用户控件包装器的设计与实现

The Design and Actualization of User Control Wrapper

林茂长 周东清 （大连理工大学电信学院计算机系 大连市 116024）

摘要：本文介绍了基于 Microsoft SharePoint Portal Server 2003（以下简称 SPS2003）上的 Web Part 开发的利器－用户控件包装器的开发原理和步骤。通过用户控件包装器的包装，Web Part 开发人员只需要对 ASP. NET 中的用户控件进行开发即可，实现了 Web Part 开发的可视化编程，保留了 Web Part 的原有功能，简易化了 Web Part 之间的数据传递编程，进而使得基于 SPS2003 的门户网站设计更为方便，个性化网站的特性更容易得到展现。

关键词：SPS2003 Web Part 用户控件

1 引言

SPS2003 是一个安全、可伸缩的企业级门户服务器。可以利用它将 SharePoint 站点、信息和应用程序汇集到一个单一的门户位置，用户可以通过门户内容和布局进行个性化的定制，更快地找到相关信息。目前，许多企业和政府部门已开始基于 SPS2003 服务在互联网上提供信息共享与应用服务，并构建跨企业的虚拟组织或虚拟企业，以实现大规模的资源共享。Web Part 的自定义开发是实现基于 SPS2003 上的复杂应用的关键，微软官方提供的 Web Partforvs2003 模版是开发 Web Part 的标准编程环境，但是非常可惜，它不提供 Web Part 开发的可视化界面，对于复杂界面的 Web Part 的开发，将是一件非常吃力的事情。用户控件包装器巧妙的利用 Web Partforvs2003 模版，通过 Web Part 的属性将用户控件载入页面，通过菜单编辑用户控件，通过 Web Part 的数据传递实现用户控件之间的数据传递。

2 用户控件与 Web Part 的联系

ASP. NET 为扩展服务器控件框架提供了两个抽象：用户控件和自定义控件。

用户控件实质上是可插入其他页面中的 ASP. NET 页面，它们在一定程度上类似于传统 ASP 中使用的 Include 文件。使用 Visual Studio . NET，可以使用将控件拖到页面设计器的方式，将服务器控件拖到用户控件设计器上，从而轻松地构建用户控件。

ASP. NET 自定义控件实质是一个从 System. Web. UI. Control 直接或间接继承于的类，它不受 Visual Studio . NET 中图形工具的支持。可以通过覆盖 System. Web. UI. Control 的 CreateChildControls 方法，往自定义控件中添加服务器控件，定义其属性和事件。然后覆盖 Control 类的 Render 方法（直接继承于 Control 类）或覆盖 WebControl 类的 RenderContent 方法（继承于 WebControl 类），编写自己的代码来发出 html。

Web Part 是 SPS2003 网站上的基本单元，它的概念类似于 ASP. NET 自定义控件，自己开发的 Web Part 将从 Microsoft. SharePoint. Web PartPages. WebPart 类继承，也是间接的继承了 System. Web. UI. Control 类。一样可以通过覆盖 System. Web. UI. Control 的 CreateChildControls 方法，添加服务器控件，定义其属性和事件。然后覆盖 WebPart 类的 RenderWebPart 方法，编写自己的代码发出 html。

Web Part 可以通过添加子控件的方式将用户控件和自身相联系，需要定义的大量用户界面和业务事件在用户控件中完成，Web Part 起着选择用户控件、编辑用户控件和传递用户控件数据的作用。

3 用户控件包装器的具体实现

3.1 用户控件的载入和属性编辑

用户控件的载入和属性编辑都借助了 Web Part 的属性和工具面板。Web Part 的属性分为默认属性和自定义属性。默认属性对 Web Part 的外观（如标题，

高度，宽度），布局（如所在 Web Part 区域，显示的次序）和更高级的控制（如是否允许关闭，是否允许区域更改及选择访问 Web Part 的群体）进行了设置，是 Web Part 自带的属性。自定义属性是用户自己定义的属性，便于更灵活的编辑 Web Part。

