自定义菜单开发实例

在 ToolbarControl 上不仅可以驻留命令和工具，还可以驻留菜单。要在工具条上驻留菜单，首先通过 IItemDef 接口将菜单中的各个命令类组织在一个项目文件（DLL）中。其次，其他项目通过添加该 DLL 的引用，进而将菜单添加到 ToolbarControl 上。

下面举例说明如何将地图符号化的六个命令和包含这些命令的**菜单项 SymbologyMenu** 组织在一个**项目 Symbology** 中。

本实例将工具条菜单和六种符号化命令的代码封装在一个动态链接库文件 DLL 中，并在新建的工程中引用封装好的 DLL，具体步骤如下：

1.新建 DLL 文件。打开 VS，新建项目（project），选择新建项目对话框中的“class library”（ 类库），命名为“Symbology”，点击确定

2. 鼠标右键新建的项目 Symbology，选择添加“class”（类），在内置的模板中选择“Base Menu”，将其命名为“SymbologyMenu”，并点击“Add”（添加）

3.在“SymbologyMenu”类中添加 GetItemInfo 方法、ItemCount 属性，重写 Caption、Name 属性，实现代码如下：

```
public sealed class SymbologyMenu : BaseMenu  { //……
                  public SymbologyMenu(){     } //构造函数
                  public override string Caption{  get{return "SymbologyMenu";      }       }
                  public override string Name { get{return "SymbologyMenu";        }       }
                  public int ItemCount { get { return 6; } }
                  public void GetItemInfo(int pos, IItemDef itemDef)
                  {   switch (pos) {
                           case 0: itemDef.ID = "Symbology.SimpleRender"; break;//简单着色
                           case 1: itemDef.ID = "Symbology.UniqueValueRender"; break;//唯一值着色
                           case 2: itemDef.ID = "Symbology.ClassBreakRender"; break;//分级着色
                           case 3: itemDef.ID = "Symbology.ProportionalSymbol"; break;//按比例着色
                           case 4: itemDef.ID = "Symbology.DotDensityRender"; break;//点密度图
                           case 5: itemDef.ID = "Symbology.BarChartRender";break; //饼图 }
                  }
         }
```

4.在项目“Symbology”中，分别添加六个新类，选择新类的基类为“Base Command”，将这些新类分别命名为：SimpleRender、UniqueValueRender、ClassBreakRender、ProportionalSymbol、DotDensityRender、BarChartRender（可在六个类中分别加入实现代码，后文举例说明 DotDensityRender 的实现代码）

5.在 SymbologyMenu 的构造函数中添加下拉菜单项，代码如下：

```
public SymbologyMenu()  { //构造函数
      AddItem("Symbology.SimpleRender");       AddItem("Symbology.UniqueValueRender");
      AddItem("Symbology.ClassBreakRender");   AddItem("Symbology.ProportionalSymbol");
      AddItem("Symbology.DotDensityRender");   AddItem("Symbology.BarChartRender");        }
```

6.生成点密度图 DotDensityRender 的代码如下：

```
using System;  using System.Drawing;  using System.Runtime.InteropServices;
using ESRI.ArcGIS.ADF.BaseClasses;    using ESRI.ArcGIS.ADF.CATIDs;    using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Geodatabase;     using ESRI.ArcGIS.Carto;     using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Display;
namespace Symbology
{  //……
          public sealed class DotDensityRender : BaseCommand
          { private IHookHelper m_hookHelper;
            public DotDensityRender() {
            base.m_category = "Symbology.Item"; base.m_caption = "DotDensityRender";
```

```
            base.m_message = "DotDensityRender";  base.m_toolTip = "点密度图";
            base.m_name = "DotDensityRender";
            try{//此处代码不用修改，省略 } catch (Exception ex){ //此处代码不用修改，省略  }
                  }
            public override void OnCreate(object hook) {  //此处代码不用修改，省略  }
            public override void OnClick() {
               try {
                  string strPopField = "value"; //字段
                  IActiveView pActiveView = m_hookHelper.ActiveView;//活动视图
                  IMap pMap = m_hookHelper.FocusMap;//地图
                  IGeoFeatureLayer pGeoFeatureL = pMap.get_Layer(0) as IGeoFeatureLayer;//要素图层
                  IDotDensityRenderer pDotDensityRenderer = new DotDensityRendererClass(); //渲染
                  IRendererFields pRendererFields = (IRendererFields)pDotDensityRenderer; //渲染字段
                  pRendererFields.AddField(strPopField, strPopField);
                  IDotDensityFillSymbol pDotDensityFillS = new DotDensityFillSymbolClass();
                  pDotDensityFillS.DotSize = 5;
                  pDotDensityFillS.Color = GetRGB(0, 0, 0);
                  pDotDensityFillS.BackgroundColor = GetRGB(239, 228, 190);
                  ISymbolArray pSymbolArray = (ISymbolArray)pDotDensityFillS;
                  ISimpleMarkerSymbol pSimpleMarkerS = new SimpleMarkerSymbolClass(); //形状 symbol
                  pSimpleMarkerS.Style = esriSimpleMarkerStyle.esriSMSCircle;
                  pSimpleMarkerS.Size  = 5;
                  pSimpleMarkerS.Color = GetRGB(128, 128, 255);
                  pSymbolArray.AddSymbol((ISymbol)pSimpleMarkerS);
                  pDotDensityRenderer.DotDensitySymbol = pDotDensityFillS;
                  pDotDensityRenderer.DotValue = 0.5;
                  pDotDensityRenderer.CreateLegend();
                  pGeoFeatureL.Renderer = (IFeatureRenderer)pDotDensityRenderer;
                  pActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography, null, null);
                  }
               catch (Exception ex) {
                  MessageBox.Show(ex.ToString(),"错误信息提示", MessageBoxButtons.OKCancel,
                  MessageBoxIcon.Error);     }   }
            private IRgbColor GetRGB(int red, int green, int blue)  {
                           IRgbColor rgbColor = new RgbColorClass();
                           rgbColor.Red = red;  rgbColor.Green = green;
                           rgbColor.Blue = blue;  return rgbColor;       }
         }
```

7.确定以上代码无误后，在 VS 中选择“Build”（生成）→“Build Solution”，在项目 Symbology 的“Bin”→“debug”文件夹中将看到生成的 DLL 文件“Symbology.dll”

8.打开需要引用该菜单的工程（如 IDNAME）中，选择项目→添加引用→浏览，添加上一步生成的 DLL 文件“Symbology.dll”，并利用“using Symbology;”将其导入程序

9.要在 IDNAME 项目中的 ToolbarControl 控件上添加该菜单，需要在主窗体的 Load 事件中加入以下代码：

```
private void frmMain_Load(object sender, EventArgs e)   {
                  IMenuDef menuDef = new Symbology.SymbologyMenu();
                  axToolbarControl1.AddItem(menuDef, -1, -1, false, -1,
                                          esriCommandStyles.esriCommandStyleIconAndText); }
```