ASP.NET Ajax 框架教程

郑 健

目录

1.概述

利用业余时间学习了一下微软的ASP.NET Ajax 框架，我在学习时顺便整理了这个教程。此教程主要针对开发应用场景和功能点进行展开示例，包括 26 个精简的小例子，在实际开发中也可以作为查找手册使用。

ASP.NET Ajax 更偏重于客户端编程，因此使用它不仅使编程更加灵活，而且由于 Ajax 是轻量级，请求效率也较高。

2.应用场景代码示例

（1）ScriptManager 控件示例

1）在异步调用服务端注册客户端脚本新方法

前台页面代码:

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- 在服务端注册客户端脚本新方法 -->
        <!-- 通过 Page.ClientScript 实例注册客户端脚本方法在异步提交时不起作用。 Microsoft 采用 ScriptManager 实例，并与 Page.ClientScript 方法一一对应的方法来实现此功能，具体看示例后台代码。 -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                当前时间: <%= DateTime.Now %>
                <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click1" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

后台服务端代码:

```
public partial class _AA_ScriptManager_RegistClientScript_DefaultPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click1(object sender, EventArgs e)
    {
        //Ajax 框架中新调用方式
        ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.GetType(), "UpdateSucceed", "alert('Update time succeed!')", true);
        //默认调用方式(在异步调用 XmlHttp 方式中无效)
dateSucceed", "<script>alert('Update time succeed!')</script>");
```

```
    }
}
```

2） 捕获 Ajax 异步调用中错误(默认使用 alert 提示)

前台页面代码:

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- 在ASP.NET Ajax 框架的异步调用中, 默认情况下不会直接抛出错误详细信息. 可以借助 ScriptManager 的OnAsyncPostBackError 事件来实现 -->
        <!-- AllowCustomErrorsRedirect="false" 表示遇到错误不跳转到 Web.Config 中定义的错误处理页面
            <system.web>
                <customErrors mode="On" defaultRedirect="~/DisplayError.aspx"></customErrors>
            </system.web>
            如果设置为 true, 则出错时会自动将页面跳转到当前站点根目录下面的 DisplayError.aspx 页面.
         -->
        <asp:ScriptManager ID="ScriptManager1" runat="server" AllowCustomErrorsRedirect="false" OnAsyncPostBackError="ScriptManager1_AsyncPostBackError">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

后台页面代码:

```
public partial class _AA_ScriptManager_GetAsycError_DefaultPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        int a = 5, b = 0;
        int c = a / b;
    }
    protected void ScriptManager1_AsyncPostBackError(object sender, AsyncPostBackErrorEventArgs e)
```

```
    {
        ScriptManager.GetCurrent(this
    }
}
```

2） 捕获 Ajax 异步调用中的错误(自定义输出错误方式)

前台页面代码:

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- 在Asp.net Ajax 框架的异步调用中, 默认情况下不会直接抛出错误详细信息. 可以借助 ScriptManager 的OnAsyncPostBackError 事件来实现 -->
        <!-- AllowCustomErrorsRedirect="false" 表示遇到错误不跳转到Web.Config 中定义的错误处理页面
            <system.web>
                <customErrors mode="On" defaultRedirect="~/DisplayError.aspx"></customErrors>
            </system.web>
            如果设置为 true, 则出错时会自动将页面跳转到当前站点根目录下面的 DisplayError.aspx 页面.
         -->
        <!--注意 add_endRequest 方法不能写到<head></head>中, 因为这时ScriptManager 实例还没有创建. 不好之处是方法写在<head></head>块中使页面有些乱.-->
        <asp:ScriptManager ID="ScriptManager1" runat="server" AllowCustomErrorsRedirect="false">
       </asp:ScriptManager>
       <asp:UpdatePanel ID="UpdatePanel1" runat="server">
           <ContentTemplate>
               <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
           </ContentTemplate>
       </asp:UpdatePanel>
       <div id="divMessage" style="color:Red;"></div>
       <script type="text/javascript" language="javascript">
function(sender, e)
           {
               e.set_errorHandled(true);  //表示自定义显示错误, 将默认的 alert 提示禁止掉.
               $get("divMessage").innerHTML = e.get_error().message;

