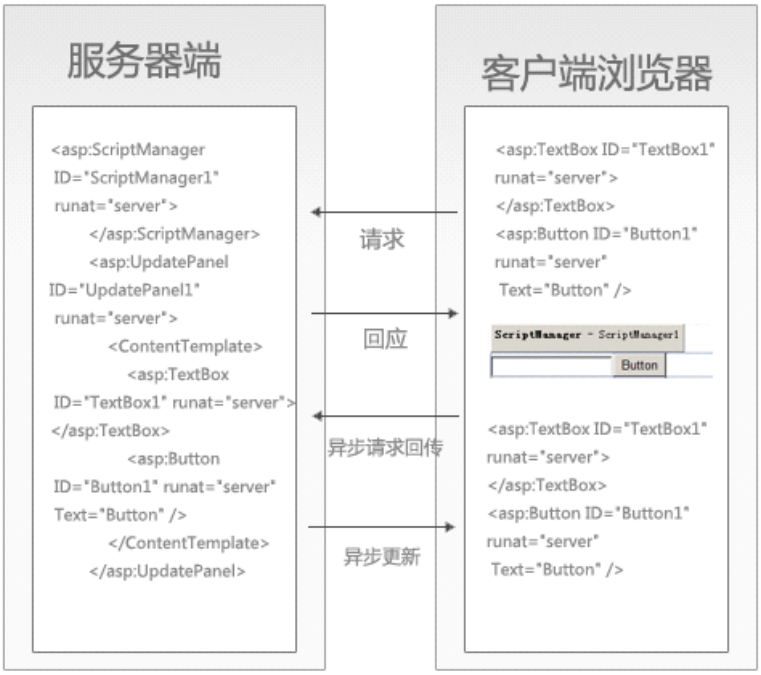

图 16-15　UpdatePanel 控件异步请求示意图

UpdatePanel 控件包括 ContentTemplate 标签。在 UpdatePanel 控件的 ContentTemplate 标签中，开发人员能够放置任何 ASP.NET 控件这些控件到 ContentTemplate 标签中，这些控件就能够实现页面无刷新的更新操作，示例代码如下所示。

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:Button ID="Button1" runat="server" Text="Button" />
    </ContentTemplate>
</asp:UpdatePanel>
```

上述代码在 ContentTemplate 标签加入了 TextBox1 控件和 Button1 控件，当这两个控件产生回发事件，并不会对页面中的其他元素进行更新，只会对 UpdatePanel 控件中的内容进行更新。UpdatePanel 控

件还包括 Triggers 标签，Triggers 标签包括两个属性，这两个属性分别为 AsyncPostBackTrigger 和 PostBackTrigger。AsyncPostBackTrigger 用来指定某个服务器端控件，以及将其触发的服务器事件作为 UpdatePanel 异步更新的一种触发器，AsyncPostBackTrigger 属性需要配置控件的 ID 和控件产生的事件名，示例代码如下所示。

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="Button" />
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="TextBox1" EventName="TextChanged" />
    </Triggers>
</asp:UpdatePanel>
```

而 PostBackTrigger 用来指定在 UpdatePanel 中的某个控件，并指定其控件产生的事件将使用传统的回发方式进行回发。当使用 PostBackTrigger 标签进行控件描述时，当该控件产生了一个事件，页面并不会异步更新，而会使用传统的方法进行页面刷新，示例代码如下所示。

```
<asp:PostBackTrigger ControlID="TextBox1" />
```

UpdatePanel 控件在 ASP.NET AJAX 中是非常重要的，UpdatePanel 控件用于进行局部更新，当 UpdatePanel 控件中的服务器控件产生事件并需要动态更新时，服务器端返回请求只会更新 UpdatePanel 控件中的事件而不会影响到其他的事件。

16.2.5 更新进度控件（UpdateProgress）

使用 ASP.NET AJAX 常常会给用户造成疑惑。例如当用户进行评论或留言时，页面并没有刷新，而是进行了局部刷新，这个时候用户很可能不清楚到底发生了什么，以至于用户很有可能会产生重复操作，甚至会产生非法操作。

