ArcGIS Template Viewer for Silverlight

Version 2.2.1

Table of Content

Page

For additional information contact:
Environmental Systems Research Institute, Inc.
Attn: Contracts and Legal Services
380 New York Street
Redlands, California, U.S.A. 92373
Email: contracts@esri.com

1. What Is New in Version 2.2.1

1) Use ArcGIS API for Silverlight 2.2.1

2) Supports compiled plug-in widgets – a widget can be created in a separate project from the viewer and plugged in a deployed viewer (see more detail in section 5 – How to create a distributable plug-in widget)

3) Holding down hot keys Shift/Space and drawing a box on the map to zoom in/out map at any time and any where

4) Widget Navigation Buttons in PagedStackPanel are separated in both sides

5) Resources.resx is added in SilverlightViewer.Xap project to suport multiple language tips

6) Class WidgetConfigureBase is added. The configuration class of a widget must extend class WidgetConfigureBase. To load a widget configuration,

   override OnDownloadConfigXMLCompleted
   and add a line like following into the function

```
widgetConfig = (QueryConfig)QueryConfig.Deserialize(xmlConfig, typeof(QueryConfig));
```

7) Overriding function OnIsActiveChanged moves into the WidgetBase class. No need to override in an individual widget anymore, but you **HAVE TO** override ResetDrawObjectMode, if you use DrawObject in your widget. Please use the code of LocatorWidget, IdentifyWidget, and QueryWidget as examples

8) Three new widgets - Social Media, Data Extraction, and Redlines

9) Found bugs are fixed and many improvements.

2. How to Deploy

1) IIS (e.g. Windows Server)
   Copy folder **SilverlightViewer.Web** to C:\inetpub\wwwroot and change it into an application (you may give it a different name, e.g. SilverlightViewer) with IIS Manager or create a virtual directory for the folder with IIS Manger if you do not copy it into C:\inetpub\wwwroot. Set default.aspx or default.htm as the default page

   Please make sure the application runs under the ASP.NET v4.0 Application Pool

2) Java Web Server (e.g. Tomcat 6.0)
   From Eclipse - copy folder **SilverlightViewerWeb.jsp** to your Eclipse workspace, RENAME the folder to **SilverlightViewerWeb**, open Eclipse, and import the content in the folder into Eclipse as a project

   Then follow the instructions at http://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html to deploy the application

3. How to Configure Silverlight Application

All of the configuration files for the application and widgets are located in the SilverlightViewer.Web\ClientBin, or SilverlightViewerWeb.JSP\WebContent\ClientBin folder.

1) **AppConfig.xml** – Application Configuration File

   a) *ApplicationLogo* - A relative path of the logo image displayed on the Taskbar.

      To replace the application logo with your logo, create an "images" folder in the ClientBin directory and put your logo image in the "images" folder, and change the logo path to "../images/yourlogo.png"

   b) *ApplicationTitle* - Application title displayed on the Taskbar, e.g., "ArcGIS Template Viewer for Silverlight"

   c) *ApplicationSubtitle* - Application subtitle displayed on the Taskbar, e.g., "Powered by ArcGIS Server"

   d) *ApplicationHelpMenu* - Holds web site and help page links.

   e) *Map* - Element used to define map content and map loading extent

      - *InitialExtent* - The map extent at the first load. Its coordinate system must be consistent with the base map coordinate system

      - *FullMapExtent* - The map extent to which the map will be zoomed when the Full Map button is clicked. Its coordinate system must be consistent with the base map coordinate system

      - *BaseMap*

        - Attribute ***enable*** - Set to "BING" to choose Microsoft Bing Map as the BaseMap; set to "ArcGISMap" to choose ArcGISOnline cached map services or local cached map services served up by ArcGIS server

        - ***BingMap*** - List of BING map layers used as base map layers. Only one base map layer is visible at a time.

          Tag <*Server*> - "Staging" for the test or development phases; "Production" for finally deployed production

          Tag <*Token*> - A permanent Bing Map Key

- ***ArcGISMap*** - List of ArcGIS cached map service used as base map layers. Only one base map is visible at a time. Use ArcGISOnline map services or local cached map/image services served up by ArcGIS server

  Attribute ***restURL*** - the URL of a REST map service

  Attribute ***serviceType*** - "Cached" or "Image"

  Attribute ***showLabel*** - "true" or "false", indicating if a label layer configured in tag <*LabelLayer*> will be used combinedly with this layer, e.g. an Imagery layer

- *LivingMaps*

  List of ArcGIS map services to be overlaid above the base map. The Map Content widget lists all the living maps and the sub-layers of dynamic map services in a tree view. The visibilities of a living map and the sub-layers of a dynamic map service can be toggled in the Map Content widget

  Attribute ***restURL*** - the URL of a REST map service or feature layer

  Attribute ***serviceType*** - "Cached", "Dynamic", or "Feature"

  Attribute ***opacityBar*** - set to "true" to show an opacity slider bar to change the opacity of this LivingMap

