第 5 章 AJAX 技术

5.1 概　　述

AJAX 全称为“Asynchronous JavaScript and XML”(异步 JavaScript 和 XML),是指一种用于创建交互式网页应用的网页开发技术,也是一种运用 JavaScript 和 XML 语言,在网络浏览器和服务器之间传送或接收数据的技术。通常称 AJAX 页面为无刷新 Web 页面。

AJAX 并没有创造出某种具体的新技术,它使用的所有技术都是在很多年前就已经存在了,然而 AJAX 以一种崭新的方式来使用所有这些技术,使得古老的 B/S 方式的 Web 开发焕发了新的活力,迎来了第二个春天。在 AJAX 技术之中,最核心的技术就是 XMLHttpRequest,XMLHttpRequest 可以在不重新加载页面的情况下更新网页,即实现了布局刷新功能。XMLHttpRequest 可以同步或异步地返回 Web 服务器的响应,并且能够以文本或一个 DOM 文档的形式返回内容,这是 AJAX 程序架构的一项关键功能。

与传统的 Web 开发不同,在 AJAX 应用中,每个页面都包括一些使用 JavaScript 开发的 AJAX 组件。这些组件使用 XMLHttpRequest 对象,以异步的方式与服务器通信,从服务器获取需要的数据后更新页面中的一部分内容,它使浏览器可以为用户提供更为自然的浏览体验。在 AJAX 之前,Web 站点强制用户进入提交/等待/重新显示范例,用户的动作总是与服务器的“思考时间”同步。借助于 AJAX,可以在用户单击按钮时,使用 JavaScript 和 DHTML 立即更新用户界面(UI),并向服务器发出异步请求,以执行更新或查询数据库。当请求返回时,就可以使用 JavaScript 和 CSS 来相应地更新 UI,而不是刷新整个页面。最重要的是,用户甚至不知道浏览器正在与服务器通信。

AJAX 应用与传统的 Web 应用的区别主要在以下 3 个方面。

- 不刷新整个页面,实现页面局部与服务器端的动态交互。
- 使用异步方式与服务器通信,不需要打断用户的操作,具有更加迅速的响应能力。
- 应用服务仅由少量页面组成。大部分交互在页面之内完成,不需要切换整个页面。

由此可见,AJAX 使得 Web 应用更加动态,具有更高的智能,并且提供了表现能力丰富的 AJAX UI 组件。目前 AJAX 已经成为了 Web 应用的主流开发技术,大量的业界巨头已经采纳并且在大力推动这个技术的发展,其中非常引人注目的如 Google 的 Google Maps 和微软的 Windows Live 等。由于 AJAX 是客户端技术,所以对浏览器的依赖性较大,用户使用老旧浏览器时可能会受影响,使用时需要有限考虑浏览器兼容问题。

到这里，读者应该对AJAX有一个总体印象了。那么AJAX具体是怎么实现的呢？

AJAX的工作原理相当于在用户和服务器之间加了一个中间层即AJAX引擎，使用户请求与服务器响应异步化。这样使页面像桌面程序一样不必每次都刷新，也不用每次将数据处理的工作都交给服务器去做，而是把以前的一些服务器负担的工作转交给客户端，利用客户端闲置的处理能力来处理，减轻服务器和带宽的负担。简而言之，就是通过XmlHttpRequest让客户端可以使用JavaScript向服务器提出请求并处理响应，而不阻塞用户。

下面以购物车为例，展示AJAX是如何减轻服务器和带宽负担的。

传统Web站点中，在用户单击一个按钮时，会触发一个页面回送效果，用于整个页面的更新，这样在客户端与服务器之间就传输了整个页面的数据。假如用户需要的只是更新页面中很小的一块区域，如购物车中的账单总额信息，上面的机制显然不合适，尤其是在带宽比较小或服务器负载比较大时，对用户的上网体验有很大的影响。如果使用AJAX技术，上面的问题就迎刃而解了。用户将需要更新的那一块小区域单独拿出来，每次单击按钮时，不再产生整个页面的回送，而仅仅是这个小区域的局部回送而已，这样，服务器就不必处理整个页面的请求了，带宽负载也由上百千字节降到几千字节而已，由此可以提供响应更加灵敏的UI，并消除页面刷新所带来的闪烁，用户体验可见一斑。

5.2 AJAX控件

图5-1给出了Visual Studio工具箱中的AJAX扩展，主要包括ScriptManager Timer、UpdatePanel以及UpdateProgress等服务器端控件。这些AJAX控件在使用时与其他ASP.NET控件一样方便。下面介绍这些控件及其使用方法。

