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	ASP.NET2.0＋VS2005利器大评析之优点篇		
来源：	[朱先忠编译]	浏览：	[414]

ASP.NET 2.0与Visual Studio 2005正式上市至今已经有好几个月了。随着时间的一天天向前推移，如今越来越多的人能够使用到这一产品。本文作者基于对该产品的试用试图从个人角度来对这一产品的优点与不足作出评析，仅供参考。

一、 引言

说实在的，我对ASP.NET 2.0与Visual Studio 2005之间的关系有些喜欢也有些讨厌；但是，我最终还是决定把我的大多数内部应用程序迁移到2.0并且决不后悔。基本上，ASP.NET 2.0中存在太多的新的特征使我的生活变得更为轻松，并且从此以后再也不会返回到以前的1.1版本中编程了。

到目前为止，这个平台一直工作良好。我已经发现了其中存在的许多改进能够大量地降低编码的复杂性和编码数量，在性能方面大约提高了10～20%，并且减少了内存需求量（这对于大型应用程序而言是十分重要的），但是对我的工作来说几乎没有太大的作用。

把2.0特征加入到现有应用程序中绝对不是一夜之间的事情。但是，随着我逐渐地习惯快速地把大量的2.0特征加入到我的新式应用程序中，我的最后感觉是：如果再回到ASP.NET 1.1和VS 2003的话，那将是一个很大的后退。

二、 优点评析

下面，让我们来逐步剖析这个新产品中的主要变化，首先来看一下它的优点。

(一) Visual Studio 2005基于文件的工程开发

现在，在Visual Studio 2005中，你能够把一个目录作为一个web工程来打开，这是一种相当不错的改进。在我的开发机器上，我可能有50个不同的web工程。使用以前的VS2003，要把所有这些作为IIS中的虚拟目录加以配置和维护并且使工程实现正确地引用是令人相当头疼的事情。你不这样认为吗？你是否想把某些工程移动到一台新机器上？在VS2005中，你只需要简单地指向一个目录就可以打开工程。你完全可以使用本地的Web服务器构建方式来运行应用程序，这样以来就免除了配置Web服务器的需要。

这个特征特别适合于共享示例的开发者—任何想检查一个示例web应用程序的开发人员都不必经受基于IIS进行配置的痛苦。现在，借助于基于文件的工程，你能够—至少在开发场所下—实现真正的“xCopy”工程。这个特征相当伟大，但是也不无缺点（一会儿后我们会详及）。

【另注】我接触到的每一位都喜欢构建到Visual Studio内部的Cassini web服务器。当然，我也喜欢，因为它极大地简化了许多问题的处理。然而，有关它的使用也存在一些缺点。主要是在使用过程中应当避免Cassini与IIS之间的相互干扰。例如，Cassini能够把所有的请求传递给ASP.NET而忽略扩展内容。如果你拥有处理特定的文件类型的定制的处理器(例如，动态地构建Excel报告，等等)，那么，你必须记住，当发布你的应用程序时，你要在IIS中为扩展内容建立定制的映射；否则的话，IIS不会把请求传递给ASP.NET。我接触过许多朋友在发布时花费大量的时间来解决他们的应用程序中的问题，因为他们在开发过程中从不担心Cassini中的配置设置问题。

(二) 母版页面

现在，你可以定义一个能够在你的应用程序中重用的母版（Master）页模板。使用这个功能能够节约你大量的开发时间。事实上，在2.0版本出现以前，已经存在基于ASP.NET 1.x版本的这种概念，但是对于我来说，吸引我的最关键的特征在于，Visual Studio提供了对它的可视化支持。这可以使你看到母版的布局，其中ContentPlaceholders可以应用于每一个页面中以提供页面级内容。

除了设计器提供的重要的可视化方面外，母版页模板还提供了一种良好的方式来把彼此相关的可重用的代码联系到一起。母版页面的目的是，把以前需要使用若干用户控件(例如，Header，Footer和Sidebar)才能实现的功能融合到一起，从而使它们能够比以前更为有效地实现逻辑分离。

【另注】你还能够在运行时刻动态地改变母版页面，从而实现更大的灵活性。这一支持使用户能够改变一个应用程序的整体外观感觉；而且这种效果是仅凭借切换层叠式样表所无法实现的。

(三) 用户控件可视化描述

说实在的，我非常希望自己在设计时就能看到整个页面的样子。就象母版页面一样，现在，Visual Studio 2005能够在Web表单编辑器内显示一个生成的用户控件。不再象是以前的老式的、非描述性的灰色的方框加上一个控件名，现在，你能够在设计器内得到一个全面生成的恰当到位的控件。双击它，则VS就能把你导航到用户控件设计器。在我的开发中，我一般不会大量地使用用户控件，而是使用母版页面来替换我的许多现有的控件，但是我发现这种用户控件可视化描述使设计模式更为有用了。对于我的现有1.1版本的应用程序来说，尤其如此—我的这些程序中通常仍然使用这样的控件来表达页面的页眉，侧栏和页脚。

【另注】完全自动地生成用户控件极大地节约了开发时间。当然，我还需要花费不少的时间从IDE到一个浏览器来回切换以观察用户控件最终生成的样子。仅此而已。

(四) 泛型

不错，这并非是一个ASP.NET特有的特征，但是.NET 2.0中泛型的引入大大丰富了代码的编写。以前，在创建定制集合时，我常常非常小心；坦诚地说，反反复复地从CollectionBase进行派生然后重新实现相同的代码是一件非常折磨人的工

作。对于定制控件开发，特别在ASP.NET中开发时，我发现当你需要集合特性时使用泛型集合效果相当好。

你只需简单地使用列表或一个特定的泛型集合，然后把它作为该控件的一个属性—问题就这么简单！Visual Studio能够看到这个集合；并且，在大多数情况下，它还能够为你提供相应的集合编辑器。通过使用泛型列表，你可以很容易地使用强类型化列表来代替许多基于ArrayList的列表，这往往使编码更为清晰。

最后，在业务对象内使用动态的类型替换消除了对令人“胆战心惊”的初始化编码（以前，在每一个业务对象中都要进行这种初始化以指定哪个实体类型与之相关联）的需要。在泛型出现以前，往往需要借助于一个小型编码代理来把业务对象和实体绑定到一起。现在，有了泛型类型后，不再需要这样的编码，而代之以一个泛型类型参数。此后，所有的类级代码就能够使用泛型类描述在运行时刻自动地生成正确的类型。借助于一个类型化参数和一组父类级方法，现在再也不需要从我的所有业务对象中剪切和粘贴大量的代码。其实，还存在许多使用泛型的场所；而且如今，我发现不使用泛型类型很多问题变得十分棘手，特别是在处理与集合相关的内容时。

【另注】泛型将会被广泛应用于集合及业务对象操作方面，而且，你也可以在页面基类和用户控件开发中从中获益。最近，我在网上看到有人构建一个泛型基页面实现自动地加载业务对象数据并建立相应的Ajax回调机制以便更新这些对象。你看，以前在每一个新页面中实现起来如此头疼的事情一下变得如此简单了！

(五) 支持嵌入式资源

我比较喜欢把大量的定制控件用于我自己的应用程序中。经常情况下，这些控件都会依赖于特定的资源，例如图像，CSS文件，XML资源等等。此时，任何这些控件的用户必须记住要在他们的应用程序中发布相应的文件。如今，在ASP.NET开发中，你可以容易地把需要的Web资源嵌入到一个工程中，然后经由一个ASP.NET生成动态的URL来存取它们。为此，你只需要简单地把[WebResource]属性添加到你的控件的AssemblyInfo文件中，然后使用Page.ClientScript.GetWebResourceUrl来检索包含这些资源内容的URL即可。

(六) Visual StudioASP.NET代码编辑器

Visual Studio 2005代码编辑器比2003版本前进了一大步。我认为，最重要的新“特征”在于，新的编辑器不会自动地“打乱”我的代码格式，除非我重新格式化文档。例如，我想让我的内容按我喜欢的方式进行组织，然而，当我使用VS2003时这却成了一个问题—无论何时把新的控件添加到页面系统都会重新格式化HTML。在VS2005中，编辑器在大多数情况下会保留用户自己的代码格式，并且还会提供一种更好的处理—把控件标记插入到代码中。

一个真正提高生产效率的改进是，在新的HTML编辑器中引入了智能感知技术—而且出现在每一处位置！我经常在一个页面内嵌入表达式，而智能感知意味着它会帮助我避免错别字。ASP.NET 2.0还会编译页面并且检查生成的嵌入式脚本代码，以便及早地在设计时刻而不是在运行时刻才捕获HTML标记中的错误。
智能感知适合于所有的控件，包括你自己的定制控件，因此你不必再提供一种私有类型模式文件。Visual Studio能够简单地找到你的控件并且在内部管理智能感知。智能感知支持真是太好了，有时它甚至能够“超越”可视化设计器。一会儿后，你就会明白为什么这可能比你想像的更为重要。

【另注】作为一名最近才从Visual Basic转到C#的新手，我特别欣赏Visual Studio 2005提供的C#智能感知支持。在Visual Studio以前的版本中好象在对VB和C#的智能感知支持方面存在很大的差距；并且，当我分析C#代码时，我常常发现我自己十分需要有一种VB风格的智能感知帮助。现在，现在这种差距消失了，而且语言之间的切换也更为容易了。