  Attribute ***refreshRate*** - set to a value greater than 0 to trigger the layer refreshed automatically in an interval of seconds set by the value

f) *Taskbar* – Holder of the application Logo and Title, website links, map tools, and the Base-Map switching button

The Taskbar's skin changes depending on the Theme chosen for the application

The Taskbar's status can change between "Docked" and "Floating" by pressing CTRL + clicking on the Taskbar when theme AliceBlue, BurlyWood or CornsilkGray is chosen

- ***ToolbarButtons*** - Map navigation buttons include Pan, ZoomIn, ZoomOut, and Identify. Note: An *Identify* button is optional in version 2.1. The Identification Mode can also be activated by opening the *Identify Widget*.

g) *Widgets* – List of widgets used in this application. *Tip: It is highly recommended to remove the widgets not needed in your application to save memory*

   Attribute **title** - the title of a widget

   [*]Attribute **xapFile** - the XAP file name, in which the plug-in widget is wrapped (see more detail in section 5 – How to create a distributable plug-in widget)

   [*]Attribute **className** - the widget class name including namespace. If the widget is a plug-in widget (not included in the viewer project), the name of the dll file, which holds the widget, must be specified in this format "xxxxx.dll;WidgetClassName", e.g.`ChartingWidget.dll;ChartingWidget.MainPage`

   Attribute **hasGraphics** - set to "true" to create a graphics layer for the widget

   Attribute **openInitial** - set to "true" to make the widget window visible when the application is initially loaded

   Attribute **icon** - the relative path of the icon for the widget

   Attribute **configFile** - the path of the configuration file for the widget. They are usually stored in the ClientBin folder

h) *OverviewMapConfigFile* - The OverviewMap widget configuration file, stored in the ClientBin folder

i) *Theme* - Predefined styles for the Taskbar skin and color

   Predefined themes include AliceBlue, BurlyWood, Aquamarine, BlueBanner, CornsilkGray, VividSummer, CornerSpring, LargeBlueWave,SmallBlueWave

2) Configuration Files of Widgets

Most widget configuration files are simple enough for people to understand by reviewing the tag names with basic XML knowledge. Here I introduce a few of widget-configuration XML files.

a) **IdentifyWidget.xml** – IdentifyWidget Configuration File

- ***Tolerance*** - pixel number for creating a buffer area around the click point to select Point type features

- ***IdentifyOption*** - set to “all”, “visible”, or “top”, used to determine identifiable layers.

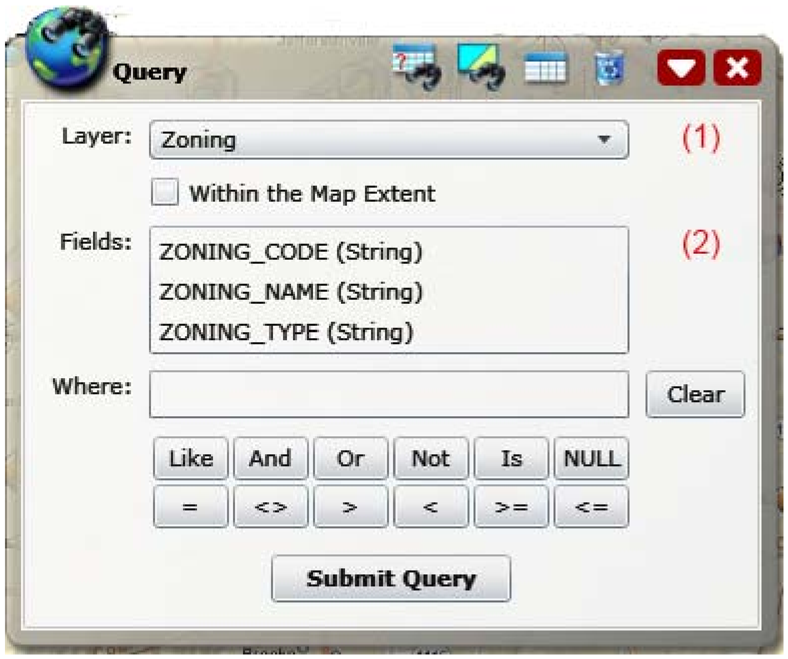
- ***IdentifyLayers*** - a list of identifiable LivingMap layers, used only when *IdentifyOption* is set to ‘all’

  Attribute ***title*** - must match the title of the LivingMap layers configured in AppConfig.xml

  Attribute ***layerIDs*** - a list of the feature layer IDs in a LivingMap separated with ',' or input '*' to indicate all layers, for example, "0,1,2" or "*"

b) **QueryWidget.xml** - QueryWidget Configuration File

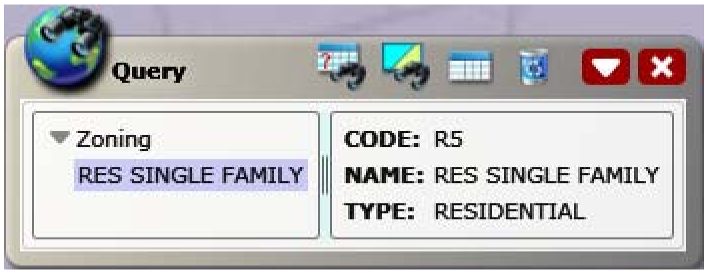
QueryLayers – List of feature layers used in QueryWidget - (1)



