汪毅 彭江平 (长沙湖南大学工商管理学院 410079)

Visual Studio.NET下自定义控件的开发与使用

摘要: 本文介绍了与自定义控件相关的概念，并利用C#语言在Visual Studio. NET下开发了一个完整的实例，详细描述了开发和使用自定义控件的方法和步骤。

关键词：面向对象程序设计 类 组件 控件 属性 事件

The Development and Use of Custom Control on Visual Studio.NET

1 引言

随着软件开发技术的不断进步，面向对象技术正日益成为目前程序开发的主流。微软将面向对象的有关概念和技术全面用于其寄予厚望的Visual Studio.NET开发平台，并开发了一种新的完全面向对象的语言——C#。面向对象程序设计的主要工作是描述和应用类，其编程类似于堆积木，把各种已有素材组合起来完成某一功能。因此，又可以把面向对象程序设计分成两类：一类是面向对象应用程序设计，这是面向对象程序设计的主体；另一类是类库设计，为面向对象程序设计提供"素材"，这是面向对象程序设计的基础。其中类库设计又可以被称为自定义组件与控件的开发。虽然现在的开发工具提供了越来越多的系统控件，但在具体应用中，尤其是一些专业领域，已有控件总不能满足我们的特殊需求，这时就必须开发我们自己的控件了。本文就介绍如何使用C#语言在Visual Studio.NET（简体中文版）下构建自定义控件。

2 自定义控件的有关概念和开发步骤

2.1 对象、类、组件与控件

(1) 对象描述客观事物的一个实体，是构成系统的基本单元，由一组属性和操作组成。

(2) 类是一组具有相同属性和相同操作的对象的集合，是对象的抽象，给出了属于该类的全部对象的抽象定义，包括类的属性、操作和其他性质。

(3) 组件是指一段经常用到的公共代码，封装在某个单元模块中。在Visual Studio.NET中，类为了变成组件，必须实现System.ComponentModel.Icomponent接口，并提供一个没有参数的构造方法。如果一个类继承自实现了这个接口的类，那它就是组件。组件还可以被包含和安置在容器中，可以通过容器提供的场所和容器进行交互。

(4) 控件实际上是特殊的组件。如果一个组件具有用户接口或界面，就被称之为控件。

2.2 自定义控件及其分类

用户通过继承和修改系统控件库所提供的控件，使其拥有新的属性、方法和事件，就可以得到自定义控件。在实际开发中，我们一般又把自定义的控件分为两类，一种是用户控件（复合控件），继承自UserControl类，由其他控件复合而成；另一种是定制控件，继承自Control类，通过在Paint事件方法中调用Graphics对象的方法绘制用户界面，这才是名副其实的自定义控件，也是本文讨论的重点。

2.3 自定义控件的开发步骤

我们以一个实例来说明自定义控件的开发步骤。我们将开发一个定制控件用来显示图像，当用户点击控件内部区域时，以鼠标位置作为显示图像的右下角；而图像的左上角保持不变，为控件的左上角，即用户可以通过点击鼠标改变图像显示的大小。

2.3.1 创建项目

在Visual Studio .NET中，选择文件—>新建—>项目，在"新建项目"对话框中把项目类型设为"Visual C# 项目"，把"模板"设为"Windows控件库"，在"名称"文本框中输入文件名，这里我们假设为ImageZoomer，选定文件存储位置，单击"确定"按钮。此时，自动创建的控件就是用户控件，继承自UserControl类，默认名称为UserControl1。在设计窗口上有一个灰色的窗体代表这个控件，可以在其上添加其他控件。在类UserControl1中实现控件的方法，就像创建主窗体或对话框一样。因此，我们把重点放在定制控件的开发上。

2.3.2 在项目中添加定制控件

选择视图—>解决方案资源管理器，在弹出的树形视图中右键单击，选择删除，把这个文件从项目中删掉。再选择项目—>添加新项，在弹出的对话框中把模板设为"自定义控件"，在"名称"框中输入此控件文件的名称（假设为ImageZoomerControl.cs），然后单击"打开"。

在"解决方案资源管理器"右击ImageZoomerControl.cs，选择"查看代码"，可以看到该文件的完整代码，从代码中我们可以知道，自定义控件类ImageZoomerControl包含在一个以ImageZoomer命名的名字空间内，即一个Windows控件库文件实际上定义了一个名字空间。此时，自定义控件包括两个空方法：构造方法和窗口刷新方法。

