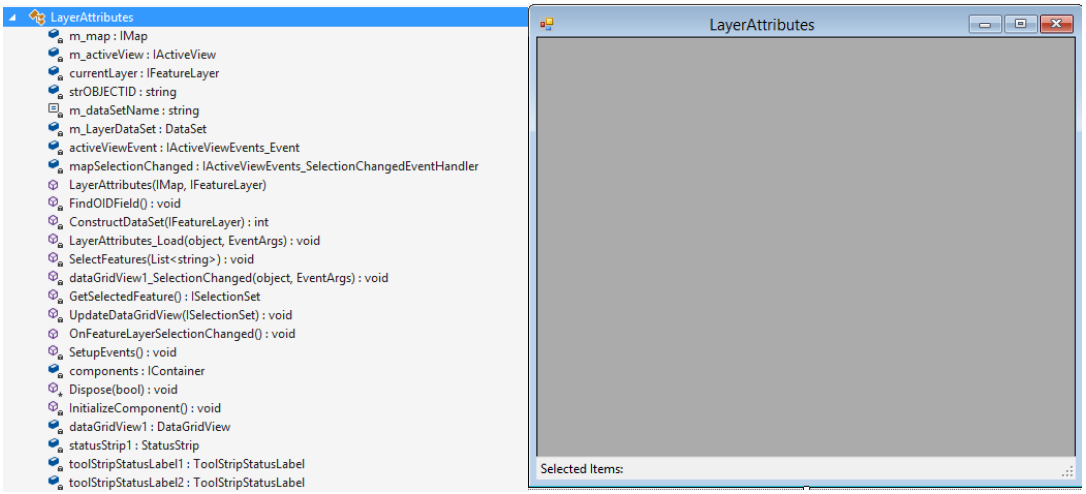
Chapter 4 Show Attributes

1. New a Windows Form, named as “LayerAttributes”, drag a DataGridView(Dock: Fill) and statusStrip into the form.



2. Implement the LayerAttributes class:

- Add class member:

```
IMap m_map = null;
IActiveView m_activeView = null;
IFeatureLayer currentLayer = null;
```

- Construction method:

```
public LayerAttributes(IMap map, IFeatureLayer featureLayer)
{
    InitializeComponent();
    m_map = map;
    m_activeView = map as IActiveView;
    currentLayer = featureLayer;

}
```

- Find OID Field:

```
string strOBJECTID=null;
private void FindOIDField()
{
    if (currentLayer == null)
        return;
    IFields fields = currentLayer.FeatureClass.Fields;
    IField field;
    for (int i = 0; i < fields.FieldCount; i++)
    {
        field = fields.get_Field(i);
        if (field.Type == esriFieldType.esriFieldTypeOID)
        {
            strOBJECTID = field.Name;
            break;
        }
    }
}
```

- ConstructDataSet:

```
const string m_dataSetName = "m_layerDataSet";
DataSet m_LayerDataSet = new DataSet(m_dataSetName);
private int ConstructDataSet(IFeatureLayer pFeatureLayer)
{
    ILayerFields pFeatureLayerFields = pFeatureLayer as ILayerFields;
    IFeatureClass pFeatureClass = pFeatureLayer.FeatureClass;
    int rows = 0;
    if (m_LayerDataSet.Tables[pFeatureLayer.Name] == null)
    {
        DataTable pTable = new DataTable(pFeatureLayer.Name);
        DataColumn pTableColumn;
        for (int i = 0; i < pFeatureLayerFields.FieldCount; i++)
        {
            pTableColumn = new DataColumn(pFeatureLayerFields.get_Field(i).AliasName);
            pTable.Columns.Add(pTableColumn);
            pTableColumn = null;
        }
        IFeatureCursor features = pFeatureLayer.Search(null, false);
        IFeature feature = features.NextFeature();
        while (feature != null)

        {
            DataRow pTableRow = pTable.NewRow();
            for (int i = 0; i < pFeatureLayerFields.FieldCount; i++)
            {
                if (pFeatureLayerFields.FindField(pFeatureClass.ShapeFieldName) == i)
                {
                    pTableRow[i] = pFeatureClass.ShapeType;
                }
                else
                {
                    pTableRow[i] = feature.get_Value(i);
                }
            }
            pTable.Rows.Add(pTableRow);
            feature = features.NextFeature();
        }
        rows = pTable.Rows.Count;
        m_LayerDataSet.Tables.Add(pTable);
        System.Runtime.InteropServices.Marshal.ReleaseComObject(features);
    }
    return rows;
}
```

- Load method of the Form (I will Define SetupEvents later):

```
private void LayerAttributes_Load(object sender, EventArgs e)
{
    this.Text = currentLayer.Name + "  Feature DataSet";
    FindOIDField();
    if (strOBJECTID == null) return;
    int rowCount = ConstructDataSet(currentLayer);
    dataGridView1.DataSource = m_LayerDataSet;
    dataGridView1.DataMember = currentLayer.Name;
    toolStripStatusLabel1.Text = "Selected Items: 0";
    toolStripStatusLabel2.Text = "Total Items: " +
        currentLayer.FeatureClass.FeatureCount(null).ToString();
    OnFeatureLayerSelectionChanged();
    SetupEvents();
}
```

3. New a Base Command Class “OpenAttributesCmd”:

- Add Class member:

```
IMapControl3 m_mapcontrol = null;

IFeatureLayer currentLayer = null;
```

- Add OnClick implementation:

```
public override void OnClick()
{
    // TODO: Add OpenAttributeTableCmd.OnClick implementation
    IMap map = null;
    if (m_hookHelper.Hook is IMapControl3)
    {
        m_mapcontrol = m_hookHelper.Hook as IMapControl3;
        currentLayer = m_mapcontrol.CustomProperty as IFeatureLayer;
        if (currentLayer == null) return;
        map = m_mapcontrol.Map;
    }
    if (map == null) return;
    LayerAttributes layerAttributeTable = new LayerAttributes(map, currentLayer);
    layerAttributeTable.Show(m_hookHelper as System.Windows.Forms.IWin32Window);
}
```

4. In the MainForm:

- Add Class member:

```
IToolbarMenu m_TocLayerMenu = new ToolbarMenuClass();
```

- In the Load method, add:

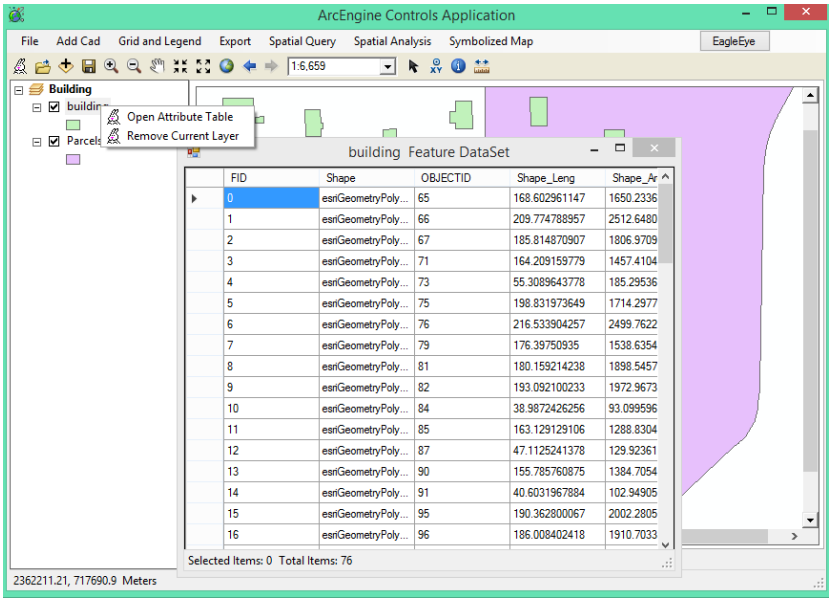
```
m_TocLayerMenu.AddItem(new OpenAttributeTableCmd(), 0, 0, false,
esriCommandStyles.esriCommandStyleIconAndText);

m_TocLayerMenu.SetHook(axMapControl1);
```

- Add MouseDown event to TOCControl:

```
private void axTOCControl1_OnMouseDown(object sender, ITOCControlEvents_OnMouseDownEvent e)
{
    IBasicMap map = new MapClass();
    ILayer layer = new FeatureLayerClass();
    object other = new object();
    object index = new object();
    esriTOCControlItem item = new esriTOCControlItem();
    axTOCControl1.HitTest(e.x, e.y, ref item, ref map, ref layer, ref other, ref index);
    if (e.button == 2)
    {
        if (layer is IFeatureLayer)
        {
            m_mapControl.CustomProperty = layer;
            m_TocLayerMenu.PopupMenu(e.x, e.y, axTOCControl1.hWnd);
        }
    }
}
```

- Now run the application (In the Chapter 16, I will add “Remove Current Layer”):



5. Back to LayerAttributes class, now I will add some methods that could show the selected items' count and refresh when selection changed (1. form selection change refresh map selection change 2. Map selection change refresh form selection change):

- SelectFeatures Method:

```
private void SelectFeatures(List<string> oidList)
{
    IFeatureClass featureClass = currentLayer.FeatureClass;
    string strID = string.Empty;
    string[] IDs = oidList.ToArray();
    for (int i = 0; i < IDs.Length; i++)
    {
        strID = IDs[i];
        IFeature selectedFeature = featureClass.GetFeature(Convert.ToInt32(strID));
        m_map.SelectFeature(currentLayer, selectedFeature);
    }
    m_activeView.PartialRefresh(esriViewDrawPhase.esriViewGeoSelection, null, m_activeView.Extent);
}
```

- SelectionChanged event of dataGridView:

```
private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    m_map.ClearSelection();
    m_activeView.PartialRefresh(esriViewDrawPhase.esriViewGeoSelection,
        null, m_activeView.Extent);
    DataGridViewSelectedRowCollection selectedRows = dataGridView1.SelectedRows;
    if (selectedRows == null) return;
    string strOID = string.Empty;
    List<string> OIDList = new List<string>();
    for (int i = 0; i < selectedRows.Count; i++)
    {
        DataGridViewRow row = selectedRows[i];
        strOID = row.Cells[strOBJECTID].Value.ToString();
        OIDList.Add(strOID);
    }
    SelectFeatures(OIDList);
    toolStripStatusLabel1.Text = "Selected Items : " + OIDList.Count.ToString();
    toolStripStatusLabel2.Text = "Total Items : " +
        currentLayer.FeatureClass.FeatureCount(null).ToString();
}
```

- GetSelectedFeature:

```
private ISelectionSet GetSelectedFeature()
{
    if (currentLayer == null) return null;

    IFeatureSelection featureSelection = currentLayer as IFeatureSelection;
    ISelectionSet selectionSet = featureSelection.SelectionSet;

    return selectionSet;
}
```

- UpdateDataGridView:

```
private void UpdateDataGridView(ISelectionSet selectedFeatures)
{
    IEnumIDs enumIDs = selectedFeatures.IDs;
    int iD = enumIDs.Next();
    while (iD != -1) //-1 is reutned after the last valid ID has been reached
    {
        for (int i = 0; i < dataGridView1.RowCount-1; i++)
        {
            if (dataGridView1.Rows[i].Cells[0].Value.ToString() == iD.ToString())
                dataGridView1.Rows[i].Selected = true;
        }

        iD = enumIDs.Next();
    }
}
```

- OnFeatureLayerSelectionChanged:

```
public void OnFeatureLayerSelectionChanged()
{

    ISelectionSet selectedFeatures = GetSelectedFeature();
    UpdateDataGridView(selectedFeatures);
}
```

- Add class member:

```
IActiveViewEvents_Event activeViewEvent = null;

IActiveViewEvents_SelectionChangedEventHandler mapSelectionChanged;
```

- SetupEvents:

```
private void SetupEvents()
{
    activeViewEvent = m_activeView as IActiveViewEvents_Event;
    mapSelectionChanged = new
        IActiveViewEvents_SelectionChangedEventHandler(OnFeatureLayerSelectionChanged);
    activeViewEvent.SelectionChanged += mapSelectionChanged;
}
```