- Attribute ***restURL*** – the REST URL of a feature layer

- Tag ***QueryFields*** – fields of the feature layer used to construct a where clause (2)

- Tag ***OutputFields*** – fields, the values of which will be displayed in the result panel below



CODE: R5
NAME: RES SINGLE FAMILY
TYPE: RESIDENTIAL

- Tag ***OutputLabels*** – aliases of output fields in the result panel above (in bold)

- Tag ***MapTipTemplate*** – a string used to format the content in a map-tip window

  A map-tip window pops up when the mouse hovers over a graphic, which contains summary information about the feature highlighted by the graphic, including Title, Content (attribute values), Image and/or Hyperlink. All of them can be bound to the feature's attribute fields. They are formatted as:

  Title={#TitleField} or Title=xxx or Title=xxx: {#TitleField}

  Content=Alias 1:{#Field1}\n:{#Field2}\nAlias 3:{#Field3}

  Image={#ImageURL}[200*160] ----- here, [200*160] represent image size

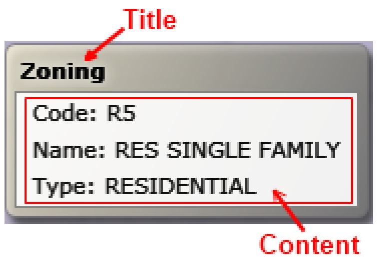
  Link={#LinkURL}

  in which, {#**FieldName**} represents a binding to an attribute field and **Content** can display in multiple lines by using "**\n**" as a separator in the format.

  The format strings of Title, Content, Image and Link must be separated with a semi-colon "**;**" Below are a few examples to show how to format a Map-Tip Template:

  ***Note: The MapTipTemplate format for all widgets is consistent in this application. In other words, descriptions here about the MapTipTemplate format can be used as a guide to build the MapTipTemplate of other widgets, e.g., SearchNearby widget and Charting widget.**

Title=Zoning;Content=Code: {#ZONING_CODE}\nName: {#ZONING_NAME}\nType: {#ZONING_TYPE}

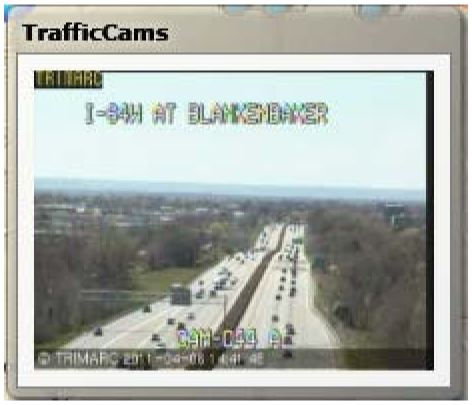


Zoning

Code: R5
Name: RES SINGLE FAMILY
Type: RESIDENTIAL

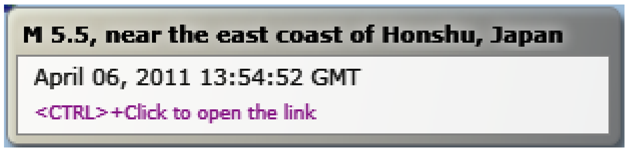
(1) A map-tip window with Title and Content

Title=TrafficCams;Image={#ImageURL}[200*160]



