相关主题 RECOMMEND ARTICLE
- 在ASP.NET 2.0中建立站点导航层次
- JSP和JSF双剑合并 打造完美Web应用
- 用ASP.NET2.0在数据库中存储二进制文件
- ASP.NET中上传文件到数据库
- ASP.NET定制简单的错误处理页面
- 基于JSF技术的WEB应用开发研究
- ASP.NET实现投票结果的图片进度条显示
- 圣殿骑兵PHP 2007年Web开发技术预言

MORE

推荐文章 RECOMMEND ARTICLE
- 数据广播方案的优化
- 网络游戏的位置同步
- 游戏音乐制作案例之《战火 红色警戒》音效制作揭秘
- 英雄连Online 原画
- 游戏音乐制作案例之《乱武天下》
- 游戏音乐制作案例之《诛仙》
- 《鹿鼎记》最新原画
- MIDP2.1规范的新特性

MORE

热门文章 HOT ARTICLE
- [电子书下载]游戏设计 — 原理与实践
- [电子书下载]网络游戏开发
- 游戏设计全过程
- [电子书下载]游戏设计技术
- [电子书下载]游戏设计理论
- CS游戏人物模型制作教程
- CG人物插画基本流程
- [转贴]MAX高级人头教程

MORE

您的位置： WEB技术

文章标题	细节决定成败 ASP.NET中的蝴蝶效应		
来源：	[ogdev]	浏览：	[461]

前言

ASP.NET的优点我说过很多次了，也就是各个控件独立负责自己内部的逻辑，这是一个好事情，因为它解决了原本ASP处理逻辑耦合度高的问题。然而这是需要代价的，那就是引入ASP.NET页面生命周期，随着控件的多层嵌套，应用的复杂度增加，我们再次陷入泥潭！

问题

其实这个文章题目我两个月前就写下了，可是一直没想写完它，直到今天我在这个泥潭中泡了几个小时，于是决定先从泥潭中跳出来把文章写完，再跳进去继续解决问题。问题是这样的：

使用MS AJAX 1.0 Beta2 + 2.0 CTP新建一个项目，同时在Bin中放上Beta2的AjaxControlToolkit.dll。

扔上一个Accordion，放置几个AccordionPane，设置一下CssClass。

在Page_Load中使用Page.LoadControl加载一个UserControl，然后添加到页面上。

接着发现UserControl内的控件无法正常触发事件，陷入泥潭中……

首先要说明，如果仅仅做第3步那个UserControl肯定正常运作，那意味着问题出在ScriptManager或Accordion中出现了问题。

正文

想知道到底是什么出问题了吗?先听我说说这个ASP.NET页面生命周期的问题吧。

由于生命周期按阶段划分，任务在不同阶段按部就班完成，所以我们的每一个操作都是阶段相关的，有些操作仅能在特定的阶段操作，有些操作在不同阶段执行会导致不同的结果。当然，MS希望尽量消除这些阶段间的差异，例如让一个操作在尽可能多的阶段中都能执行，并且尽可能减少在不同阶段中操作引发的不同结果。然而这不可能完全做到，例如我们都知道ViewState读写限制为仅能在某些阶段进行，于是依赖于ViewState的控件属性也就因此受到同样的限制。

控件属性读写受阶段限制，这很好接受，对吧?因为这仅仅是一层依赖关系。顺着依赖关系推广出去，情况会变得越来越复杂，限制的原因埋藏得越来越底层，接着我们发现复杂性这一问题在ASP.NET这种结构良好的体系中出现了，而消灭这种复杂性的银弹还没被发明。

作为控件或组件的开发人员，我们当然有义务消除阶段差异，让下游的开发人员面对更低的复杂性，而且我们也确实尽力去做了。控件的每一层封装，都包含着这种努力，并向上承诺尽可能低的阶段差异。然而为了让控件看起来简单易用，我们不可能将这些差异完整地记录在文档之中，我们尝试去隐瞒细节，控件被层层封装时我们都这样做。底层文档没告诉我的差异，我当然也没必要写到这一层的文档上去;底层文档提及了的差异，我尽力弥补了，即使弥补得不太好，也不写到这一层的文档上去。于是文档就好像神话传说一样随着世代相传而改变，最终没有人知道这个控件依赖于某些底层的阶段差异。