图5-1 ASP.NET AJAX服务器控件

5.2.1 ScriptManager控件

ScriptManager控件是AJAX功能的核心，是客户端页面和服务器之间的桥梁。它用来处理页面上的所有组件以及页面局部更新，包括将Microsoft AJAX库的JavaScript脚本下载到浏览器中生成相关的客户端代理脚本，以及能够在JavaScript中访问Web Service。主要的功能如下。

(1) 负责自动建立客户端浏览器上需要的AJAX Client-Script(也就是JavaScript代码)，并且针对页面上需要的各项JavaScript机制进行处理。

(2) ScriptManager控件对于整个异步Postback有着决定性的影响，配合UpdatePanel提供异步Postback的能力，并且“管理”异步Postback的进行。

(3) 让开发人员可以通过前端的JavaScript代码来调用后端的Web Services，提供手动的AJAX功能。

(4) 提供Microsoft AJAX Library中的Client-Script，让开发人员可以简化JavaScript的撰写，并且扩充JavaScript的功能。

因此，无论需要何种 AJAX 功能，都需要在页面上拖曳出 ScriptManager 控件，以作为一切的基础。如果只是在一小部分的页面上需要 AJAX 功能，那么通常可以将 ScriptManager 控件直接放到内容页中，如果在整个站点都需要 AJAX，那么将 ScriptManager 控件放到母版页中是一个理想的解决方案，这样在各内容页中就不需要放置 ScriptManager 控件了。但需要注意的是，所有需要支持 AJAX 的 ASP.NET 页面上有且只能有一个 ScriptManager 控件。如果在母版页中已添加了 ScriptManager 控件，则在内容页中就不能再添加 ScriptManager 控件。如果这时还要在内容页中使用 ScriptManager 控件的其他功能，可以通过添加 ScriptManagerProxy 控件来实现。

ScriptManager 控件有许多属性，其中绝大部分用于高级场景，对于简单应用来说，不需要改变 ScriptManager 控件的任何属性，但是在面对复杂的、更加丰富的应用时，就需要更改相关的属性了，感兴趣的读者可以查阅相关资料进一步学习。

5.2.2 UpdatePanel 控件

UpdatePanel 控件可以用来创建丰富的局部更新的 Web 应用程序。UpdatePanel 本身是一个容器控件，控件本身不会显示任何内容，仅相当于页面中的一个小局部区域，用于实现局部刷新和无闪烁页面。UpdatePanel 控件的使用可以大大减少客户端脚本的编写工作量。在基本的应用程序中，只要将相关控件放入 UpdatePanel 中即可。当 UpdatePanel 控件中的某个控件产生到服务器端的回送时，只刷新 UpdatePanel 区域，其外的页面部分并不会更新。

实例 5-1 认识局部刷新。

局部刷新功能在第 4 章的实例 4-10 中已经使用过，主要使用 UpdatePanel 控件，该控件提供了一个范围，即局部刷新的范围。将需要局部刷新功能部分放置在 UpdatePanel 范围内，即可实现局部刷新功能，没有放置在 UpdatePanel 范围内的控件将会引起整个页面刷新。下面通过实例进一步理解局部刷新。

(1) 新建一个 Web 窗体，页面添加一个 Button 控件(Button1)和一个 Label 控件(Label1)，然后在 AJAX 控件组中拖取一个 ScriptManager 控件和一个 UpdatePanel 控件，最后在 UpdatePanel 里面放入一个 Button 控件(Button2)和一个 Label 控件(Label2)，如图 5-2 所示。

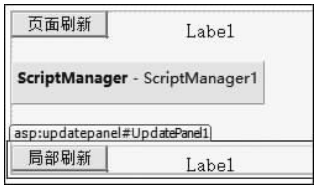

图 5-2 前台设计图

(2) 添加两个按钮事件，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToLongTimeString();
}
protected void Button2_Click(object sender, EventArgs e)
{
```

```
    Label2.Text=DateTime.Now.ToLongTimeString();
}
```

(3) 运行,单击 Button1 按钮,观察浏览器,可以看到整个页面的回送;单击 Button2 按钮,观察浏览器,看不到整个页面的回送。但是通过时间的改变,能够知道 Label2 所在的小区域发生了页面的局部回送,引起 Label2 的数据更新,如图 5-3 所示。

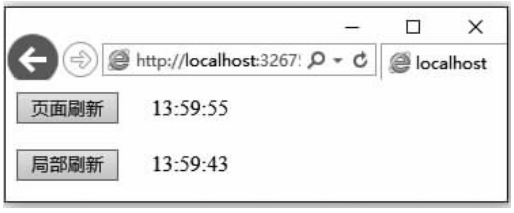

图 5-3 单击 button1 的结果

(4) 看到这里,有些读者也许会产生疑问,因为在 Button2_Click 事件中,并没有改变 Label1 显示的值,所以,在单击 Button2 按钮后,Label1 不应该有变化,也就不足以说明只有 UpdatePanel 里面的内容进行了局部刷新。下面继续以下步骤。

(5) 修改 Button2 按钮事件代码,增加 Label1 文本赋值语句,代码如下:

```
protected void Button2_Click(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToLongTimeString();
    Label2.Text=DateTime.Now.ToLongTimeString();
}
```

(6) 按 F5 键运行,单击 Button2 按钮,我们会发现,只有局部范围内的 Label2 发生了改变,说明 Button2 按钮触发的页面回送只是局部的,并不是整个页面的回送,只有 UpdatePanel 控件所包含的区域进行了局部更新。

至此,似乎可以得到结论:UpdatePanel 控件里面的控件如果能引发页面回送的话,就只更新 UpdatePanel 控件区域;UpdatePanel 控件外面的控件如果引发页面回送的话,UpdatePanel 控件区域也会更新。其实,UpdatePanel 里面的控件也可以引发其外的更新;同样,其外的控件也可以只引发 UpdatePanel 区域更新。在具体讲解前,先看一看 UpdatePanel 控件主要的属性,如图 5-4 所示。

下面,再看一下 UpdatePanel 控件的默认属性。从工具箱中拖取一个 UpdatePanel 控件,打开 UpdatePanel 的属性面板,如图 5-5 所示。

(1) ChildrenAsTriggers 属性的默认值是 True,即 UpdatePanel 控件内部的子控件引发的页面回送都会使得 UpdatePanel 区域的局部刷新。

(2) UpdateMode 属性的默认值是 Always,即页面上任意一个局部更新被触发,此 UpdatePanel 就会更新。当某个页面中有多个 UpdatePanel 共存时,由于 UpdatePanel 的 UpdateMode 属性值默认为 Always,所以页面上如果有一个局部更新被触发,则所有的

属性或方法	说明
ChildrenAsTriggers	应用于UpdateMode属性为Conditional时，指定UpdatePanel中的子控件的异步回送是否会引发UpdatePanle的更新
RenderMode	表示UpdatePanel最终呈现的HTML元素。Block（默认）表示 ，Inline表示
Triggers	用于引起更新的事件。在ASP.NET Ajax中有两种触发器，其中使用同步触发器（PostBackTrigger）只需指定某个服务器端控件即可，当此控件回送时采用传统的“PostBack”机制整页回送；使用异步触发器（AsyncPostBackTrigger）则需要指定某个服务器端控件的ID和该控件的某个服务器端事件
UpdateMode	表示UpdatePanel的更新模式，有两个选项：Always和Conditional。Always是不管有没有Trigger，其他控件都将更新该UpdatePanel，Conditional表示只有当前UpdatePanel的Trigger，或ChildrenAsTriggers属性为true时，当前UpdatePanel中控件引发的异步回送或者整页回送，或是服务器端调用Update()方法才会引发更新该UpdatePanel

图 5-4　UpdatePanel 控件属性图

UpdatePanel 都将更新。这也许与设计初衷不相符，所以为了避免这种情况，可以把 UpdateMode 属性设置为 Conditional，然后为每个 UpdatePanel 设置专用的触发器。

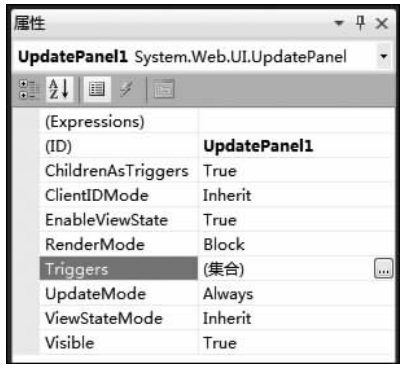

图 5-5　UpdatePanel 属性面板默认值

下面通过示例，深入讲解 UpdatePanel 的各种使用情况。

实例 5-2　内部子控件不再引发 UpdatePanel 刷新。

（1）添加一个 Web 窗体，在页面中拖放一个 ScriptManager 控件和一个 UpdatePanel 控件，在 UpdatePanel 中放入一个 Button 控件（Button1）和一个 Label 控件（Label1），并将 UpdatePanel 控件的 ChildrenAsTriggers 属性设置为 False。

（2）双击 Button1，进入.cs 后台文件，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToLongTimeString();
}
```

