Make model visible in the central map

Exercise outline

The goal of this exercise is to visualize multiple data sets on one single map. This makes it possible to compare different data sets at the same geospatial location(s) and/or for the same modelling time step(s).
With this exercise it should be possible to show, as an example, both the input data and the output data of a volume model in one single map view.

Create a new map layer provider

Add to the plugin project a new folder named ***Layers***. In this folder, create a new class named ***DrainageBasinFeatureCollection.cs*** and adapt the contents as shown below:

```
using System.Collections;
using System.ComponentModel;
using DeltaShell.Plugins.VolumeModel.Models;
using NetTopologySuite.Extensions.Features;
using SharpMap.Data.Providers;
namespace DeltaShell.Plugins.VolumeModel.Layers
{
    /// <summary>
    /// Defines a feature collection (used by layers for rendering) for a DrainageBasin
    /// </summary>
    public class DrainageBasinFeatureCollection : FeatureCollection
    {
        private readonly DrainageBasin drainageBasin;
        public DrainageBasinFeatureCollection(DrainageBasin drainageBasin)
            : base((IList)drainageBasin.Catchments, typeof(Feature))
        {
            this.drainageBasin = drainageBasin;
            // copy coordinatesystem and for monitor changes of the coordinatesystem
            CoordinateSystem = (GeoAPI.Extensions.CoordinateSystems.ICoordinateSystem) drainageBasin.
CoordinateSystem;
            drainageBasin.PropertyChanged += DrainageBasinPropertyChanged;
        }
        private void DrainageBasinPropertyChanged(object sender, PropertyChangedEventArgs e)
        {
            if (e.PropertyName == "CoordinateSystem")
            {
                CoordinateSystem = (GeoAPI.Extensions.CoordinateSystems.ICoordinateSystem) drainageBasin.
CoordinateSystem;
            }
        }
        public override void Dispose()
        {
            // Desubscribe from drainageBasin so this DrainageBasinFeatureCollection can be disposed
            drainageBasin.PropertyChanged -= DrainageBasinPropertyChanged;
            base.Dispose();
        }
    }
}
```

Then create a new class ***VolumeModelLayerProvider.cs*** in the same folder

```
using System.Collections.Generic;
using DelftTools.Shell.Gui;
using DeltaShell.Plugins.VolumeModel.Models;
using SharpMap.Api.Layers;
using SharpMap.Layers;
namespace DeltaShell.Plugins.VolumeModel.Layers
{
    public class VolumeModelMapLayerProvider : IMapLayerProvider
    {
        /// <summary>
        /// Defines that layers can be provided for volume models and DrainageBasins
        /// </summary>
        public bool CanCreateLayerFor(object data, object parentData)
        {
            return data is Models.VolumeModel ||
                   data is DrainageBasin;
        }
        /// <summary>
        /// Creates a volume model group layer and DrainageBasin layer
        /// </summary>
        public ILayer CreateLayer(object data, object parentData)
        {
            var volumeModel = data as Models.VolumeModel;
            if (volumeModel != null)
            {
                return new GroupLayer(volumeModel.Name);
            }
            var drainageBasin = data as DrainageBasin;
            if (drainageBasin != null)
            {
                return new VectorLayer("Drainage basin")
                    {
                        DataSource = new DrainageBasinFeatureCollection(drainageBasin)
                    };
            }
            return null;
        }
        /// <summary>
        /// Returns all children for which a child layer should be created in volume model group layers
        /// </summary>
        public IEnumerable<object> ChildLayerObjects(object data)
        {
            var volumeModel = data as Models.VolumeModel;
            if (volumeModel != null)
            {
                // In the end a child layer should be created for both the basin input data and the volume
output data
                yield return volumeModel.Basin;
                yield return volumeModel.Volume;
            }
        }
    }
}
```

The map layer provider class is derived from the *IMapLayerProvider* interface so that it can be registered in the gui plugin (see the next step).

The comments in the code explain the different parts of the provider implementation.

A description on the backgrounds and usage of (group) layers is not part of this tutorial.

Register the map layer provider in the gui plugin class

Register the map layer provider in the gui plugin by adding the following code to ***VolumeModelGuiPlugin.cs***:

```
using DeltaShell.Plugins.VolumeModel.Layers;
```

and

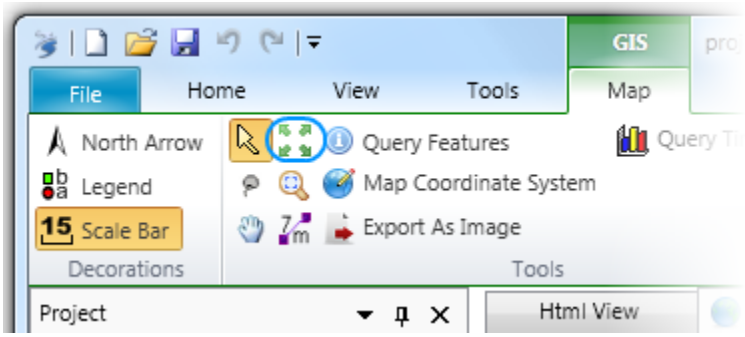
```
        public override IMapLayerProvider MapLayerProvider
        {
            get { return new VolumeModelMapLayerProvider(); }
        }
```

Delta Shell should now be able to open a map view for volume models, containing both their basin input data and their volume output data (if present).

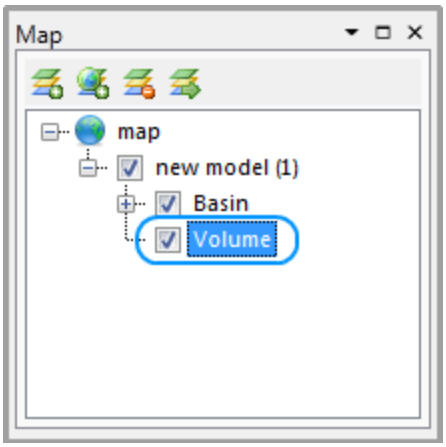
Exercise results

Set up a volume model as described in the results of a previous exercise (Create a simple Volume model).
After creating the volume model, the corresponding map view should have been opened automatically. However, the data will not be immediately shown in this view, even after running the volume model. Two further actions are required.

First of all, the input of the volume model data is present, but the map is zooming into a different area. To solve this problem, open the *Map* Ribbon tab and click the *Zoom to map extent* button:



Secondly, the output data of the volume model is initially hidden. To solve this problem, expand the volume model group layer in the *Map* window and tick the check box of the *Volume* layer:



The order of the different data layers in a map view can be adjusted using the context menus in the *Map* window (right click on a data layer | Order | ...).

Additionally, WMS layers can be included to help identifying the actual geographical location of map data. Follow these steps:

1. Add a new WMS layer to the map using the *Map* window (click the *Add New Wms Layer ...* button). In the *Open web link ...* dialog, select *Microsoft Bing Maps - Satellite Hybrid*. A new map layer will be added to the view.