更新进度控件（UpdateProgress）就用于解决这个问题，当服务器端与客户端进行异步通信时，需要使用 UpdateProgress 控件告诉用户现在正在执行中。例如当用户进行评论时，当用户单击按钮提交表单，系统应该提示“正在提交中，请稍后”，这样就提供了便利从而让用户知道应用程序正在运行中。这种方法不仅能够让用户操作更少的出现错误，也能够提升用户体验的友好度。UpdateProgress 控件的 HTML 代码如下所示：

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
    <ProgressTemplate>
        正在操作中，请稍后 ...<br />
    </ProgressTemplate>
</asp:UpdateProgress>
```

上述代码定义了一个 UpdateProgress 控件，并通过使用 ProgressTemplate 标记进行等待中的样式控制。ProgressTemplate 标记用于标记等待中的样式。当用户单击按钮进行相应的操作后，如果服务器和客户端之间需要时间等待，则 ProgressTemplate 标记就会呈现在用户面前，以提示用户应用程序正在运行。完整的 UpdateProgress 控件和 UpdatePanel 控件代码如下所示。

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
    <asp:UpdateProgress ID="UpdateProgress1" runat="server">
```

```
            <ProgressTemplate>
                正在操作中，请稍后 ...<br />
                </ProgressTemplate>
            </asp:UpdateProgress>
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
                <asp:Button ID="Button1" runat="server" Text="Button"
                onclick="Button1_Click" />
        </ContentTemplate>
    </asp:UpdatePanel>
```

上述代码使用了 UpdateProgress 控件用户进度更新提示，同时创建了一个 Label 控件和一个 Button 控件，当用户单击 Button 控件时则会提示用户正在更新，Button 更新事件代码如下所示。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        System.Threading.Thread.Sleep(3000);                    //挂起 3 秒
        Label1.Text = DateTime.Now.ToString();                  //获取时间
    }
```

上述代码使用了 System.Threading.Thread.Sleep 方法指定系统线程挂起的时间，这里设置 3000 毫秒，这也就是说当用户进行操作后，在这 3 秒的时间内会呈现“正在操作中，请稍后…”几个字样，当 3000 毫秒过后，就会执行下面的方法，运行后如图 16-16 和图 16-17 所示。

图 16-16　正在操作中

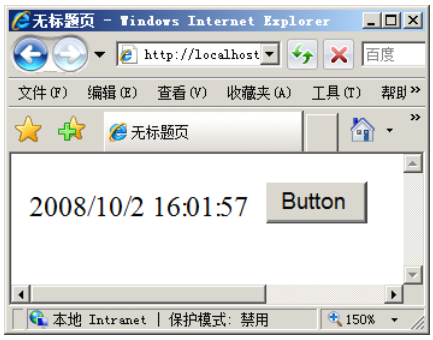

图 16-17　操作完毕后

在用户单击后，如果服务器和客户端之间的通信需要较长时间的更新，则等待提示语会出现正在操作中。如果服务器和客户端之间交互的时间很短，基本上看不到 UpdateProgress 控件的显示。虽然如此，UpdateProgress 控件在大量的数据访问和数据操作中能够提高用户友好度，并避免错误的发生。

16.3　AJAX 编程

通过编程的方法实现 AJAX 高级功能，能够补充现有的 AJAX 功能。例如在执行局部更新时，如果出现了异常，则需要通过编程的方法实现错误信息提交，这样不仅能够提升用户体验的友好度，也能够提升应用程序的健壮性。

16.3.1　自定义异常处理

在 AJAX 应用程序开发和使用中，用户很容易输入错误信息的信息造成异常。例如在 UpdatePanel 控件中执行应用程序操作时，如果发生了错误，则会弹出一个对话框，这个对话框对用户来说非常晦涩

并且极不友好，这里就需要自定义异常处理。在页面中，首先需要创建一个 ScriptManage 控件和 UpdatePanel 控件，示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

上述代码创建了一个 ScriptManage 控件和 UpdatePanel 控件，在 UpdatePanel 控件中，开发人员可以拖放用户控件，以便进行页面局部更新，示例代码如下所示。