（3）按 F5 键运行，结果如图 5-6 所示。

（4）将 UpdateMode 属性值更改为 Conditional，运行后，单击 Button1 按钮并没有显示时间。通过设置断点进行调试，可以知道事件代码语句确实运行了，如图 5-7 所示，说明此时 UpdatePanel 内部子控件没有引发局部刷新。

图 5-6 运行结果图

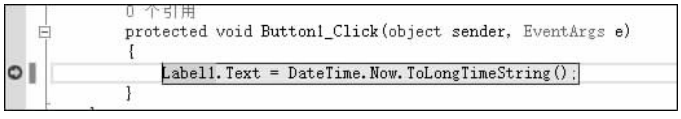

图 5-7 设断点运行图

(5) 将 ChildrenAsTriggers 属性重新设置为 True，按 F5 键运行，单击 Button1 按钮，如图 5-8 所示，说明此时 UpdatePanel 局部刷新了。

图 5-8 运行结果图

(6) 结论：无论 UpdateMode 属性值设置为 Always 还是 Conditional，触发 UpdatePanel 局部刷新时需要将 ChildrenAsTriggers 属性设置为 True。

实例 5-3 页面内多个 UpdatePanel，各自实现局部刷新。

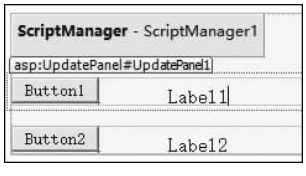

图 5-9 前台设计

(1) 添加 Web 窗体，在页面中拖放一个 ScriptManager 控件和两个 UpdatePanel 控件，在每个 UpdatePanel 中各放入一个 Button 控件和一个 Label 控件，保持 UpdatePanel 的默认属性，如图 5-9 所示。

(2) 添加两个按钮事件代码，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToLongTimeString();
    Label2.Text=DateTime.Now.ToLongTimeString();
}
```

```
protected void Button2_Click(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToLongTimeString();
    Label2.Text=DateTime.Now.ToLongTimeString();
}
```

(3) 按 F5 键运行,无论单击 Button1 或者 Button2 按钮,两个 Label 都会刷新时间,结果如图 5-10 所示。

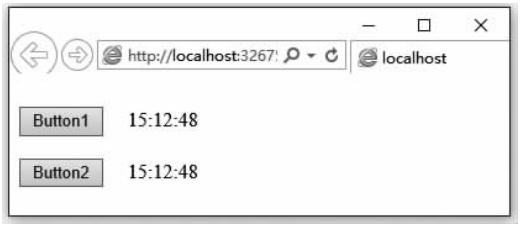

图 5-10　默认属性下运行结果

这是什么原因呢? 这是由于 UpdateMode 属性默认值是 Always。因此,凡是能引发页面回送的操作都会引发 UpdateMode 属性是 Always 的 UpdatePanel 的局部刷新。

(4) 将两个 UpdatePanel 的 UpdateMode 属性值都设置为 Conditional 时,按 F5 键运行。单击 Button1 按钮或者 Button2 按钮时,就只会刷新各自所在 UpdatePanel 的时间了。

默认情况下,用户总会在各自 UpdatePanel 内放置按钮,这样的结果就是:

- 当 UpdateMode 属性值为 Always 时,任何一个 UpdatePanel 内部的任何变化都会引发其他 UpdatePanel 更新;
- UpdateMode 属性值设置为 Conditional,分别引发各自的 UpdatePanel 的刷新。

其实,当 UpdateMode 属性值设置为 Conditional 时,可以设置 UpdatePanel 外部控件作为触发器,也可以在其他 UpdatePanel 的内部设置子控件作为触发器,甚至某个内部子控件都可以引发整个页面的刷新,具体使用方法通过一个实例来认识。

实例 5-4　Conditional 前提下引发的刷新。

(1) 添加一个 Web 窗体,在页面中拖放一个 ScriptManager 控件、设置前台页面如图 5-11 所示。

(2) 首先添加“外部控件引发局部刷新功能”中的按钮(Button1)事件代码,代码如下:

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToLongTimeString();
    Label2.Text=DateTime.Now.ToLongTimeString();
}
```

图 5-11 前台设计图

(3) 首次运行,单击 Button1 按钮,可以看到 UpdatePanel 内外的 Label 控件都刷新了,事件保持一致,并且整个页面都回送了,这是因为 Button1 在 UpdatePanel 外部,会引发整个页面的回送更新。结果如图 5-12 所示。

图 5-12 默认属性下运行结果

(4) 继续将 UpdateMode 属性设置为 Conditional,运行结果基本相同。说明 UpdatePanel 外部控件造成的页面回送是整个页面范围的,包括 UpdatePanel 区域。那么有没有一种方法可以做到 UpdatePanel 外部的控件只引发 UpdatePanel 区域的局部刷新,而不造成整个页面的回送呢? 有的,使用 UpdatePanel 的 Triggers 属性。