           });
       </script>
    </div>
</form>
```

```
</body>
后台服务端代码:
public partial class
_AA_ScriptManager_GetAsycDetailError_CustomDisplayError_DefaultPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        int a = 5, b = 0;
        int c = a / b;
    }
}
```

（2）UpdatePanel 控件示例4）RenderMode 属性用法示例

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- RenderMode 属性功能与 Html 标签的 style.display 属性作用一
样, 只是 UpdatePanel 只有 Block 和 Inline 两种方式 -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
       </asp:ScriptManager>
       <asp:UpdatePanel ID="UpdatePanel1" runat="server"
RenderMode="Block">
          <ContentTemplate>
              UpdatePanel1(Display 设置为 Block)
          </ContentTemplate>
       </asp:UpdatePanel>
       <asp:UpdatePanel ID="UpdatePanel2" runat="server"
RenderMode="Block">
          <ContentTemplate>
              UpdatePanel2(Display 设置为 Block)
          </ContentTemplate>
       </asp:UpdatePanel>
       <asp:UpdatePanel ID="UpdatePanel3" runat="server"
RenderMode="Inline">
          <ContentTemplate>
              UpdatePanel3(Display 设置为 Inline)
          </ContentTemplate>
       </asp:UpdatePanel>
       <asp:UpdatePanel ID="UpdatePanel4" runat="server"
RenderMode=Inline>
          <ContentTemplate>
              UpdatePanel4(Display 设置为 Inline)
```