```
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Label ID="Label1" runat="server" Text="计算器"></asp:Label>
                <br />
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                除以<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                等于<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="计算" />
        </ContentTemplate>
    </asp:UpdatePanel>
```

上述代码编写了一个简单的计算器，用户能够通过输入相应的数字进行运算，CS 页面代码如下所示。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            int a, b;                                          //创建整型变量
            float c;                                           //创建浮点型变量
            a = Convert.ToInt32(TextBox1.Text);                //获取数值
            b = Convert.ToInt32(TextBox2.Text);                //获取数值
            c = a / b;                                         //进行计算
            TextBox3.Text = c.ToString();                      //结果呈现
        }
        catch(Exception ee)
        {
            ee.Data["error"] = "自定义错误";                    //编写自定义错误
            throw ee;                                          //抛出自定义错误
        }
    }
```

上述代码描述了当用户单击按钮后，页面执行转换，即将文本框中的文本进行转换。转换完成后再相除，相除后输出到 TextBox3 中。但是这里会有一个问题，这个问题就是如果用户输入的不是数字，或者输入数字的除数是 0，都会导致异常。开发人员能够自定义异常并抛出异常，示例代码如下所示。

```
            ee.Data["error"] = "自定义错误";                    //编写自定义错误
            throw ee;                                          //抛出自定义错误
```

当重新抛出异常后，ScriptManage 控件能够捕捉该异常。为了让 ScriptManage 控件捕捉异常，可以编写 ScriptManage 控件的 AsyncPostBackError 事件，示例代码如下所示。

```
protected void ScriptManager1_AsyncPostBackError(object sender, AsyncPostBackErrorEventArgs e)
{
    if (e.Exception.Data["error"] != null)                          //判断自定义错误
    {
        ScriptManager1.AsyncPostBackErrorMessage =
        "发生了一个错误" + e.Exception.Data["error"].ToString();      //呈现自定义错误
    }
    else
    {
        ScriptManager1.AsyncPostBackErrorMessage = "发生了一个错误";  //默认系统错误
    }
}
```

上述代码通过 ScriptManage 控件的 AsyncPostBackError 事件捕获一个异常，如果抛出的异常被 ScriptManage 控件捕获，则会通过对话框的形式呈现在客户端浏览器，如图 16-18 所示。

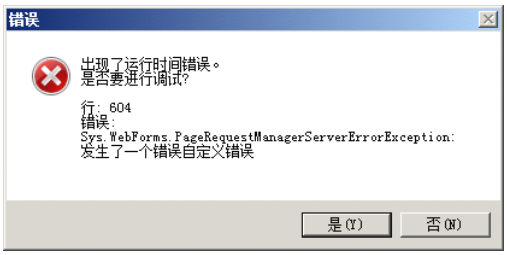

图 16-18　自定义异常处理

16.3.2　使用母版页的 UpdatePanel

在 AJAX 应用程序的开发中，常常需要制作大量的相同页面，这些相同的页面可以使用母版页进行样式控制，而内容页只需要进行控件布局即可。如果在母版页中需要完成和实现 AJAX 应用，则可以在母版页中使用 UpdatePanel 控件进行局部更新。母版页示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div style="width:300px; float:left; background:#f0f0f0; height:300px">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <asp:ScriptManager ID="ScriptManager1" runat="server">
                </asp:ScriptManager>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="获取当前时间" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    <div style="width:300px; float:left; background:gray;color:White;height:300px">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">
        </asp:ContentPlaceHolder>
```