(2) A map-tip window with Title and Image

Title={#Title};Content={#Description};Link={#Link}



M 5.5, near the east coast of Honshu, Japan

April 06, 2011 13:54:52 GMT

<CTRL>+Click to open the link

(3) A map-tip window with Title and Hyperlink

4. How to Create a New Widget

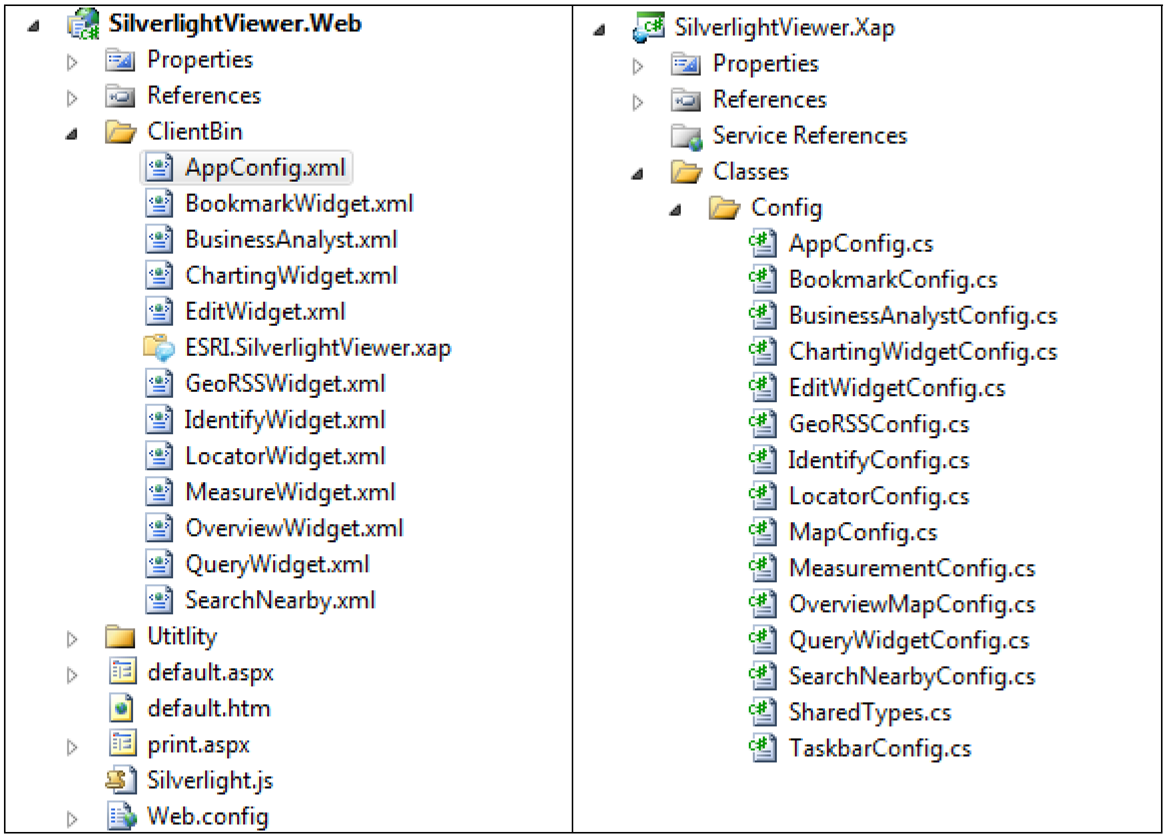
The creation of a new widget is similar as the creation of a custom Silverlight UserControl, but a widget is based on class WidgetBase, not class UserControl. An easy way to create a new widget is to copy the code of an existing widget, e.g. LocatorWidget, rename it, and then modify the code. The base namespace must be referred. Below is an example:

```
<base:WidgetBase x:Class="ESRI.SilverlightViewer.UIWidget.LocatorWidget"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:base="clr-namespace:ESRI.SilverlightViewer.Widget"
    ClearButtonImage="../Images/buttons/btn_clear.png">
    ⋮
</base:WidgetBase>

public partial class LocatorWidget : WidgetBase
{
    private GeocodeTool geocodeTool = null;
    private LocatorConfig widgetConfig = null;
    ⋮
}
```

Below are a few of important points about a widget:

- If a widget has an XML configuration file, a corresponding configuration class that parses the XML file must be created. A widget's XML configuration file is stored in the SilverlightViewer.Web\ClientBin folder and its configuration XML-parsing class is created in the "Classes\config" folder. Please look screenshots below:



```
SilverlightViewer.Web
    Properties
    References
    ClientBin
        AppConfig.xml
        BookmarkWidget.xml
        BusinessAnalyst.xml
        ChartingWidget.xml
        EditWidget.xml
        ESRI.SilverlightViewer.xap
        GeoRSSWidget.xml
        IdentifyWidget.xml
        LocatorWidget.xml
        MeasureWidget.xml
        OverviewWidget.xml
        QueryWidget.xml
        SearchNearby.xml
    Utitlity
    default.aspx
    default.htm
    print.aspx
    Silverlight.js
    Web.config
```

```
SilverlightViewer.Xap
    Properties
    References
    Service References
    Classes
        Config
            AppConfig.cs
            BookmarkConfig.cs
            BusinessAnalystConfig.cs
            ChartingWidgetConfig.cs
            EditWidgetConfig.cs
            GeoRSSConfig.cs
            IdentifyConfig.cs
            LocatorConfig.cs
            MapConfig.cs
            MeasurementConfig.cs
            OverviewMapConfig.cs
            QueryWidgetConfig.cs
            SearchNearbyConfig.cs
            SharedTypes.cs
            TaskbarConfig.cs
```

- If a widget has an XML configuration file, function **OnDownloadConfigXMLCompleted** must be overridden to get the widget configuration and the widget's properties can be initialized after loading configuration is completed if they are based on the XML file.

- If a widget needs to use DrawObject, e.g., drawing a rectangle, a line, or a point on the map, it is required to override the **ResetDrawObjectMode** function. Here is the sample code copied from LocatorWidget.

```
public override void ResetDrawObjectMode()
{
    // Enable DrawObject to enable reverse geocoding a click point on the map
    if (locatorRadio_Coords.IsChecked.Value)
    {
        this.DrawWidget = this.GetType();
        this.DrawObject.IsEnabled = true;
        this.DrawObject.DrawMode = DrawMode.Point;
        this.MapControl.Cursor = Cursors.Arrow;
    }
    else
    {
        WidgetManager.ResetDrawObject();
    }
}
```