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

5） UpdateMode 用法示例

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- UpdateMode 属性可以设置为Always 和Conditional 两种方式. 默认情况下属性值为 Always. -->
        <!-- 如果设置为 Conditional, 则只有当前 UpatePanel 内部的元素(比如 button)提交时, 才能引起当前 UpdatePanel 更新;-->
        <!-- 如果设置为Always, 则不管点击UpdatePanel 内部还是外部的按钮都会使当前 UpdatePanel 更新 -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode=always>
            <ContentTemplate>
                UpdatePanel1 时间:<%= DateTime.Now %>
                <asp:Button ID="Button1" runat="server" Text="Button" />
            </ContentTemplate>
        </asp:UpdatePanel>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server" UpdateMode=conditional>
            <ContentTemplate>
                UpdatePanel2 时间:<%= DateTime.Now %>
                <asp:Button ID="Button2" runat="server" Text="Button" />
            </ContentTemplate>
        </asp:UpdatePanel>
        <br />
    </div>
</form>
</body>
```

6） ChildrenAsTriggers 属性用法示例

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- ChildrenAsTriggers 属性可以设置为true 或false. 默认情况下属性值为 true. -->
        <!-- 如果设置为 false, 则点击当前 UpdatePanel 中的元素不会引起当前 UpdatePanel 更新;但它可能会引起本 UpdatePanel 之外的页面局部更新. -->
```

```
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                UpdatePanel1 时间: <%= DateTime.Now %>
            </ContentTemplate>
        </asp:UpdatePanel>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server"
UpdateMode=conditional ChildrenAsTriggers="false">
            <ContentTemplate>
                UpdatePanel2 时间: <%= DateTime.Now %>
                <asp:Button ID="Button1" runat="server" Text="Button" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

5） Triggers 属性用法示例

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- PostBackTrigger(AsyncPostBackTrigger)标记可以指定当前
UpdatePanel 之外的元素来对自己进行更新.不同:一个是一般页面提交, 一个
是异步无刷新提交 -->
        <!-- 假如设置了 UpdateMode="Conditional", 则只有点击当前
UpdatePanel 中的 button 才能更新本 Panel 中的内容. 如果想设置本
UpdatePanel 外的元素对本 Panel 内容进行更新, 则可以设置该属性. -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server"
UpdateMode="Conditional">
            <ContentTemplate>
                当前时间: <%= DateTime.Now %>
                <asp:Button ID="Button1" runat="server" Text="Button" />
            </ContentTemplate>
            <Triggers>
                <asp:PostBackTrigger ControlID="Button2" />

                <asp:AsyncPostBackTrigger ControlID="Button3" />
            </Triggers>
        </asp:UpdatePanel>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button2" runat="server" Text="Button" />
            </ContentTemplate>
```

```
        </asp:UpdatePanel>
        <asp:UpdatePanel ID="UpdatePanel3" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button3" runat="server" Text="Button" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

（1）UpdateProgress 控件示例8）在异步更新时显示滚动进度条

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- UpdateProgress 控件 -->
        <!-- AssociatedUpdatePanelID 表示由哪个 UpdatePanel 来使自己呈现; -->
        <!-- DynamicLayout 表示 UpdateProgress 是否固定占有一定空间,即使是隐藏时; 如果该值为 true, 则只有显示时才占用页面空间. -->
        <!-- DisplayAfter 表示显示 UpdateProgress 内容之前需要等待的时间.  -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
       </asp:ScriptManager>
       <asp:UpdateProgress ID="UpdateProgress1" runat="server" DynamicLayout=false AssociatedUpdatePanelID="UpdatePanel1" DisplayAfter="1000">
           <ProgressTemplate>
                <asp:Image ID="Image1" runat="server" ImageUr="~/(C)UpdateProgress/(8)UpdateProgress/Progress.gif" ImageUrl="~/(C)UpdateProgress/(8)UpdateProgress/Progress.gif" />
           </ProgressTemplate>
       </asp:UpdateProgress>
       <asp:UpdatePanel ID="UpdatePanel1" runat="server">
           <ContentTemplate>
               当前时间: <%= DateTime.Now %>
               <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
           </ContentTemplate>
       </asp:UpdatePanel>
    </div>
    </form>
</body>
```

（2） Timer 控 件 示 例9）在客户端无刷新定时执行服务端方法

```
<body>
```

```
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- Timer 控件 -->
        <!-- 通过设置 Interval 值(毫秒)可以定期的更新页面; 可以配合
UpdateMode 来禁止某些 UpdataPanle 不更新. -->
        <!-- 如果把Timer 置于UpdatePanel 外面, 可以非异步提交整个页面.
-->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                当前时间: <%= DateTime.Now %>
                <asp:Timer ID="Timer1" runat="server" Interval="1000">
                </asp:Timer>
            </ContentTemplate>
        </asp:UpdatePanel>
        <%--<asp:Timer ID="Timer1" runat="server" Interval="1000">
        </asp:Timer>--%>
        <%--<asp:UpdatePanel ID="UpdatePanel2" runat="server"
UpdateMode=conditional>
            <ContentTemplate>
                当前时间: <%= DateTime.Now %>
            </ContentTemplate>
        </asp:UpdatePanel>--%>
    </div>
    </form>
</body>
```

(3) Ajax 中新 Validators 控件用法示例

使用VS 2005 默认的验证控件在异步操作时会有些问题, 微软推出了与现有控件名称一一对应的一系列验证控件。

10) Validators 控件的使用配置示例

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- Validators 控件 -->
        <!-- Asp.net Ajax 与 VS 2005 自带的系列验证控件在使用时,会有些
问题.固 Microsoft 又推出一系列与原控件名称一一对应的控件集, 在 bin 目录
的 Validators.dll -->
        <!-- 另外, 还要在 Web.Config 中指定使用 bin 下面的
Validators.dll 中的验证控件:
            <system.web>
                <pages>
                    <tagMapping>
ors"/>
```