```
    </div>
    </form>
</body>
```

上述代码在母版页中声明了一个 ScriptManage 控件和一个 UpdatePanel 控件，在母版页中可以通过向 UpdatePanel 控件拖动服务器控件以完成页面局部刷新。在编写内容窗体时，可以无需再创建 ScriptManage 控件也同样能够进行页面拒不更新，内容窗体示例代码如下所示。

```
<%@ Page Language="C#"
MasterPageFile="~/Site1.Master"
AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="_16_4.WebForm1" Title="无标题页" %>
    <asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
        </asp:Content>
        <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    </asp:Content>
    <asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">
    <asp:UpdatePanel ID="UpdatePanel3" runat="server">
        <ContentTemplate>
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="get time" />
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

在内容窗体中，内容窗体使用了母版页，而母版页中已经包含了 ScriptManage 控件，所以当母版页和内容窗体整合在一起呈现一个新页面时，新的页面已经包含了 ScriptManage 控件。所以在对内容窗体中 UpdatePanel 控件中的控件进行更新时，就算内容窗体中没有 ScriptManage 控件，也能够进行局部更新。

注意：如果在内容窗体中使用 ScriptManage 控件，则会抛出异常，因为一个页面只允许使用一个 ScriptManage 控件，当需要在内容窗体中也使用 ScriptManage 控件时，可以使用 ScriptManageProxy 控件。

16.3.3 母版页刷新内容窗体

在母版页中使用 ScriptManage 控件能够方便的将整个页面进行 AJAX 全局控制。当 Web 应用中有很多相似页面，又需要执行 AJAX 应用时，可以在母版页中使用 ScriptManage 控件，在内容窗体中使用 UpdatePanel 控件，这样母版页中的 ScriptManage 控件也会整合到内容窗体中并进行整合输出。

同样，母版页也能够刷新内容窗体中的控件信息，在上一节中，母版页使用了 ScriptManage 控件进行全局控制，而内容窗体则通过本身的按钮实现时间的获取。在母版页中，可以创建一个按钮控件以执行内容窗体中文本框控件的局部更新，母版页局部示例代码如下所示。

```
    <asp:UpdatePanel ID="UpdatePanel2" runat="server">
        <ContentTemplate>
            <asp:ScriptManager ID="ScriptManager1" runat="server">
                </asp:ScriptManager>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="获取当前时间" />
                <br />
            <asp:Button ID="Button2" runat="server" Text="刷新子母版的值" />
        </ContentTemplate>
```

```
    </asp:UpdatePanel>
```

上述代码在母版页中增加了一个按钮，通过该按钮控件能够刷新内容窗体中控件的值。但是如果要实现母版页刷新内容窗体的值，首先需要在母版页代码中注册这两个按钮为异步提交按钮，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    ScriptManager1.RegisterAsyncPostBackControl(Button2);                    //注册异步操作
}
```

上述代码通过使用 RegisterAsyncPostBackControl 方法进行异步提交按钮控件的注册，注册完毕后就能够为控件编写相应的事件，示例代码如下所示。