5. How to Create a Distributable Plug-in Widget

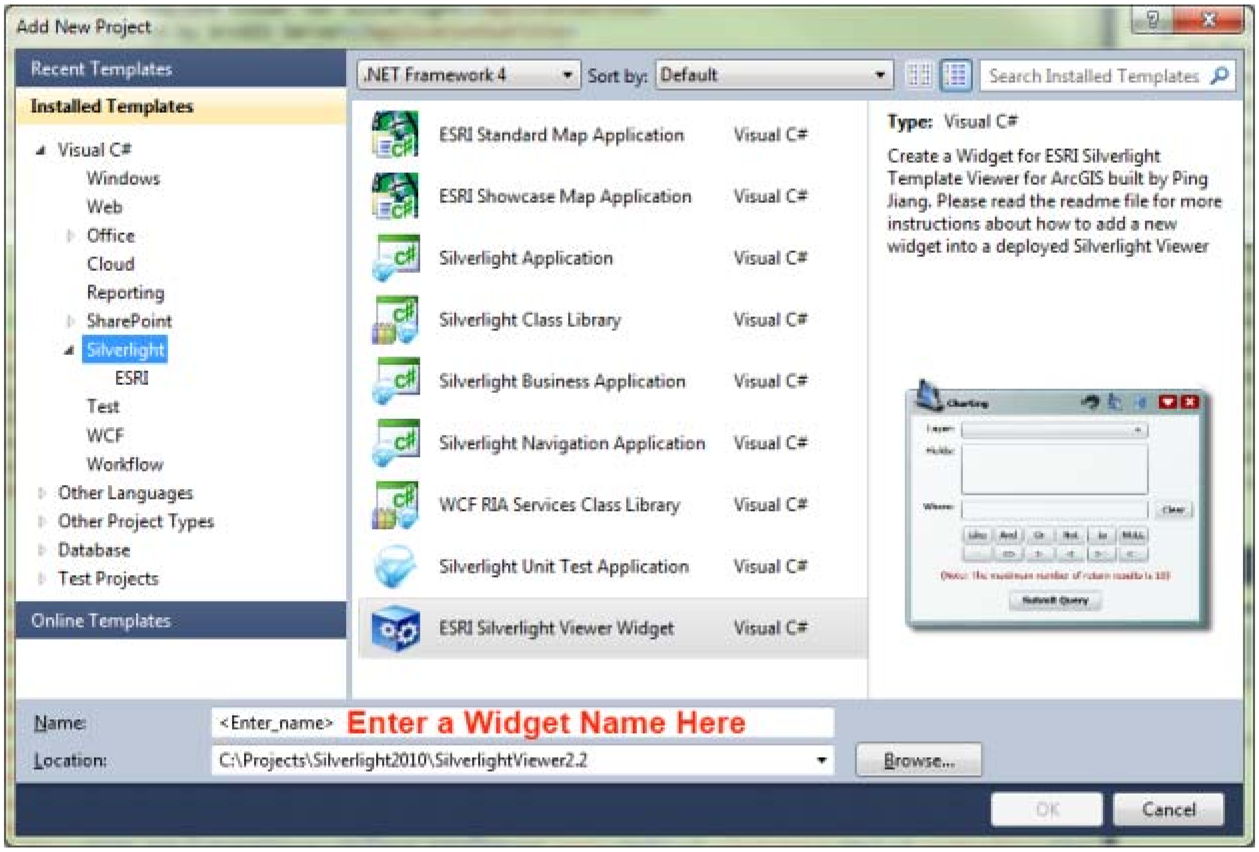
1) Create a "**Silverlight**" subfolder under "C:\Users\<YourLoginName>\Documents\Visual Studio 2010\Templates\ProjectTemplates\Visual C#\" if your computer system is Windows 7 or "C:\Documents and Settings\<YourLoginName>\My Documents\Visual Studio 2010\Templates\ProjectTemplates\Visual C#\" if your computer system is Windows XP

2) Copy **ESRISilverlightViewerWidget.Zip** into the Silverlight folder you just created

3) Open PluginWidgets.sln with Visual Studio 2010 (four existing Widget projects may be used as examples.) If you create a new solution, you **MUST** add the SilverlightViewer.Web project into the solution and set it as the StartUp project to make you able to debug the new widget project

4) Add a new project into the solution and select **ESRI Silverlight Viewer Widget** as the project template (see the picture below)

5) After the project is created, a widget page named MainPage.xaml will be created into the project