```
                    </tagMapping>
                </pages>
            </system.web>
        -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:TextBox ID="TextBox1"runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator
ControlToValidate="TextBox1" ID="RequiredFieldValidator1"
runat="server" ErrorMessage="不能为
NULL!"></asp:RequiredFieldValidator>
                <br />
                <asp:Button ID="Button1" runat="server" Text="Button" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

（1） 在客户端请求服务端最基本的执行方式

11） 使用 Ajax library 类库中的客户端 WebReqest 对象请求服务端

前台页面代码:

```
<body>
    <form id="form1" runat="server">
        <!-- 注释 -->
        <!-- 使用客户端 WebRequest 类, 进行一个标准的 Ajax 请求示例 -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <script language="jscript" type="text/javascript">
            function ExecuteAjaxRequest(text)
            {
                var request = new
                request.set_url('ClientWebRequest.ashx');
                request.set_httpVerb("POST");
                request.add_completed(onComplete);
                request.set_body('text=' + encodeURIComponent(text));

                request.invoke();
            }
            function onComplete(response)
            {
                if (response.get_responseAvailable())
                {
                    var text = response.get_object();
                    alert(text);
```

```
            }
        }
    </script>
    <input type="button" value="Hello Word!"
onclick="ExecuteAjaxRequest('Hello Word!')" />
</form>
</body>
处理程序 ashx 文件代码:
using System;
using System.Web;
using
public class ClientWebRequest : IHttpHandler {
    public void ProcessRequest (HttpContext context) {
"text/plain";
        string"text"];
        JavaScriptSerializer jsSerializer = new JavaScriptSerializer();
    }
    public bool IsReusable
    {
        get
        {
            return false;
        }
    }
}
```

（1） 在客户端调用页面后台(Page behind)中的方法

12） 在客户端调用页面后台(Page behind)中的方法示例

```
前台页面代码:
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- 在开发中使用较多的是，在客户端调用当前页面的服务端方法.
-->
        <!-- 设置ScriptManager 控件的EnablePageMethods 属性为:true，并
且在需要使用后台方法前加属性标记[WebMethod] -->
        <!-- 服务端方法必须为静态的，需要客户端调用的方法才设置为静态
的 -->
        <asp:ScriptManager ID="ScriptManager1" runat="server"
EnablePageMethods="true" />
        <input type="button" value="调用服务端方法"
onclick="ExecuteServerMethod('ChengKing')" />
        <script language="javascript" type="text/javascript">
            function ExecuteServerMethod(value)
            {
```

```
    PageMethods.ReturnStringServerMethod(value,CallBackResult);
            }
            function CallBackResult(result)
            {
                alert(result);
            }
        </script>
    </div>
    </form>
</body>
```

页面后台服务端代码:

```
using
public partial class _G_Ajax_Visit_PageServer_Method_DefaultPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    [WebMethod]
    public static string ReturnStringServerMethod(string str)
    {
        return "Hello " + str;
}
}
```

（1） 在客户端调用 WebService 中的服务端方法

13） 调用 WebService 示例

前台页面代码:

```
<body>
    <form id="form1" runat="server">
    <div>
        <!-- 注释 -->
        <!-- 如果WebService 类代码在App_Code 中，则直接可以通过：类名.方法调用；否则，要通过：命名空间.类名.方法名称 调用. -->
        <!-- InlineScript=true 表示在客户端把调用服务端生成的客户端代码脚本输出到页面上；一般用于调试使用，但肯定占用时间. -->
        <asp:ScriptManager ID="ScriptManager1" runat="server">
            <Services>
                <asp:ServiceReference Path="SimpleTransferWebservice.asmx" InlineScript=false />
            </Services>
        </asp:ScriptManager>
        <script language="javascript" type="text/javascript">
          function showEmployee(str)
          {
              SimpleTransferWebservice.HelloWorld(
                  str,
                  onCompleted);
```