2.3.3 为控件添加属性

为了实现我们在上文提到的功能，要为这个控件添加两个int类型的属性ViewWidth、ViewHeight和一个Bitmap类型的属性ViewImage，其中前两个属性分别指定图像尺寸改变后的宽度和高度，而ViewImage则指定要显示的图像。这里我们只详细介绍添加属性ViewWidth。

(1) 添加私有域成员 width

```
private int width;
```

(2) 在类的构造方法中初始化成员，用窗体的尺寸初始化这两个与成员：

```
width=this.Width;
```

(3) 定义属性

```
[Category("ImageZoomer"),Description("The Viewed image width.")]
public int ViewWidth
{get {return width;}
set {if (value>=0)
    {width=value; Invalidate(this.ClientRectangle);}
    }
}
```

这一段代码实现对属性 ViewWidth 的读写。其中，set 方法对 value 值进行了检查，只接受大于或等于0的值，然后刷新窗口。方括号内的部分称为特性，是声明语句中的标记，可以在运行时刻通过内省获取它提供的附加信息。特性也是类，继承自 System.Attribute 类，位于 System.Component 名字空间中，几乎可以应用于所有的语言元素，如类型、属性、方法等。Category 特性制定属性或事件组的分类名称，以使控件的属性和事件可以在属性窗口中分组显示。Description 特性定义了一小段文字，当用户在属性窗口中选中这个属性或事件时，文本会显示在属性窗口的底部。这个属性的set方法对value值进行了检查，只接受大于或等于0的值。类似的，我们可以添加属性 ViewWidth 和 ViewImage。

2.3.4 为控件添加事件

事件是某个"动作"发生时对象发出的消息。发出（触发）事件的对象称为事件发送者，捕捉事件并对它做出响应的对象称为事件接收者。在事件的发送中，事件接收者告诉事件发送者它对某个事件感兴趣，并告诉发送者当事件发生时调用自身的某个方法；事件发送者记录下对它感兴趣的方法，并在事件发送时调用这一方法。在 Visual Studio.NET 中，使用代表作为中介实现这种交流功能。

代表是包含方法引用的类，具有署名特征，只能包含与署名特征一致方法的引用。事件发送者对象可以声明一个代表对象，并使用它记录下事件接收者的方法，当事件发生时，就可以通过代表调用这个方法。

在 Visual Studio.NET 中，如果发出事件的类被命名为 EventName，则需提供如下两个类：

(1) 一个包含事件数据的类，名称为 EventNameEventArgs，这个类必须继承自基类 System.EventArgs；

(2) 一个名为 EventNameEventHandler 的事件代表。

如果 ASP .NET 中已有的事件数据类和事件代表合适，可以不必声明这两个类。但自定义的类必须提供以下内容：

① 用事件代表声明一个公共的事件成员或属性，以便事件接收者注册方法。

② 声明一个名为 OnEventName 的发出事件的方法，在这个方法中调用事件代表对象中注册的方法。

本例中，当图像的显示大小改变后，控件将以事件的形式通知使用它的客户，报告新的大小，因此，我们可以依以上所述为自定义控件添加一个名为 ViewSizeChanged 的事件。

• 添加事件数据类

由于需要在事件中报告图像的显示尺寸，因此默认的 EventArgs 类不能使用，而要自定义事件数据类。我们可以在 ImageZoomerControl 的声明之前加入如下代码声明事件数据类：

```
public class ViewSizeChangedEventArgs : EventArgs
{ public int Width, Height;
  public ViewSizeChangedEventAr( ){}
}
```

• 添加事件代表

由于这里的事件代表署名将接受类型 ViewSizeChangedEventArgs 的参数，因此已的 EventHandler 事件代表将不能使用，而需声明新的事件代表。同样，我们可以把事件代表的声明写在自定义控件的前面：

```
public delegatevoidViewSizeChangedEvent Handler(object sender,
ViewSizeChangedEventArgs e);
```

• 添加事件属性

为了能够在窗体设计工具的属性窗口中看到

事件，需要把事件声明为属性，而不是公开的域成员。除了需要在属性类型前面加上 event 关键字外，其添加方法与添加其它类型的属性一样：

```
public event DisplaySizeChangedEventHandler DisplaySizeChanged
{add {eventHandler+=value;}
remove {eventHandler-=value;}}
```