工具面板由不同的 Tool Part 组成。

默认属性对应 WebPartToolPart 类，自定义属性对应 CustomPropertyToolPart 类，这两个类都继承于 ToolPart 类。WebPart 类的 GetToolParts 方法决定将哪些 Tool Part 显示在工具面板里。Web Part 框架默认在该方法中将这两个类的实例写入 ToolPart 数组中，这两个属性将被工具面板中对应的 Tool Part 所编辑。同理，创建继承于 ToolPart 的类，就可以在工具面板中很好的控制 Web Part 中除了属性的内容。

3.1.1 用户控件的载入

用户控件的载入通过 Web Part 的自定义属性和工具面板实现。主要步骤如下：

（1）将用户控件对应的 dll 文件放入 SPS2003 的 bin 目录下，以待执行；

（2）创建 WebPart 的子类；

（3）在子类中添加一个自定义属性，用于保存载入的用户控件的完整路径。

（4）创建 ToolPart 的子类；

（5）将 ascx 文件放在 SPS2003 的某个虚拟目录下，在 ToolPart 子类中实现从该虚拟目录获取所有用户控件信息的方法；

（6）在 ToolPart 子类中实现一个返回一个字符串的方法，该字符串用于创建一个项值为用户控件完整路径，项的文本为用户控件名称或描述的下拉列表框；

（7）ToolPart 的子类覆盖虚方法 RenderToolPart（HtmlTextWriter），将第 7 步得到的字符串传入并被 HtmlTextWriter 对象写到浏览器上，用于在工具面板中显示包含所有用户控件的下拉列表框；

（8）ToolPart 的子类覆盖虚方法 ApplyChanges（该虚方法用于用户点击工具面板中的"确定"或"应用"按钮时发生，将相应 Tool Part 中的值作编辑）将通过表单形式提交到服务器端的下拉列表框的选中值传给相应 Web Part 的保存用户控件路径的自定义属性，通过该属性载入用户控件；

（9）WebPart 的子类覆盖虚方法 GetToolParts，在该方法中返回的 ToolPart 数组中加入第 4 步创建的类的实例；

（10）WebPart 的子类覆盖虚方法 CreateChildControls，将载入的用户控件作为 Web Part 的子控件加入；

（11）WebPart 的子类覆盖虚方法 RenderWebPart，通过 RenderControl 方法将该用户控件呈现到浏览器上。

3.1.2 用户控件属性的编辑

当载入用户控件后，可以将用户控件属性映射到工具面板的一个 Tool Part 上，通过 Tool Part 来编辑用户控件属性。主要步骤如下：

（1）创建 ToolPart 的另一个子类，添加一个自定义属性，将载入的用户控件传给它；

（2）WebPart 的子类覆盖 WebPart 的虚方法 GetToolParts，在该方法中返回的 ToolPart 数组中加入第 1 步创建的类的实例；

（3）创建一个编辑不同类型属性的基类，将载入的用户控件及及时的 http 请求和要求编辑的属性传给它。在该类的子类中具体实现不同类型属性的编辑。基类中创建一个返回一个字符串的抽象方法，该字符串是编辑属性的 html 控件的 html 标记的字符串形式，在子类中对该方法进行具体实现；

（4）创建一个编辑用户控件所有属性的类，该类利用第 3 步创建的类，逐一将编辑用户控件属性的 html 控件封装进一个 HtmlTable 中；

（5）第 1 步创建的类覆盖 ToolPart 的虚方法 RenderToolPart，利用第 4 步创建的类将对应用户控件属性信息的一个 HtmlTable 呈现到工具面板当中；

以上步骤将用户控件的属性映射到工具面板的一个 Tool Part 当中，以下步骤将实现通过 Tool Part 编辑用户控件的属性。

（6）在 WebPart 的子类中添加一个自定义属性保存用户控件的所有属性值；

（7）在 WebPart 的子类中实现一个由外部传入的属性新值更新保存属性值的自定义属性的方法；

（8）在 WebPart 的子类中实现一个利用上述自定义属性更新用户控件属性的方法；

（9）在编辑不同类型属性的基类中创建一个返回 Object 类型的抽象函数，该返回值代表以表单形式提交到服务器端的编辑属性的 html 控件的值，该抽象函

数在子类中得到具体的实现；

（10）第 1 步创建的类覆盖 ToolPart 的虚方法 ApplyChanges，利用第 9 步创建的函数得到提交到服务器端的 html 控件值，将该值传给 WebPart 的子类更新保存属性值的自定义属性，然后利用该自定义属性更新用户控件的属性值。

