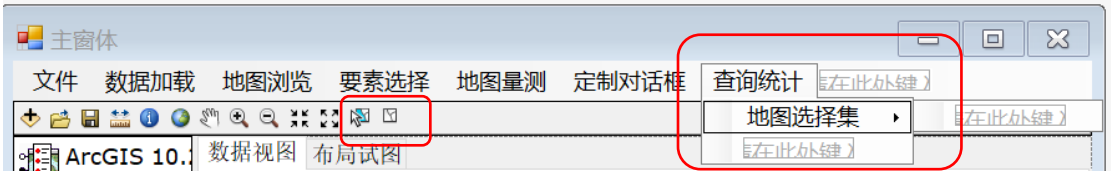
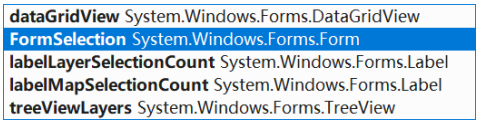
查询统计——地图选择集

1. 在主窗体中添加菜单项：查询统计，添加子菜单项：地图选择集

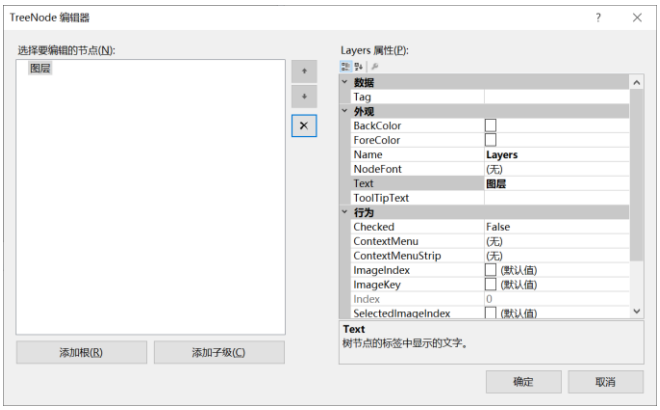


在 ToolbarControl 中添加 SelectFeature 和 ClearSelectedFeature 两个工具

2. 在解决方案中添加窗体：FeatureSelection，搭建窗体，详细控件如下：



序号	工具控件类型	Name	Text	备注
1	Form	FormSelection	地图选择集	
2	DataGridView	dataGridView		
3	Label	labelLayerSelectionCount	当前图层选择了 0 个要素。	
4	Label	labelMapSelectionCount	当前地图共选择了 0 个要素。	
5	TreeView	treeViewLayers		在 Nodes 属性中添加根“图层”，如下图



3. 在 FeatureSelection 窗体中添加如下代码：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geodatabase;

namespace GIS2022
{
    public partial class FormSelection : Form
    {
        private IMap currentMap;    //当前MapControl控件中的Map对象
        private IFeatureLayer currentFeatureLayer; //设置临时类变量来使用IFeatureLayer接口的当前图层对象

        /// <summary>
        /// 获得当前MapControl控件中的Map对象。
        /// </summary>
        public IMap CurrentMap
        {
            set
            {
                currentMap = value;
            }
        }

        public FormSelection()
        {
            InitializeComponent();
        }

        //在窗体加载时执行本函数，加载所有具有选择要素的图层
        private void FormSelection_Load(object sender, EventArgs e)
        {
            IFeatureLayer featureLayer; //设置临时变量存储当前矢量图层
            string layerName;   //设置临时变量存储当前图层的名称
            TreeNode treeNode;  //设置临时变量存储当前树节点的信息

            //对Map中的每个图层进行判断并加载图层名称
            for (int i = 0; i < currentMap.LayerCount; i++)
```

```
    {
        //如果该图层为图层组类型，则分别对所包含的每个图层进行操作
        if (currentMap.get_Layer(i) is GroupLayer)
        {
            //使用ICompositeLayer接口进行遍历操作
            ICompositeLayer compositeLayer = currentMap.get_Layer(i) as ICompositeLayer;
            for (int j = 0; j < compositeLayer.Count; j++)
            {
                //得到图层的名称
                layerName = compositeLayer.get_Layer(j).Name;
                //得到矢量图层对象的IFeatureLayer接口
                featureLayer = (IFeatureLayer)compositeLayer.get_Layer(j);
                //如果该图层选择集中的要素不为空，则在TreeView控件中添加一个树节点
                if (((IFeatureSelection)featureLayer).SelectionSet.Count > 0)
                {
                    //新建一个树节点，将图层名称作为树节点的名称
                    treeNode = new TreeNode(layerName);
                    //利用树节点的Tag属性，存储当前图层的IFeatureLayer接口信息
                    treeNode.Tag = featureLayer;
                    //将新建的树节点添加到TreeView控件中的根节点下
                    treeViewLayers.TopNode.Nodes.Add(treeNode);
                }
            }
        }
        //如果图层不是图层组类型，则同样在TreeView控件中的根节点下添加节点
        else
        {
            layerName = currentMap.get_Layer(i).Name;
            featureLayer = (IFeatureLayer)currentMap.get_Layer(i);
            if (((IFeatureSelection)featureLayer).SelectionSet.Count > 0)
            {
                treeNode = new TreeNode(layerName);
                treeNode.Tag = featureLayer;
                treeViewLayers.TopNode.Nodes.Add(treeNode);
            }
        }
    }
    //添加完节点后将根节点展开以显示所有的图层
    treeViewLayers.TopNode.Expand();
    //通过IMap接口的SelectionCount属性可以获取被选择要素的数量
    labelMapSelectionCount.Text = "当前地图共选择了 " + currentMap.SelectionCount + " 个要素。";
}