其中，eventHandler 初始值为 null，表示一个私有的事件代表，声明方法与 ViewWidth 属性中的 width 类似。

• 发出事件

发出事件就是调用事件代表所指向的方法。代码如下：

```
protected virtual void OnViewSizeChanged (ViewSizeChangedEventArgs e)
{if(eventHandler!=null)
  {Invoke(eventHandler,new object []{this, e});}
}
```

代码中的 Invoke 是 Control 类中的方法，带有两个参数，第 1 个是事件代表对象，第 2 个是 object 数组，表示事件数据。

2.3.5 重载基类方法

为了实现图像大小随鼠标位置的变化而变化的功能，Control 类中的 onMouseDown 方法应被重载，实现在单击鼠标时，以鼠标位置作为事件数据，调用 OnViewSizeChanged 方法发出事件，同时更新窗口的显示的功能。

```
protected override void OnMouseDown (MouseEventArgs e)
{ width=e.X; height=e.Y;
Invalidate(this.ClientRectangle);
ViewSizeChangedEventArgs eventData=new
ViewSizeChangedEventArgs( );
eventData.Width=width; eventData.Height=height;
OnViewSizeChanged(eventData);
base.OnMouseDown(e);
}
```

2.3.6 绘制控件

定制控件必须通过在 Paint 方法中调用 Graphics 对象的方法绘制用户界面：

```
protected override void OnPaint(PaintEventArgs pe)
{ if(bitmap==null){return;}
Graphics g=pe.Graphics;
g.Transform=new System.Drawing.Drawing2D.Matrix(1,0,0,1,0,0);
g.DrawImage(bitmap,new
Rectangle(0,0,width,height),0,0,bitmap.Width,bitmap.Height,GraphicsUnit.Pixel);
g.ResetTransform();
}
```

2.3.7 编译控件，生成 DLL 文件

完成程序后，还必须编译，生成 DLL 文件，才能像其他控件一样被应用程序正常调用。选择生成—>生成，编译这个控件即可。

3 使用自定义控件

自定义控件构建完成后，就可以将其作为素材用于应用程序开发，ASP .NET 环境下，在应用程序中使用自定义控件有两种方法。

3.1 使用工具箱

选择工具—>自定义工具箱，然后选择.NET 框架组件选项卡，再点击“浏览”按钮，选择控件所在的 DLL 文件，单击“打开”。这时，所选控件的有关内容就会出现在选项卡上。点击其前面的方框，选中这个控件，确定之后自定义控件就可以像其他的ASP控件一样出现在开发环境下的工具箱中，其使用方法与其它控件并无区别。

3.2 不使用工具箱

如果不想使用工具箱，情况就有所不同了。

(1) 选择文件—>新建—>项目，创建一个 C# 类型的 Windows 应用程序项目。

(2) 添加控件引用。选择项目—>添加引用，在弹出的对话框中单击“浏览”按钮，同样选择控件所在的 DLL 文件，单击“打开”。在“选定的组件”对话框中选定控件，确定后即可。这样，ASP .NET 将把控件所在的 DLL 文件拷贝到当前项目所在的输出子目录（如 Debug）下，使得该项目在设计时刻和运行时刻均能使用此控件。

(3) 创建控件对象。完成以上各步后，还必须用手工方法在窗体上添加自定义的控件。假设在 Form1 上使用我们刚才开发的 ImageZoomer Control 控件。

① 打开 Form1.cs 源代码文件，在类 Form1 的声明之前添加对名字空间 ImageZoomer 的引用，这是 ImageZoomerControl 所在的名字空间：

```
using ImageZoomer;
```

② 在类的内部声明一个 ImageZoomer Control 类型的成员：

```
private ImageZoomer.ImageZoomerControl imageZoomer;
```

③ 在InitializeComponent方法中添加以下语句，初始化这个成员，并把它加入到窗体上：

```
this.imageZoomer=new ImageZoomer.ImageZoomerControl( );
imageZoomer.Size=new System.Drawing.Size ( 200, 200 );
this.Controls.Add(this.imageZoomer);
```

这样，在Form1.cs设计窗口就会出现一个灰色的区域，这就是ImageZoomerControl控件，其使用方法与系统原有控件一样。

参考文献

1 郑小平，Visual C#.NET开发实践 [M]，人民邮电出版社，2001。

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