第19卷 第5期 云南农业大学学报 Vol. 19 No. 5
2004年 10月 Journal of Yunnan Agricultural University Oct. 2004

ASP.NET的可复用代码*

段美英[1],鲁绍坤[1],邱苑梅[2],许 亮[3]
(1. 云南农业大学基础与信息工程学院,云南 昆明 650201;
2. 云南农业大学水利水电与建筑学院,云南 昆明 650201; 3. 昆明理工大学管理与经济学院,云南 昆明 650093)

摘要: ASP.NET的可复用代码用于构建动态网页。介绍了ASP.NET的可复用代码,阐述了ASP.NET的4种可复用代码方式,包括它们的定义、如何创建可复用代码和如何运用它们,从不同角度对4种可复用代码方式进行了比较,示例说明了用可复用代码构建ASP.NET应用程序。ASP.NET的4种可复用代码协同工作,使编程模块化,降低了复杂性,保证了应用程序的质量和可维护性。

关键词: ASP.NET; 可复用代码; 用户控件; 后台编码; .NET程序集; 定制控件
中图分类号: TP 311.52 **文献标识码**: A **文章编号**: 1004-390X(2004)05-0600-04

Reusable Code in ASP.NET

DUAN Mei-ying[1], LU Shao-kun[1], QIU Yuan-mei[2], XU Liang[3]
(1. College of Fundamental and Information Engineering, Y A U, Kunming 650201, China;
2. College of Water Resources Hydraulic Power and Architectural, Y A U, Kunming 650201, China;
3. College of Management and Economics, Kunming University of Science and Technology, Kunming 650093, China)

Abstract: Reusable code in ASP.NET is used for constructing dynamic web pages. This paper introduces reusable code in ASP.NET. It expounds four kinds of methods of reusable codes, including their definitions, how to create them and how to quote them in web form respectively. Then this article makes a comparison between them from different angles. Finally it illustrates how to build an ASP.NET application using reusable codes. By co-operation with four kinds of reusable codes, the application is made of modules and has less complexity, high quality and easy maintenance.

Key words: ASP.NET; reusable code; user control; code behind; NET component assembly; custom control

万维网(World Wide Web)和因特网(Internet)两大技术把人们带入了信息时代,越来越多的公司将电子商务集成到其业务策略中,网络已经成为人们生活的一部分,基于Web的系统和应用正爆炸式地增长。面对日趋复杂的Web应用,如何快速地构造一个可靠、美观、可维护的高质量Web网站? ASP.NET提供了一个很好的解决方案。

ASP.NET是Microsoft公司的ASP(动态服务器页面)和.NET两项技术的结合,它具有功能强大的服务器技术,用于创建动态WEB页。ASP.NET优势在于不仅提供了大量的、可复用的预定义控件,用户还可以编写可复用代码[1]。例如,传统的ASP,所有的内容全部放在一个文件中,代码与显示内容不能分离,应用程序复杂,难以扩展和维护。而ASP.NET服务器控件为ASP.NET应用程序实现代码复用提供了一个很好机制,使得动态生成的ASP.NET代码从HTML和各种控件的内容中分离出来成为可能,可复用代码放到可从.ASPX页面中引用的一个独立组件上。

ASP.NET中,可复用代码是封装了有用功能组的完全独立的一组文件,用于满足Web应用程序的体系结构。Web页面通过接口,组装这些可复

* 收稿日期: 2004-05-19
作者简介: 段美英(1967-),女,云南昭通人,昆明理工大学在读硕士研究生,主要从事管理信息系统的研究。

用代码,程序结构清晰,简化了编程工作,减少了需要编写的代码数量,降低了出错率,使开发应用程序的速度加快;另外,最终的应用程序较小,可能出现的问题也较少,增强了代码的易维护性。

1　ASP.NET 的可复用代码方式

ASP.NET 可复用代码的方式有 4 种:用户控件、后台编码、.NET 程序集和定制控件。

1.1　用户控件

用户控件是封装到可复用控件中的 Web 窗体,它包含了许多页面都需要的代码块;或者说用户控件将可复用的代码或内容创建为独立的 ASP.NET 控件,然后可以在其它页面中使用这些控件。例如,一个公司的 Web 站点上有多个页面,每个页面都有相同的徽标和标题栏,只需创建一个用户控件,其中包含共享的徽标和标题栏,那么,每个页面就可以引用这个用户控件,实现代码的复用。

创建用户控件与创建 Web Form 页面基本相同,但用户控件中没有 < HTML > , < BODY > 和 < FORM > 标签;用户控件可以包含客户端脚本、HTML 元素、ASP.NET 代码和其它服务器控件,保存为扩展名为 .ascx 文件。

例如,创建了一个标题用户控件 header.ascx,要在 ASPX 文件中引用它,必须在 ASPX 文件的顶部且任何 HTML 代码之前,用@Register 指令进行声明:

< %@Register TagPrefix = “MyTitle” TagName = “TitleControl” Src = “header.ascx”% >

TagPrefix 属性指出控件的前缀是 MyTitle,Tagname 属性指出控件的名称为 TitleControl,Src 属性指出控件的源文件为 header.ascx。

然后在 HtmlForm 服务控件的开始标记与结束标记之间(< form runat = “server” > < /form >)引用该用户控件:

< MyTitle:TitleControl id = “MyTitleControl” runat = “server” >

1.2　后台编码

后台编码是把 ASP.NET 页面的所有脚本块从 ASPX 文件中分离出来,存储为 vb 或 c # 等.NET 语言的代码文件;它让 ASPX 文件只包含指令和布局,实现了 ASP.NET 页面的代码与内容在物理上的分离,使代码模块化[2]。比如,对于复杂的 Web 站点,通过后台编码,使设计人员专门负责维护外观内容,而开发人员注重编写代码,负责应用程序的功能和逻辑,这样不仅考虑了外观效果,还保证长期稳定的可维护性。

后台编码的原则是为自己的代码创建一个新类,该类从 ASP.NET 的 Page 对象上继承,能够访问页面代码并与回送结构相互交互;ASP.NET 页面使用一个页面指令来继承新创建的类。

例如,创建后台编码文件 My_CodeBehind.vb,首先导入重要的命名空间,用于访问 ASP.NET 页面上所包含的类,然后定义一个继承于 Page 的类:

```
Import System
Import System.Web.UI
Import System.Web.UI.WebControls
Public Class MyCodeBehind:Inherits Page
事件代码
End Class
```

在 ASPX 文件的顶部且任何 HTML 代码之前加入以下指令,该页面就可以引用后台编码文件 MyCodeBehind.vb:

< % @ Page Inherits = “MyCodeBehind” Src = “My_CodeBehind.vb”% >

后台编码也可应用于用户控件,与 ASP.NET 页面的后台编码不同之处仅在于,后台编码文件中创建的类继承于 UserControl 类而非 Page 类即 Inherits UserControl,用户控件引用后台编码的指令是 @Control 并非@Page.

1.3　.NET 程序集

.NET 程序集是封装到一个已编译文件中相关类和接口定义的组合,所有的.NET 应用程序可以编程访问这些类和接口。程序集中可以放入任何内容,包含所需要的各种功能;用任何.NET 兼容语言编写,并预编译为 .dll 文件。例如,多个页面或应用程序要复用后台编码文件包含的功能,就可以把后台编码文件编译成程序集。

创建程序集分为两步,首先创建具有命名空间的组件,然后编译组件并保存到 Web 应用程序的 bin 目录下。其它页面和应用程序导入该命名空间,就可以访问程序集中包含的所有类。

例如,首先创建组件 My_Components.vb,其命名空间 MyComponents 中定义了类 GetBooks:

```
Namespace MyComponents
Public Class GetBooks
类代码
End Class
End Namespace
```

然后通过特定的命令,将组件 My_Components.vb 编译为程序集 My_Components.dll:

vbc /t:library /out:bin/My-Components.dll My-Components.vb

最后,在 ASP.NET 页面中导入命名空间 My-Components,就可以访问类 GetBooks 的属性和方法:

<%@Import Namespace="MyComponents"%>

另外,如果要访问的程序集不在 Web 应用程序根目录的 bin 目录下,而在其它的地方,则使用配置文件 web.config,加入引用的程序集,就可以引导页面找到所需组件。

1.4 定制控件

ASP.NET 页面主要是围绕服务器而展开的,除了 ASP.NET 提供的 HTML 和 Web 表单两类重要的内置服务器控件外,还可以灵活地自定义服务器控件即创建定制控件。定制控件是直接或间接从 System..Web.UI.Control 派生的已编译好的类,有一个可见的交互式用户界面;但 ASPX 页面中没有包含 UI 代码,其显示方式完全通过编程来控制[3]。

编写一个简单的定制控件,首先导入重要的命名空间,然后创建具有命名空间的类,该类继承于 System..Web.UI.Control,编程来指定其显示方式,最后编译该控件。

例如,创建一个 Welcome.vb 控件:

```
Import System
Import System.Web.UI
Namespace CustomControls
Public class Welcome:Inherits Control
显示代码
End Class
End Namespace
```

编译文件:

vbc/t:library /out:bin/Welcome.dll Welcome.vb /r:System.dll,System.Web.dll

在 ASPX 页上使用 Welcome 定制控件时,用 Register 页指令命名空间为 CustomControls,前缀为 Custom,并为 ASP.NET 提供包含该控件的程序集名称 Welcome:

```
<%@Register TagPrefix="Custom" Namespace
="CustomControls"
Assembly="Welcomel">
<html>
<body>
<Custom:MyControl runat="server"/>
</body>
</html>
```

2 可复用代码方式的比较

从本质上看,用户控件是一个 Web 页面上划分出来的一部分,与 Web 窗体页不同的是,不能独立地请求用户控件,用户控件必须包括在 Web 窗体页内才能使用;但通过用户控件的属性,可以逐步脱离控制它的页面,改变其行为。后台编码把页面逻辑封装到后台编译文件中,隐藏了处理 Web 窗体的代码,本质上是一个从 Page 对象上继承的类,实现了 Web 页面上的脚本和内容的成功分离。.NET 程序集实质上是用组件代替脚本代码,为协同工作而生成编译好的类和接口集合,封装了业务逻辑,具有按逻辑组合起来再放入物理文件的功能。定制控件是一种自定义的可视化控件,用于生成可浏览的输出结果,但需要编写许多代码;通过对某种控件进行改造,使它具有自己所希望的外形或者结果,而不是它缺省的方式运行[4]。

从编程模式上看,用户控件使用的编程模型与 ASP.NET 页相同,保存为一个文本文件 .ascx 页。其他组件涉及面向对象的编程,使用的语言以公共语言运行库为目标,如 Visual Basic .NET 和 C# 等;但.NET 程序集和定制控件是已编译好的类,后台编码只有当用户第一次浏览 Web 页时,才被编译成一个动态链接库(.dll)文件。

从可视性看,用户控件和定制控件可呈现用户界面,对模板的支持使主程序代码大大简化;后台编码和.NET 程序集是代码,无用户界面;4 种复用方式中只有定制控件有可见的外观,用户控件的界面实际上是通过服务器控件表现出来的,它本身没有可视外观。

从适用场景上看,用户控件适用于页面上重复使用的元素,如标题、菜单、注册控件等;对重复元素进行封装,使之成为一个可以嵌入另一页面的组件,呈现出公共用户界面。后台编码适用于将实现应用程序逻辑的服务器端代码隔离在后台,使用户界面上的代码与内容相分离。.NET 程序集允许具有类似需求的不同应用程序和 Web 页面重用一组共同的类和接口,实现一组特定的功能。定制控件用于显示具有自己特色的风格用户界面,能完成自己想完成的任何功能,可使复杂的页面开发变得容易。

从复杂性上看,用户控件和后台编码这两种组

件复用方式相当简单,仅是把页面的一部分代码和服务器端的脚本代码分别移到不同的地方,而.NET 程序集和定制控件是可复用组件的高级方式,它们是已编译好的类。

从可复用性上看,用户控件能被多个 Web 页面共用,由于每个页面提供给它的属性不同,呈现出的界面有可能不同;后台编码只能对应一个 Web 页面或一个用户控件,若要被共享,必须把后台编码编译成.NET 程序集; .NET 程序集和定制控件能被其它的页面和应用程序复用,复用功能非常强,其中,定制控件复用的粒度最细,因此,灵活性和可复用性是 4 种复用方式中最强的。

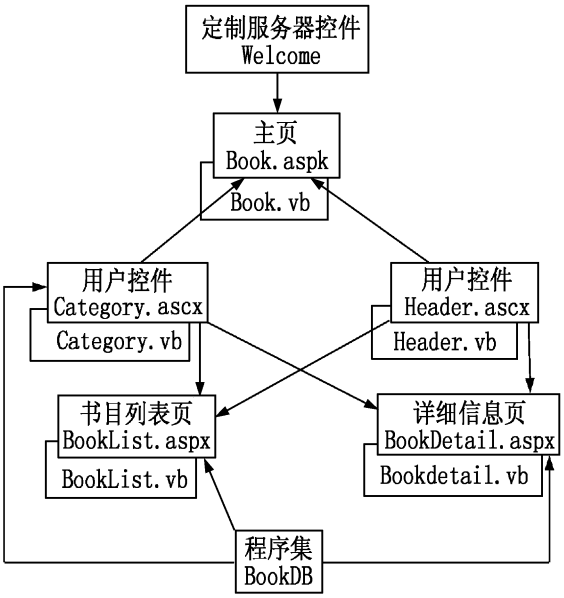

图 1　网上书展结构图

Fig. 1　Structural Drawing of Book Display on Web

3　用可复用代码构建ASP.NET 应用程序实例

以一个简单的网上书展为例,来说明 APS.NET 中可复用代码的应用。

该网上书展的目的是让那些对图书感兴趣的人,不必亲自到书展现场,只需上网轻轻地按动鼠标,就可浏览到展出的书目以及它们的特定信息。其应用程序包括 3 个 Web 页面。第 1 个 Web 页面是主页 Book.aspx,此页顶部是书展的徽标和标识,左侧是书的分类目录,剩下的部分是欢迎信息和书展的简单介绍。鼠标单击左侧某个分类目录时,调用第 2 个 Web 页面 BookList.aspx,显示所有指定类别的书目列表;当选定一本书后,调用第 3 个 Web 页面 BookDetail.aspx,显示该书的详细信息。后两个页面的顶部和左侧所呈现出的内容与主页一致。

通过以上分析,可用 Header.ascx 和 Category.ascx 两个用户控件分别显示顶部和左侧的内容,这样保持了三个 Web 页的界面一致性。主页要显示的欢迎信息和书展简介,用定制服务器控件 Welcome 来实现,满足具有特殊显示格式的界面;BookList.aspx 页面中属于某种分类的书目列表和 BookDetail.aspx 页面中具体某一本书的细节内容则由不同格式的服务器控件 Datalist 来显示。为了完成应用程序访问数据和执行业务逻辑的功能,创建组件 BookDB 并编译成程序集,实现检索所有书目类别、指定类别的书目列表和特定单本书的具体信息等任务。因此,用可复用代码组装网上书展应用程序的结构图如图 1 所示。

从图 1 中可以看出,任何 ASP.NET 应用程序可由 1 个或多个 Web 页构成,而每个 Web 页又由 ASPX 文件及其相关的可复用代码组成;每个 ASPX 页面都有一个后台编码文件,每个用户控件.ascx 也都有一个后台编码文件,户控件可以用于任何一个 ASP. NET 页面;存储在程序集中的业务逻辑组件,用于任何一个 ASP.NET 页面、用户控件或定制服务器控件,任意用户控件或 ASP.NET 页面可使用定制服务器控件[1]。ASP.NET 4 种可复用代码的方式,虽然它们各有其用途,但是在编写 APS.NET 应用程序时,它们被当作一个密不可分的整体。

4　结束语

ASP.NET 用于创建动态 WEB 页,而用户控件、后台编码、.NET 程序集和定制服务器控协同工作,为 ASP.NET 应用程序提供了强大的可复用代码和简洁的编程模式,提高了编程效率和易维护性。

[参　考　文　献]

[1] CHRIS GOODE, JOHN KAUFFMAN. ASP.NET 1.0 入门经典[M]. 康博等译. 北京: 清华大学出版社, 2002.

[2] DINO ESPOSITO. Reusability in ASP. NET[J/OL]. http://www.msdn.com/msdnmag/issues/01/08/cutting

[3] RICHARD ANDERSON, BRIAN FRANCIS. ASP.NET 高级编程[M]. 王毅等译. 北京:清华大学出版社,2002.

[4] 天极网新技术研究室. ASP.NET 完全入门[M]. 重庆: 重庆出版社, 2001.