3.2 用户控件的编辑

菜单是 Web Part 的一个重要组成部分，菜单的充分利用可以对 Web Part 的内容进行方便的编辑。Web Part 自带的菜单可以实现对 Web Part 进行有效的编辑。这里介绍利用菜单实现对用户控件的复制和粘贴。

确定一个用户控件的完整信息需要得到用户控件的路径及用户控件的所有属性。可以考虑将这两者复制到一个"剪切板"上。类的静态成员在该类的所有实例里拥有一样的值，巧妙地起到"剪切板"的作用，基于这个思想，在 WebPart 的子类中添加两个静态域，用于保存用户控件的信息。

复制过程：

（1）添加复制菜单及相应菜单的服务器端函数；

（2）在函数中将 WebPart 子类中的自定义属性的值赋予添加的两个静态域；

粘贴过程：

（1）添加粘贴菜单及相应菜单的服务器端函数；

（2）在函数中将两个静态域的值赋予 WebPart 子类的自定义属性；

（3）根据保存用户控件路径的自定义属性添加用户控件；

（4）根据保存用户控件属性的自定义属性赋予用户控件新的属性值。

（5）保存自定义属性的值，以便再次加载页面时用户控件的状态得以保留；

3.3 用户控件之间的数值传递

3.3.1 Web Part 之间的数值传递

用户控件之间的数值传递依赖于 Web Part 之间的数值传递。Web Part 之间的数值通过实现 Web Part 框架提供的 6 对接口之一得以传递，这 6 对接口分别是：

表 1　Web Part 的六对接口

连接的接口对	描述
ICellProvider，ICellConsumer	由实现 ICellProvider 的对象提供一个 Object 类型的单一值给实现 ICellConsumer 的对象，实现 ICellConsumer 的对象在接收值之前可以向实现 ICellProvider 的对象提供 String 类型的初始信息，实现 ICellProvider 的对象也可以在发送值之前向实现 ICellConsumer 的对象提供 String 类型的初始信息
IRowProvider，IRowConsumer	由实现 IRowProvider 的对象提供一个 DataRow 类型的数组给 IRowConsumer，实现 IRowProvider 的对象在传递值之前可以向实现 IRowConsumer 的对象提供 String 类型的初始信息
IListProvider，IListConsumer	由实现 IListProvider 的对象提供一个 DataTable 类型的数给 IListConsumer 的对象，实现 IRow Provider 的对象在传递值之前可以向实现 IRowConsumer 的对象提供 String 类型的初始信息。
IFilterProvider，IFilterConsumer	提供或者消费一个 String 类型的过滤值的接口对. 例如，SharePoint 列表实现了 IRowProvider，IListProvider ，IFilterConsumer. 那么两个不同的列表能够互相连接，并且一个列表可以过滤另一个列表的内容.
IParametersInProvider，IParametersInConsumer	实现 IParametersInProvider 接口的对象可以向 IParametersInConsumer 的对象提供任意组的参数值，值的内容由 String 类型组成. 实现 IParametersInConsumer 的对象在接受参数之前可以向实现 IParametersInProvider 的对象提供所需参数的初始信息
IParametersOutProvider，IParametersOutConsumer	实现 IParametersOutProvider 接口的对象可以向 IParametersOutConsumer 的对象提供任意组的参数值，值得内容由 string 类型组成. 实现 IParametersOutProvider 的对象在发送参数之前可以向实现 IParametersOutConsumer 的对象提供所需参数的初始信息

Web Part 连接的设计和 SharePoint 的对象模型有紧密的联系。但是，从本质上说，提供的数据类型分为 Object 和 String 两大类（DataRow 和 DataTable 实际上也就是 Object 类型的数组组成）；提供的相关信息都是 String 类型，方向要么由数据者提供向数据者接收提前发送，要么由数据者接收向数据者提供者提前发送。用户控件对数据交流的最大要求是既可以传递数据，又可以接收数据，并且可以是任意类型的数据类型。