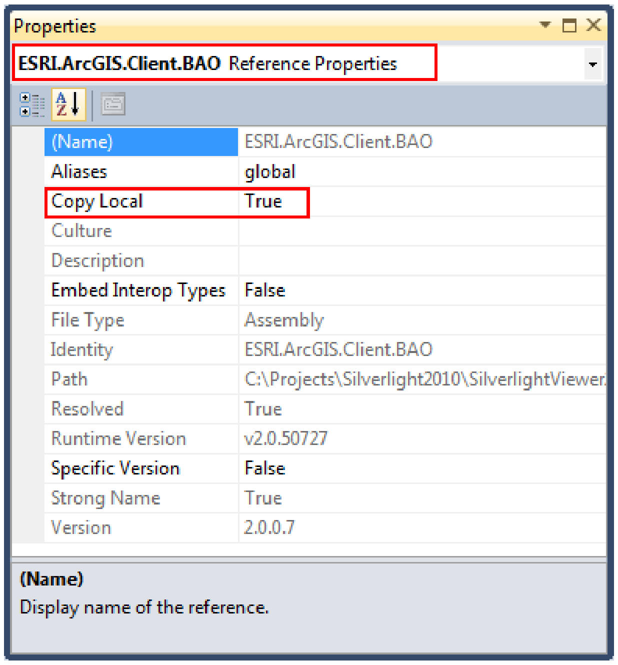
ESRI Standard Map Application	Visual C#
ESRI Showcase Map Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Business Application	Visual C#
Silverlight Navigation Application	Visual C#
WCF RIA Services Class Library	Visual C#
Silverlight Unit Test Application	Visual C#
ESRI Silverlight Viewer Widget	Visual C#

Type: Visual C#

Create a Widget for ESRI Silverlight Template Viewer for ArcGIS built by Ping Jiang. Please read the readme file for more instructions about how to add a new widget into a deployed Silverlight Viewer

Name: <Enter_name> Enter a Widget Name Here

Location: C:\Projects\Silverlight2010\SilverlightViewer2.2

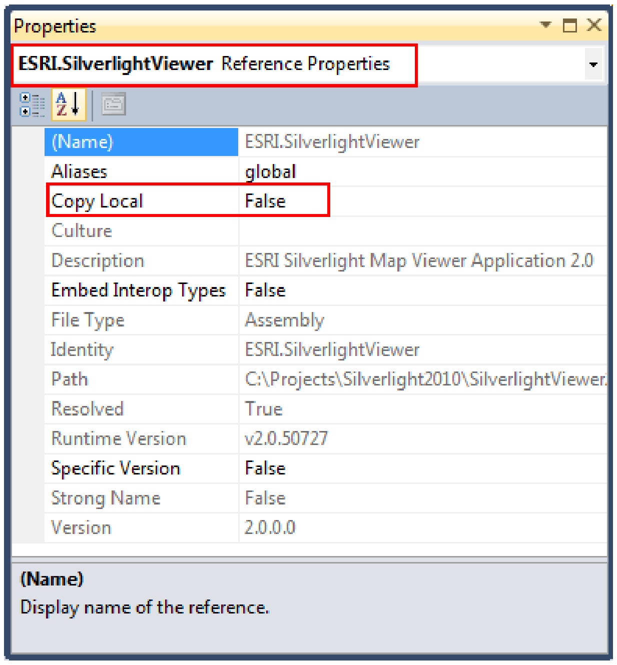


Properties

ESRI.ArcGIS.Client.BAO Reference Properties

(Name)	ESRI.ArcGIS.Client.BAO
Aliases	global
Copy Local	True
Culture	
Description	
Embed Interop Types	False
File Type	Assembly
Identity	ESRI.ArcGIS.Client.BAO
Path	C:\Projects\Silverlight2010\SilverlightViewer
Resolved	True
Runtime Version	v2.0.50727
Specific Version	False
Strong Name	True
Version	2.0.0.7

(Name)
Display name of the reference.

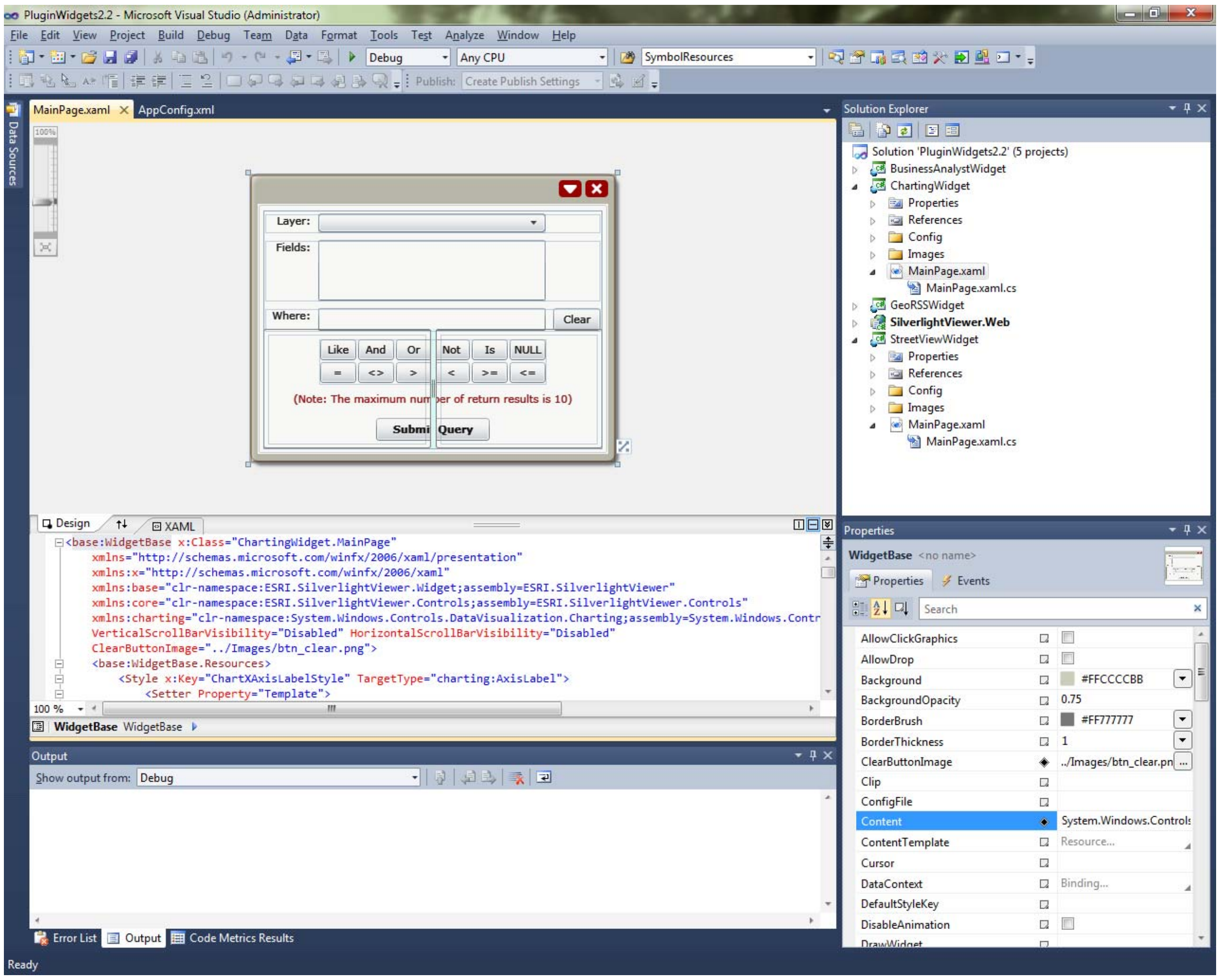


Properties

ESRI.SilverlightViewer Reference Properties

(Name)	ESRI.SilverlightViewer
Aliases	global
Copy Local	False
Culture	
Description	ESRI Silverlight Map Viewer Application 2.0
Embed Interop Types	False
File Type	Assembly
Identity	ESRI.SilverlightViewer
Path	C:\Projects\Silverlight2010\SilverlightViewer
Resolved	True
Runtime Version	v2.0.50727
Specific Version	False
Strong Name	False
Version	2.0.0.0

(Name)
Display name of the reference.



```
<base:WidgetBase x:Class="ChartingWidget.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:base="clr-namespace:ESRI.SilverlightViewer.Widget;assembly=ESRI.SilverlightViewer"
    xmlns:core="clr-namespace:ESRI.SilverlightViewer.Controls;assembly=ESRI.SilverlightViewer.Controls"
    xmlns:charting="clr-namespace:System.Windows.Controls.DataVisualization.Charting;assembly=System.Windows.Contr
    VerticalScrollBarVisibility="Disabled" HorizontalScrollBarVisibility="Disabled"
    ClearButtonImage="../Images/btn_clear.png">
    <base:WidgetBase.Resources>
        <Style x:Key="ChartXAxisLabelStyle" TargetType="charting:AxisLabel">
            <Setter Property="Template">
```

Properties

WidgetBase <no name>

Properties Events

AllowClickGraphics	☐
AllowDrop	☐
Background	#FFCCCCBB
BackgroundOpacity	0.75
BorderBrush	#FF777777
BorderThickness	1
ClearButtonImage	../Images/btn_clear.pn
Clip	
ConfigFile	
Content	System.Windows.Controls
ContentTemplate	Resource...
Cursor	
DataContext	Binding...
DefaultStyleKey	
DisableAnimation	☐
DrawWidget	

6) Afterward you may follow the instructions in section 4 to create a new widget

7) To be able to debug the widget project, a post-build event must be added into the project properties. If the widget project references any other assemblies than those that the SilverlightViewer.Xap project has referenced, the **Copy Local** property of the new assembly references must be set to **true** and the widget project's XAP output file must be copied into SilverlightViewer.Web\ClientBin. Then the post-build event command line must like this (e.g. BusinessAnalysWidget references ESRI.ArcGIS.Client.BAO and ChartingWidget references System.Windows.Controls.DataVisualization.Toolkit):

```
echo COPY $(OutDir)$(ProjectName).xap
cd $(ProjectDir)
copy $(OutDir)*.xap  ..\SilverlightViewer.Web\ClientBin
echo DONE
```

and the configuration of the plug-in widget in AppConfig.xml must like this:

```
<Widget xapFile="ChartingWidget.xap"
className="ChartingWidget.dll;ChartingWidget.MainPage" title="Charting"
openInitial="false" hasGraphics="true" initialTop="300" initialLeft="500"
icon="../Images/i_barchart.png" configFile="ChartingWidget.xml"/>
```