//在图层被选中时执行本函数，完成该图层所有选中要素的列表
private void treeView_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)
{
    try
```

```
{
    //首先清空DataGridView中的行和列
    dataGridView.Columns.Clear();
    dataGridView.Rows.Clear();

    //通过树节点的Tag属性获取以该节点名称命名的矢量图层
    currentFeatureLayer = e.Node.Tag as IFeatureLayer;
    //通过接口转换，使用IFeatureSelection接口获取图层的选择集
    IFeatureSelection featureSelection = currentFeatureLayer as IFeatureSelection;
    //通过ISelectionSet接口获取被选择的要素集合
    ISelectionSet selectionSet = featureSelection.SelectionSet;
    //通过ISelectionSet接口的Count属性可以获取被选择要素的数量
    labelLayerSelectionCount.Text = "当前图层选择了 " + selectionSet.Count + " 个要素。";

    //对当前图层要素的属性字段进行遍历，从而建立DataGridView中的列
    //获取所有的属性字段
    IFields fields = currentFeatureLayer.FeatureClass.Fields;
    for (int i = 0; i < fields.FieldCount; i++)
    {
        //通过遍历添加列，使用字段的AliasName作为DataGridView中显示的列名
        dataGridView.Columns.Add(fields.get_Field(i).Name, fields.get_Field(i).AliasName);
    }

    //对选择集进行遍历，从而建立DataGridView中的行
    //定义ICursor接口的游标以遍历整个选择集
    ICursor cursor;
    //使用ISelectionSet接口的Search方法，使用null作为查询过滤器，cursor作为返回值获取整个选择集
    selectionSet.Search(null, false, out cursor);
    //进行接口转换，使用IFeatureCursor接口来获取选择集中的每个要素
    IFeatureCursor featureCursor = cursor as IFeatureCursor;
    //获取IFeature接口的游标中的第一个元素
    IFeature feature = featureCursor.NextFeature();
    //定义string类型的数组，以添加DataGridView中每一行的数据
    string[] strs;
    //当游标不为空时
    while (feature != null)
    {
        //string数组的大小为字段的个数
        strs = new string[fields.FieldCount];
        //对字段进行遍历
        for (int i = 0; i < fields.FieldCount; i++)
        {
            //将当前要素的每个字段值放在数组的相应位置
            strs[i] = feature.get_Value(i).ToString();
        }
        //在DataGridView中添加一行的数据
        dataGridView.Rows.Add(strs);
```

```
                //移动游标到下一个要素
                feature = featureCursor.NextFeature();
            }
        }
        catch { }
    }

    //在列表中点击某一行时发生，将主窗体的地图定位到该行所表示的要素范围
    private void dataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        //获取当前所点击的行
        DataGridViewRow row = dataGridView.Rows[e.RowIndex];
        //每行的第一列是要素的ObjectID，获取该信息
        int objectID = Convert.ToInt32(row.Cells[0].Value);
        //使用IFeatureClass接口的GetFeature方法根据ObjectID获取该要素
        IFeature feature = currentFeatureLayer.FeatureClass.GetFeature(objectID);
        //定义新的IEnvelope接口对象获取该要素的空间范围
        ESRI.ArcGIS.Geometry.IEnvelope outEnvelope = new ESRI.ArcGIS.Geometry.EnvelopeClass();
        //通过IGeometry接口的QueryEnvelope方法获取要素的空间范围
        feature.Shape.QueryEnvelope(outEnvelope);
        //将主窗体地图的当前可视范围定义为要素的空间范围，并刷新地图
        IActiveView activeView = currentMap as IActiveView;
        activeView.Extent = outEnvelope;
        activeView.Refresh();
    }
  }
}
```

4. 完成 FeaureSelection 窗体代码后，在主窗体中添加菜单项：地图数据集的 click 事件响应函数

```
private void 地图选择集ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //新创建地图选择集窗体
        FormSelection formSelection = new FormSelection();
        //将当前主窗体中MapControl控件中的Map对象赋值给FormSelection窗体的CurrentMap属性
        formSelection.CurrentMap = mainMapControl.Map;
        //显示地图选择集窗体
        formSelection.Show();
    }
```