一个 Web Part 可以实现多个接口以实现既传递数据又接受数据的功能，但是两个 Web Part 之间不能既提供数据给对方，又从对方接受数据，这样将形成闭环。可以考虑的方案是传递数据而接收数据的初始化信息。综上因素，ICellProvider 和 ICellConsumer 是不错的选择，因为传递的数据是 Object 类型，而接收方可以提前传递初始化信息给发送方。主要实现步骤如下：

（1）创建供用户控件待以实现的数据提供接口和数据接收接口；

（2）创建两个类，分别用于实现 ICellProvder 和 ICellConsumer 接口；

（3）在 WebPart 的子类中，添加第 2 步创建的类的对象作为域成员；

（4）在 WebPart 的子类中，覆盖虚方法 EnsureInterfaces，根据包装的用户控件实现的接口类型，注册 ICellProvider 接口或 ICellConsumer 接口；

（5）在 WebPart 的子类中，覆盖虚方法 CanRunAt，指明连接的位置在服务器端还是客户端；

（6）在 WebPart 的子类中，覆盖虚方法 PartCommunicationConnect，该方法被 Web Part 框架用来通知 Web Part 已被连接；

（7）在 WebPart 的子类中，覆盖虚方法 PartCommunicationInit，该方法被 Web Part 框架用来传送初始化信息，注册了 ICellConsumer 的 Web Part 可以从用户控件得到要传送的初始化信息在实现 ICellConsumer 的域成员中进行传送，注册了 ICellProvider 的 Web Part 也可以从用户控件得到要传送的初始化信息在实现 ICellProvider 的域成员中进行传送；

（8）在 WebPart 的子类中，覆盖虚方法 PartCommunicationMain，注册了 ICellProvider 的 Web Part 可以将实现 ICellProvider 的域成员在此方法中得到的传递过来的初始化信息转发给用户控件；注册了 ICellConsumer 的 Web Part 也可以将实现 ICellConsumer 的域成员在此方法中得到的传递过来的初始化信息转发给用户控件；同时，注册了 ICellProvider 接口的 Web Part 可以在此时接收用户控件传来的数据，利用实现 ICellProvider 的域成员发送数据给注册 ICellConsumer 接口的 Web Part；

（9）在 WebPart 的子类中，覆盖虚方法 RenderWebPart，注册了 ICellConsumer 接口的 Web Part 的实现 ICellConsumer 的域成员在此方法中得到注册了 ICellProvider 的 Web Part 传来的数据，注册了 ICellConsumer 接口的 Web Part 可以将数据传递给包装的用户控件，将用户控件呈现在浏览器上；

这些虚方法都是 Web Part 框架依次调用，次序和步骤顺序一致。

4 结束语

用户控件包装器是巧妙的利用 Web Part 本身的功能将 Web Part 的缺陷加以克服。它的实现给 Web Part 的开发带来极大的便利，ASP. NET 程序员根本就不需要理解 Web Part 的开发原理，就可以利用以往的编程思想快速地开发自己需要的 Web Part，进而使得 SPS2003 的门户网站开发和设计事半功倍。

参考文献

1 计算机世界方案评析实验室. 十大协同应用解决方案[N]，计算机世界，2005－05－16(C08).

2 Andy Baron. 面向开发人员的 Web 部件介绍[EB/OL]. http://www.microsoft.com/china/MSDN/library/Office/sharepoint/SP_northwindwebparts.mspx, 2004－07－09.

3 Nilly Banerjee. 修改 Web 部件和 Web 部件页的用户界面[EB/OL]. http://www.microsoft.com/china/MSDN/library/Office/sharepoint/SPmodifyingui.mspx, 2004－07－09.

4 Patrick Tisseghem , Jan Tielens. Building Web Parts the Smart Way [EB/OL]. http://www.microsoft.com/belux/msdn/nl/community/columns/u2u/smartpart.mspx.

5 Microsoft Corporation. Microsoft SharePoint Products and Technologies 2003 Software Development Kit [z]. USA:Microsoft Corporation,2005.