(5) 打开 UpdatePanel1 的属性面板,单击 Triggers 集合。在打开的编辑器中,单击"添加"右边的小三角,选择 AsyncPostBackTrigger(一个 UpdatePanel 可以添加多个触发器),在绑定到触发器的 ControlID 中选择 Button1,EventName 选择 Click,最后单击"确定"按钮,如图 5-13 所示。

(6) 按 F5 键运行,单击 Button1 按钮,结果如图 5-14 所示,只有 UpdatePanel 刷新了。

(7) 现在添加"其他内部控件引发局部刷新功能"按钮事件,代码如下:

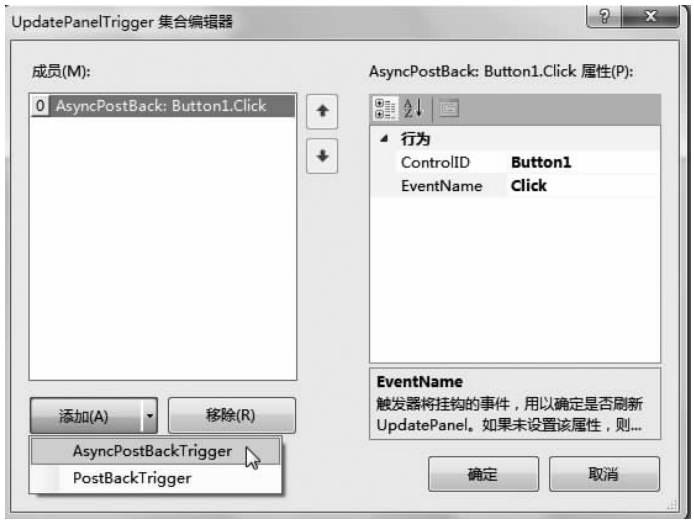

图 5-13　Trigger 编辑器集合

图 5-14　异步回送触发器下运行结果

```
protected void Button2_Click(object sender, EventArgs e)
{
    Label3.Text=DateTime.Now.ToLongTimeString();
    Label4.Text=DateTime.Now.ToLongTimeString();
}

protected void Button3_Click(object sender, EventArgs e)
{
    Label3.Text=DateTime.Now.ToLongTimeString();
    Label4.Text=DateTime.Now.ToLongTimeString();
}
```

(8) 根据之前步骤的讲解，给 UpdatePanel2 和 UpdatePanel3 分别设置属性值：ChildrenAsTriggers＝false，UpdateMode＝Conditional，Triggers 设置为需要的按钮事件。

(9) 按 F5 键运行，单击 Button2 按钮，会引发 UpdatePanel3 区域的局部刷新，同样，单击 Button3 按钮，会引发 UpdatePanel2 区域的局部刷新。这样就实现了内部子控件控

制其他区域的局部刷新功能。

(10) 最后添加“内部子控件引发页面刷新功能”按钮事件代码,代码如下:

```
protected void Button4_Click(object sender, EventArgs e)
{
    Label5.Text=DateTime.Now.ToLongTimeString();
    Label6.Text=DateTime.Now.ToLongTimeString();
}
```

(11) Triggers属性可以添加两种属性值,异步可以理解为局部刷新,同步可以理解为页面刷新。这里实现UpdatePanel4内的子按钮引发整个页面刷新,只需要给该按钮添加同步回送触发器PostBackTrigger即可,设置如图5-15所示,运行程序,可看到Label5和Label6都显示相同的时间,说明整个页面刷新了,如图5-16所示。

图5-15 UpdatePanel4添加PostBackTrigger

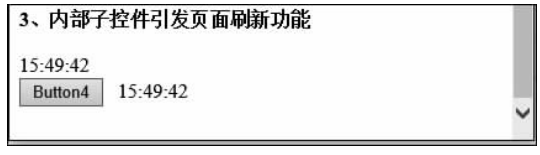

图5-16 同步回送触发器下运行结果

UpdatePanel控件涉及的属性不多,但要灵活掌握才能准确引发开发所需要的局部刷新。特别需要注意的是,用户一般通过aspx设计页面添加UpdatePanel控件,然后继续在设计页面中定位UpdatePanel范围,添加子控件。如果用户需要在HTML页面添加UpdatePanel以及子控件,需要注意一对重要的标签<ContentTemplate></ContentTemplate>。例如在UpdatePanel控件中添加Button控件,生成的HTML代码如下: