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	用ASP.NET2.0在数据库中存储二进制文件		
来源：	[朱先忠编译]	浏览：	[696]

一、 引言

在构建数据驱动的应用程序时，经常需要捕获文本和二进制数据。这样的程序可能需要存储图像，PDF，Word文件或其它二进制数据。能够使用两种方式来存储这些二进制数据：存储在web服务器的文件系统上并添加一个对数据库中相应文件的引用；或直接存储在数据库本身。

文本数据，例如字符串，数字，日期，GUID，货币值，等等-在数据库系统中都有适当的和相应的数据类型定义。例如，在Microsoft SQL Server中，你可以使用int数据类型来存储一个整数值；而为了存储一个字符串值，你可以使用一个varchar或nvarchar类型。另外，数据库还提供了用于存储二进制数据的类型定义。在Microsoft SQL SERVER 2000及早期版本中，使用image数据类型来存储二进制数据；而在SQL SERVER 2005中，使用varbinary(MAX)数据类型。使用上面两种方式中的任何一种，这些数据类型都能够存储可达2GB大小的二进制数据。

不过，当直接把二进制数据存储在数据库时，需要增加一些额外工作来实现插入、更新和检索二进制数据。幸好，我们可以通过更高级的数据存取库-例如ADO.NET-对这种复杂的低级T-SQL操作加以抽象，从而使问题变得相当简单。然而，通过ADO.NET方式使用二进制数据与使用文本数据的确有点不同。在本文中，我们将分析如何使用ADO.NET和ASP.NET 2.0 SqlDataSource控件直接通过一个数据库来存储和检索图像文件。请接着往下阅读！

二、 把数据存储在数据库中与存储在文件系统中的比较

正如刚才介绍的，当捕获一个应用程序中的二进制数据时，该二进制数据既可以直接存储在数据库中也可以作为一个文件保存在web服务器的文件系统中-仅保持一个对数据库中文件的引用。根据我的体验，我发现大多数开发者更喜欢在文件系统中存储二进制数据，这主要基于下列原因：

· 需要较少的工作-存储和检索存储在数据库中的二进制需要更多的编码工作。而且，更新这些二进制数据也会更为容易-不需要与数据库通讯，只须直接修改文件即可！

· 指向文件的URL更为直接-正如我们将在本文中所看到的，为了提供存取存储在一个数据库中的二进制数据，我们需要创建另一个能够返回该数据的ASP.NET页面。典型地，会把相应于数据库中对应记录（返回它的二进制数据）的一个唯一的标识符传递给这个页面。结果是，为了存取该二进制数据-比方说一个上传的图像-该URL看上去如http://www.yourserver.com/ShowImage.aspx?ID=4352的形式，而如果该图像直接存储在文件系统中，URL将更为直接些-例如http://www.yourserver.com/UploadedImages/Sam.jpg。

· 为显示图像提供更好的工具支持-如果你在使用ASP.NET 2.0，那么，你可以在GridView或DetailsView控件中使用ImageField控件来显示一个图像（它的图像路径存储在数据库中）。然而，遗憾的是，这个ImageField却无法直接显示数据库中的图像数据(既然它要求查询一个外部页面并且返回相应的数据)。

· 性能-既然二进制文件存储在web服务器的文件系统而不是存储在数据库中，那么，应用程序可以访问数据库中较少的数据，从而减少了对数据库的要求，也相应地减少了存在于web和数据库服务器之间的网络拥挤。

把数据直接存储在数据库的主要优点在于，它能够使数据成为"自包含的"。既然所有的数据都包含在数据库中，那么，数据支持、数据在数据库服务器间的移动以及数据库复制等等就容易得多了，因为不存在担心复制或备份存储在文件系统中的二进制内容这样的问题。

如往常一样，至于选择哪种存储方案要具体依赖于实际的使用场所和业务需要。例如，我开发过一个客户端，其中的二进制数据必须存储在数据库中，因为它们使用的报告软件仅能够在报告中包括二进制数据-如果它来自于数据库的话。在另一种情况下，我的一个同事在开发一个工程，其中的二进制文件需要为web应用程序使用并且可经由FTP使用，这种情况很有必要把二进制数据存储在文件系统中。

三、 创建一个存储二进制数据的数据库表格

本文中的其它部分将分析一个简单的ASP.NET2.0图像画廊应用程序，我使用微软SQL Server 2005 Express Edition编写的，用于展示本文所阐述的直接从一个数据库中存储和检索二进制数据的相关概念。

这个图像画廊应用程序的数据模型包括一个表格-Pictures，其中的每一个记录对应画廊中的一幅图片。这个Pictures表格的MIMEType域中存储了上载图像(对于JPG文件是image/jpeg，对于GIF文件是image/gif，等等)的MIME类型；这里的MIME类型向浏览器指定如何生成该二进制数据。其中的ImageData列则存储了该图片实际的二进制内容。

一、 引言

在构建数据驱动的应用程序时，经常需要捕获文本和二进制数据。这样的程序可能需要存储图像，PDF，Word文件或其它二进制数据。能够使用两种方式来存储这些二进制数据：存储在web服务器的文件系统上并添加一个对数据库中相应文件的引用；或直接存储在数据库本身。

文本数据，例如字符串，数字，日期，GUID，货币值，等等-在数据库系统中都有适当的和相应的数据类型定义。例如，在Microsoft SQL Server中，你可以使用int数据类型来存储一个整数值；而为了存储一个字符串值，你可以使用一个varchar或nvarchar类型。另外，数据库还提供了用于存储二进制数据的类型定义。在Microsoft SQL SERVER 2000及早期版本中，使用image数据类型来存储二进制数据；而在SQL SERVER 2005中，使用varbinary(MAX)数据类型。使用上面两种方式中的任何一种，这些数据类型都能够存储可达2GB大小的二进制数据。

不过，当直接把二进制数据存储在数据库时，需要增加一些额外工作来实现插入、更新和检索二进制数据。幸好，我们可以通过更高级的数据存取库-例如ADO.NET-对这种复杂的低级T-SQL操作加以抽象，从而使问题变得相当简单。然而，通过ADO.NET方式使用二进制数据与使用文本数据的确有点不同。在本文中，我们将分析如何使用ADO.NET和ASP.NET 2.0 SqlDataSource控件直接通过一个数据库来存储和检索图像文件。请接着往下阅读！

二、 把数据存储在数据库中与存储在文件系统中的比较

正如刚才介绍的，当捕获一个应用程序中的二进制数据时，该二进制数据既可以直接存储在数据库中也可以作为一个文件保存在web服务器的文件系统中-仅保持一个对数据库中文件的引用。根据我的体验，我发现大多数开发者更喜欢在文件系统中存储二进制数据，这主要基于下列原因：

· 需要较少的工作-存储和检索存储在数据库中的二进制需要更多的编码工作。而且，更新这些二进制数据也会更为容易-不需要与数据库通讯，只须直接修改文件即可！

· 指向文件的URL更为直接-正如我们将在本文中所看到的，为了提供存取存储在一个数据库中的二进制数据，我们需要创建另一个能够返回该数据的ASP.NET页面。典型地，会把相应于数据库中对应记录（返回它的二进制数据）的一个唯一的标识符传递给这个页面。结果是，为了存取该二进制数据-比方说一个上传的图像-该URL看上去如http://www.yourserver.com/ShowImage.aspx?ID=4352的形式，而如果该图像直接存储在文件系统中，URL将更为直接些-例如http://www.yourserver.com/UploadedImages/Sam.jpg。

· 为显示图像提供更好的工具支持-如果你在使用ASP.NET 2.0，那么，你可以在GridView或DetailsView控件中使用ImageField控件来显示一个图像（它的图像路径存储在数据库中）。然而，遗憾的是，这个ImageField却无法直接显示数据库中的图像数据(既然它要求查询一个外部页面并且返回相应的数据)。

· 性能-既然二进制文件存储在web服务器的文件系统而不是存储在数据库中，那么，应用程序可以访问数据库中较少的数据，从而减少了对数据库的要求，也相应地减少了存在于web和数据库服务器之间的网络拥挤。

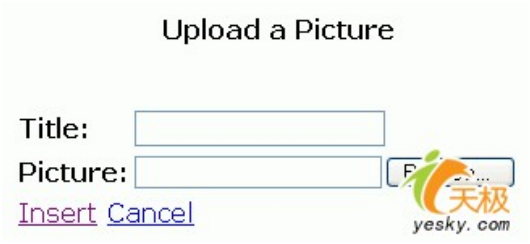
把数据直接存储在数据库的主要优点在于，它能够使数据成为"自包含的"。既然所有的数据都包含在数据库中，那么，数据支持、数据在数据库服务器间的移动以及数据库复制等等就容易得多了，因为不存在担心复制或备份存储在文件系统中的二进制内容这样的问题。

如往常一样，至于选择哪种存储方案要具体依赖于实际的使用场所和业务需要。例如，我开发过一个客户端，其中的二进制数据必须存储在数据库中，因为它们使用的报告软件仅能够在报告中包括二进制数据-如果它来自于数据库的话。在另一种情况下，我的一个同事在开发一个工程，其中的二进制文件需要为web应用程序使用并且可经由FTP使用，这种情况很有必要把二进制数据存储在文件系统中。

三、 创建一个存储二进制数据的数据库表格

本文中的其它部分将分析一个简单的ASP.NET2.0图像画廊应用程序，我使用微软SQL Server 2005 Express Edition编写的，用于展示本文所阐述的直接从一个数据库中存储和检索二进制数据的相关概念。

这个图像画廊应用程序的数据模型包括一个表格-Pictures，其中的每一个记录对应画廊中的一幅图片。这个Pictures表格的MIMEType域中存储了上载图像(对于JPG文件是image/jpeg，对于GIF文件是image/gif，等等)的MIME类型；这里的MIME类型向浏览器指定如何生成该二进制数据。其中的ImageData列则存储了该图片实际的二进制内容。



一旦用户选择了一个文件并且寄送了这个表单(例如通过点击"Insert"按钮)，那么，指定文件的二进制内容即被寄送到web服务器。从服务器端代码中，这种二进制数据通过FileUpload控件的PostedFile.InputStream属性成为可用，这正如下面的标记和代码所展示的：

```
Protected Sub btnInsert_Click(ByVal sender As Object,  ByVal e As System.EventArgs) Handles btnInsert.Click
'确保一个文件被成功上传
If UploadedFile.PostedFile Is Nothing OrElse String.IsNullOrEmpty(UploadedFile.PostedFile.FileName) OrElse
UploadedFile.PostedFile.InputStream Is Nothing Then
  ... 显示错误信息...
  Exit Sub
End If
'确保我们在操作一个JPG或者GIF文件
Dim extension As String = Path.GetExtension(UploadedFile.PostedFile.FileName).ToLower()
Dim MIMEType As String = Nothing
Select Case extension
  Case ".gif"
    MIMEType = "image/gif"
  Case ".jpg",  ".jpeg",  ".jpe"
    MIMEType = "image/jpeg"
  Case ".png"
    MIMEType = "image/png"
  Case Else
    '无效的文件类型上传
    ... 显示错误信息...
    Exit Sub
End Select

'与数据库连接并且把一条新记录插入到Products表格中
Using myConnection As New SqlConnection(ConfigurationManager.ConnectionStrings
("ImageGalleryConnectionString").ConnectionString)

Const SQL As String = "INSERT INTO [Pictures] ([Title],  [MIMEType],  [ImageData]) VALUES (@Title,
@MIMEType,  @ImageData)"
Dim myCommand As New SqlCommand(SQL,  myConnection)
myCommand.Parameters.AddWithValue("@Title",  PictureTitle.Text.Trim())
myCommand.Parameters.AddWithValue("@MIMEType",  MIMEType)
```

```
'把FileUpload控件的InputStream加载到字节数组中
Dim imageBytes(UploadedFile.PostedFile.InputStream.Length) As Byte
UploadedFile.PostedFile.InputStream.Read(imageBytes, 0, imageBytes.Length)
  myCommand.Parameters.AddWithValue("@ImageData", imageBytes)
  myConnection.Open()
  myCommand.ExecuteNonQuery()
  myConnection.Close()
End Using
End Sub
```

在此，这个事件处理器首先确保已经上传一个文件。然后，它根据被上传的文件的扩展名来决定MIME类型。

上面最值得注意的是那些设置@ImageData参数的代码部分。首先，创建一个名为imageBytes的字节数组并且使其长度为被上传的文件相应的InputStream。然后，从InputStream中使用Read方法把二进制内容填入这个字节数组。注意，正是这个字节数组被指定为@ImageData的值。

五、上传图像并使用ASP.NET 2.0数据源控件代码存储二进制数据

尽管ADO.NET方法工作在一个ASP.NET 2.0应用程序环境下，但是，你还能够使用ASP.NET 2.0的数据源控件来把二进制数据存储到一个数据库，这不要求你编写ADO.NET代码。在这个演示程序中所使用的SqlDataSource控件包含了一个InsertCommand，以及相应于Title，MIMEType和ImageData值的参数：

```
＜asp:SqlDataSource ID="UploadPictureDataSource" runat="server"
ConnectionString="..."
InsertCommand="INSERT INTO [Pictures] ([Title], [MIMEType], [ImageData]) VALUES (@Title, @MIMEType,
@ImageData)"＞

＜InsertParameters＞
＜asp:Parameter Name="Title" Type="String" /＞
＜asp:Parameter Name="MIMEType" Type="String" /＞
＜asp:Parameter Name="ImageData" /＞
＜/InsertParameters＞
＜/asp:SqlDataSource＞
```

注意，在此，ImageData参数并没有指定一个类型。如果你试图使用GUI向导来构建SqlDataSource的语法，那么，它将可能给它指定Type="Object"，然而，这个Type="Object"将会产生一个sql_variant类型的参数。然而，该sql_variants类型不能用来存储图像或varbinary(MAX)数据类型，因为该sql_variant的内在数据大小不能超过8，000个字节。(如果你试图使用Type="Object"，然后试图保存超过8，000字节大小的二进制数据，那么，系统将抛出一个异常并显示消息"Parameter '@ImageData' exceeds the size limit for the sql_variant datatype"；如果你试图添加不到8，000字节大小的二进制数据，那么，该异常将显示消息"Implicit conversion from data type sql_variant to varbinary(max) is not allowed")。

另外，DetailsView控件包含了两个TemplateField。其中，一个TemplateField中使用一个TextBox控件来显示标题栏；另一个使用一个FileUpload控件来表示ImageData栏。最终结果是得到一个看上去类似于在"上传图像并使用ADO.NET代码存储二进制数据"一节中的用户接口。当点击DetailsView的"Insert"按钮时，它的Inserting事件激发，这时二进制数据必须从FileUpload控件中获取，读到一个字节数组中，并且赋值给适当的参数：

```
Protected Sub UploadPictureUI_ItemInserting(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.DetailsViewInsertEventArgs) Handles UploadPictureUI.ItemInserting
  '引用FileUpload控件
  Dim UploadedFile As FileUpload = CType(UploadPictureUI.FindControl("UploadedFile"), FileUpload)

  '确保已经成功上传一个文件
  If UploadedFile.PostedFile Is Nothing OrElse String.IsNullOrEmpty(UploadedFile.PostedFile.FileName) OrElse
UploadedFile.PostedFile.InputStream Is Nothing Then
    ...显示错误信息...
    e.Cancel = True
  Exit Sub
End If

'确保我们在处理一个JPG或GIF文件
Dim extension As String = Path.GetExtension(UploadedFile.PostedFile.FileName).ToLower()
Dim MIMEType As String = Nothing
Select Case extension
  Case ".gif"
    MIMEType = "image/gif"
  Case ".jpg", ".jpeg", ".jpe"
    MIMEType = "image/jpeg"
  Case ".png"
    MIMEType = "image/png"
  Case Else
    '无效文件类型上载
    ... 显示错误信息...
    e.Cancel = True
    Exit Sub
End Select

"指定MIMEType和ImageData参数的值
e.Values("MIMEType") = MIMEType
'把FileUpload的InputStream加载进字节数组中
Dim imageBytes(UploadedFile.PostedFile.InputStream.Length) As Byte
UploadedFile.PostedFile.InputStream.Read(imageBytes, 0, imageBytes.Length)
  e.Values("ImageData") = imageBytes
End Sub
```

就象前面的"Insert"按钮的Click事件处理器一样，该DetailsView的Inserting事件处理器也执行相同的逻辑-只有一些小小的语法差别。首先，既然FileUpload控件位于一个模板内，所以，必须使用FindControl("controlID")方法以编程方式来引用它。一旦对它进行了引用，即对之进行相同的检查以确保一个文件被成功上传，并且允许相应的扩展名。对于DetailsView的Inserting事件处理器存在一个微小的区别是，如果出现了错误，那么，我们需要通知该DetailsView停止相应的插入工作-这是通过把e.Cancel属性设置为True实现的。

检查完之后，MIMEType和ImageData参数将被使用e.Values("parameterName")=value语法进行赋值。就象在前面的ADO.NET示例中一样，首先把该二进制数据读取到一个字节数组中，然后把该字节数组赋值给该参数。

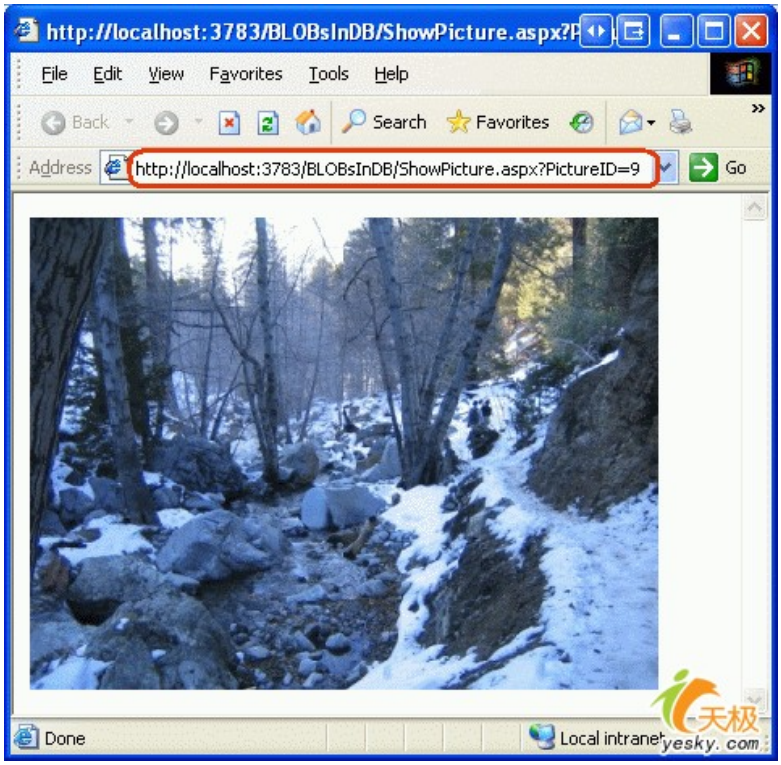
六、 显示二进制内容

无论你使用什么技术把数据存储在数据库中，为了检索并显示二进制数据，我们需要创建一个新的ASP.NET页面。这个名字为ShowPicture.aspx的页面，将通过querystring把一个PictureID传递给它，并且从指定的产品的ImageData域中返回该二进制数据。一旦完成，通过访问/ShowPicture.aspx?PictureID=picutreID地址即可看到一个特定的图片。因此，为了把一个图像显示在一个web页面上，我们可以使用一个图像控件并把它的ImageUrl属性设置成适当的URL。

注意，这个ShowPicture.aspx在其.aspx页面中并没有包括任何HTML标记。在code-behind类的Page_Load事件处理器中，将使用ADO.NET代码从数据库中检索指定的Pictures行的MIMEType和ImageData。然后，该页面的ContentType被设置为MIMEType域的值，并且使用Response.BinaryWrite(ImageData)输出该二进制数据：

```
Protected Sub Page_Load(ByVal sender As Object,  ByVal e As System.EventArgs) Handles Me.Load
  Dim PictureID As Integer = Convert.ToInt32(Request.QueryString("PictureID"))

  '与数据库连接并且返回指定的图片的图像内容和MIME类型
  Using myConnection As New SqlConnection(ConfigurationManager.ConnectionStrings
("ImageGalleryConnectionString").ConnectionString)

  Const SQL As String = "SELECT [MIMEType],  [ImageData] FROM [Pictures] WHERE [PictureID] = @PictureID"
  Dim myCommand As New SqlCommand(SQL,  myConnection)
  myCommand.Parameters.AddWithValue("@PictureID",  PictureID)

  myConnection.Open()
  Dim myReader As SqlDataReader = myCommand.ExecuteReader

  If myReader.Read Then
    Response.ContentType = myReader("MIMEType").ToString()
    Response.BinaryWrite(myReader("ImageData"))
  End If

  myReader.Close()
  myConnection.Close()
  End Using
End Sub
```

当ShowPicture.aspx页面完成后，你就可以通过直接访问URL或通过一个图像web控件(或经由静态＜img src="ShowPicture.aspx?ProductID=productID" .../＞标记)来观看图像。下面的第一幅屏幕快照展示了当直接通过ShowPicture.aspx观察时的一个图像;第二个屏幕快照展示了该图像画廊的Default.aspx页面-程序中在一个FormView控件内使用了一个Image Web控件，从而允许用户按页面浏览画廊中的图像。



七、 结论

当构建数据驱动的应用程序（其中必须捕获二进制数据）时，开发者必须决定是把二进制数据存储在文件系统中或是直接存储在数据库内。两种方案都存在各自的优点与缺点-正如在本文中所讨论的。如果你选择把二进制数据保存在数据库中，那么，你需要多付出一些努力来实现插入、更新和检索数据。在本文中，我们通过使用ADO.NET编码和ASP.NET 2.0 SqlDataSource控件分析了如何直接把图像文件上传到一个数据库。