```
protected void Button2_Click(object sender, EventArgs e)
{
    ((UpdatePanel)ContentPlaceHolder2.FindControl("UpdatePanel3")).Update();    //查找相应的控件
    TextBox tex = ((TextBox)ContentPlaceHolder2.FindControl("TextBox2"));       //创建 TextBox
    tex.Text = DateTime.Now.ToString();                                         //更改控件值
}
```

上述代码通过 FindControl 的方法进行控件的寻找和更改，找到目标控件后再进行目标控件的值的更改。在母版页注册异步提交按钮的方法并实现相应事件后，母版页并不能直接的进行内容窗体的更改，在内容窗体的 UpdatePanel 控件中，必须将 UpdateMode 属性更改为 Conditional，这样才能在内容窗体中接受母版页中进行局部更新的事件。运行后如图 16-19 所示。

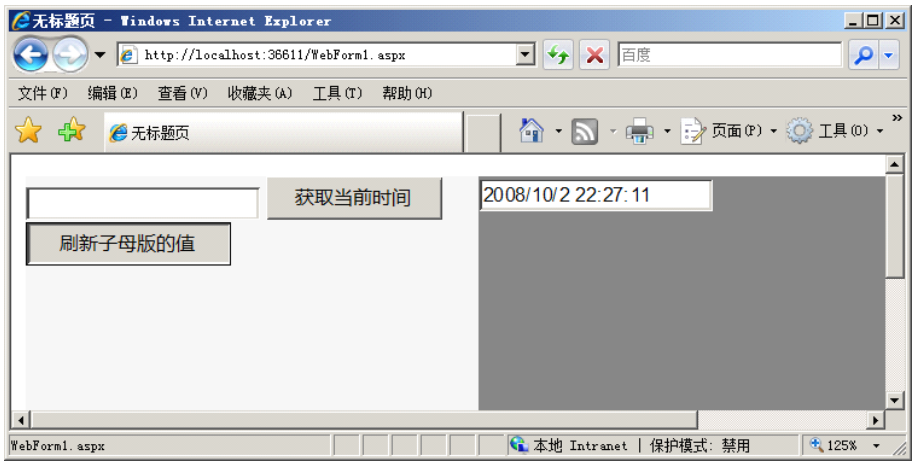

图 16-19　通过母版页进行内容窗体局部刷新

如果 ASP.NET AJAX Web 应用中很多的页面都需要执行相同的操作，而内容窗体同样需要执行这些操作，可以通过在母版页中注册异步传送控件以支持内容窗体中 AJAX 应用的需求，这样只需要在内容窗体中进行控件布局，而无需在内容窗体中再次创建局部更新控件和进行事件的编写。

16.4　小结

本章介绍了 ASP.NET AJAX 的一些控件和特性，并介绍了 AJAX 基础。在 Web 应用程序开发中，使用一定的 AJAX 技术能够提高应用程序的健壮性和用户体验的友好度。使用 AJAX 技术能够实现页面无刷新和异步数据处理，让页面中其他的元素不会随着“客户端——服务器”的通信再次刷新，这样不

仅能够减少客户端服务器之间的带宽，也能够提高 Web 应用的速度。

虽然 AJAX 是当今热门的技术，但是 AJAX 并不是一个新技术，AJAX 是由一些老技术组合在一起，这些技术包括 XML、JavaScript、DOM 等，而且 AJAX 并不需要在服务器安装插件或安装应用程序框架，只需要浏览器能够支持 JavaScript 就能够实现 AJAX 技术的部署和实现。尽管 AJAX 包括如上诸多的好处，但是 AJAX 也有一些缺点，就是对多媒体的支持还没有 Flash 那么好，并且也不能很好的支持移动移动设备。本章除了介绍 AJAX 基础知识，还介绍了 ASP.NET AJAX 开发中必备的控件。本章还包括:

- ASP.NET 2.0 AJAX：讲解了如何在 ASP.NET 2.0 中实现 AJAX 功能。
- 脚本管理控件（ScriptManger）：讲解了如何使用脚本管理控件。
- 更新区域控件（UpdatePanel）：讲解了如何使用更新区域控件进行页面局部更新。
- 更新进度控件（UpdateProgress）：讲解了如何使用更新进度控件进行更新中进度的统计。
- 时间控件（Timer）：讲解了如何使用时间控件进行时间控制。
- 自定义异常处理：讲解了如何自定义 AJAX 异常。
- 使用母版页的 UpdatePanel：讲解了如何在内容窗体中使用母版页的局部更新控件。
- 母版页刷新内容窗体：讲解了如何在母版页中进行内容窗体控件的局部更新。

虽然 AJAX 包括诸多功能和特性，但是 AJAX 也增加了服务器负担，如果在服务器中大量使用 AJAX 控件的话，有可能造成服务器假死，熟练和高效的编写 AJAX 应用对 AJAX Web 应用程序开发是非常有好处的。