做过控件开发的人都知道，有时候我们必须根据实际情况采用不同的方式构建看起来一样的控件。例如最简单的数据控件都会存在是否PostBack的构建差异，如果是非PostBack，则需要在DataBind时构建并将数据保存到ViewState，如果是PostBack则根据ViewState直接构建，如果PostBack后又遇到了DataBind则需要清除原来的构建并重新根据新数据构建。再复杂一些的控件，还会分步骤构建，默认情况下为了消除使用方的阶段差异，部分构建步骤会尽可能靠前到Init时执行，而另外一部分构建步骤则尽可能推迟到PreRender时执行，中间部分则尽可能减少自己的变化以便使用方操作。然而事情不会那么简单，使用方的某些操作(通常是访问某个属性)如果依赖于某个构建步骤的完成，因此一旦这些操作出现，原本在PreRender才执行的特定构建步骤就要提前执行，当这样的操作在不同阶段进行多几次，构建步骤就已经散落在页面生命周期的各阶段。

构建步骤可能散落于页面生命周期的各阶段对于控件设计师来说是一个严峻的问题，这意味着他要保证任何一个构建步骤在任何一个阶段执行都是无差异的，当然这不可能做到，于是又要引入别的机制来减少这种差异，复杂性就此产生了，接下来随着复杂性的增加控件设计师越来越无法确保较低的阶段差异程度，这就到控件使用者遭殃了，如果控件使用者又再把控件封装，并且依然企图降低阶段差异程度，那么灾难也就发生了……

结果

我花了几个小时在泥潭中泡了几个小时，边泡边写这篇文章，问题当然已经有结果了。

如果Accordion设置了HeaderCssClass或者ContentCssClass，那就会出问题，但如果为AccordionPane都加上以上两个属性，又不会有问题了。这样的情况当然通过用Reflector查看这两个类的代码来解决，结果发现Accordion会检测每一个AccordionPane是否有设置这两个属性，如果没有就把AccordionPane的设置为和自己的一样。在AccordionPane被设置时，会调用this.EnsureChildControls()，这是一个会导致构建步骤提前执行的方法，于是控件构建的顺序就改变了，不仅仅Accordion内部的顺序改变了，整个Page的都改变了。由于控件的ID是按顺序自动分配的，包括我那个UserControl，构建顺序的改变意味着ID的改变，也就相当于整个控件树都改变了，事件当然不能正常触发。

最后的解决方案当然是为我那个UserControl指定ID。我花了那么多个小时才发现自己做了件蠢事，一早打开Trace来看控件树就应该能发觉UniqueID的变化。

总结

虽然这个问题看起来不是一个太好的例子，因为一打开Trace就应该能找到问题的来源，但实际上它却正好揭示了ASP.NET框架内部的“蝴蝶效应(Butterfly Effect)”——随着复杂度的增加，任何一个细微的改变都会导致全局上的巨大变化。在设计ASP.NET的时候，MS可能也在想着解耦，在简单的情况下这东西确实也解耦，然而在复杂的情况下却正好背道而驰，这真的是很讽刺。

无任何评论！

请您注意：

- 尊重网上道德，遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规
- 尊重网上道德，遵守中华人民共和国的各项有关法律法规
- 承担一切因您的行为而直接或间接导致的民事或刑事法律责任
- 中国网游研发中心新闻留言板管理人员有权保留或删除其管辖留言中的任意内容
- 您在中国网游研发中心留言板发表的作品，中国网游研发中心有权在网站内转载或引用
- 参与本留言即表明您已经阅读并接受上述条款

发表评论：

昵　　称：

GO

联系EMAIL：