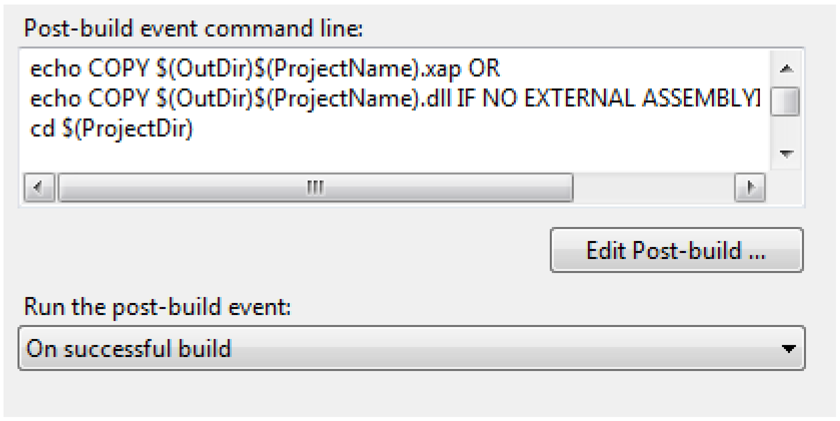
*****Note: if you deploy the application onto Windows Server 2003 (IIS 6.0) or a Java Web Server, the XAP file MUST be used**

If the widget project does not reference any other assemblies and **the application will be running on .NET Framwork 4.0/IIS 7.0 or later**, the DLL output file is only needed to copy into SilverlightViewer.Web\ClientBin. Then the post-build event command line can be set like:

```
cd $(ProjectDir)
copy $(OutDir)$(ProjectName).dll  ..\SilverlightViewer.Web\ClientBin
echo DONE
```

and the configuration of the plug-in widget in AppConfig.xml likes this:

```
<Widget className="GeoRSSWidget.dll;GeoRSSWidget.MainPage" title="Earthquakes"
openInitial="false" hasGraphics="true" initialTop="350" initialLeft="300"
icon="../Images/i_georss.png" configFile="GeoRSSWidget.xml"/>
```



```
echo COPY $(OutDir)$(ProjectName).xap OR
echo COPY $(OutDir)$(ProjectName).dll IF NO EXTERNAL ASSEMBLYI
cd $(ProjectDir)
```

Run the post-build event: On successful build

6. How to Disable or Customize a Widget's Animation

If you want to disable animation for a widget, you may add **DisableAnimation="True"** into the widget XAML, or set **widget.DisableAnimation = true** in the **LoadWidgets** function in the MapPage class (MapPage.cs) file, for example,

```
<base:WidgetBase x:Class="ESRI.SilverlightViewer.UIWidget.BookmarkWidget"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:base="clr-namespace:ESRI.SilverlightViewer.Widget" DisableAnimation="True">
     ⋮
</base:WidgetBase>
```

If you want to customize a widget's animation, you may override the two functions following in the widget base class:

```
protected virtual void CreateOpenCloseStoryboard(bool initialized)
{
    Random RandMaker = new Random();
    int axis = RandMaker.Next(-1, 1);
    int direct = RandMaker.Next(-1, 1);
    double offset = axis * direct * 600;
    if (offset == 0) offset = -600;

    if (this.Projection == null || !(this.Projection is PlaneProjection))
    {
        PlaneProjection projection = new PlaneProjection();
        this.Projection = projection;

        if (initialized)
        {
            switch (axis)
            {
               case -1: projection.RotationX = 270; break;
               case 0: projection.RotationZ = 270; break;
               case 1: projection.RotationY = 270; break;
            }

            switch (direct)
            {
               case -1: projection.LocalOffsetX = offset; break;
               case 0: projection.LocalOffsetZ = offset; break;
               case 1: projection.LocalOffsetY = offset; break;
            }
        }
    }

    string sGUID = Guid.NewGuid().ToString();
    CloseStoryboardName = "Close_" + sGUID;
    OpenStoryboardName = "Open_" + sGUID;

    SaveOpenCloseStoryboard(OpenStoryboardName, axis, direct, 0, 0);
    SaveOpenCloseStoryboard(CloseStoryboardName, axis, direct, 270, offset);
}

protected virtual void SaveOpenCloseStoryboard(string name, int axis, int direct,
double rotate, double offset)
{
    Storyboard sb = new Storyboard();
    sb.SetValue(FrameworkElement.NameProperty, name);
    if (name == OpenStoryboardName) sb.Completed += new
EventHandler(OpenAnimation_Completed);
    if (name == CloseStoryboardName) sb.Completed += new
EventHandler(CloseAnimation_Completed);

    string rotationAxis = "RotationX";
    switch (axis)
    {
       case -1: rotationAxis = "RotationX"; break;
       case 0: rotationAxis = "RotationZ"; break;
       case 1: rotationAxis = "RotationY"; break;
    }

    DoubleAnimation dbAnimR = new DoubleAnimation() {
        To = rotate, Duration = TimeSpan.FromSeconds(0.75), AutoReverse = false };
    Storyboard.SetTarget(dbAnimR, this.Projection);
    Storyboard.SetTargetProperty(dbAnimR, new PropertyPath(rotationAxis));
    sb.Children.Add(dbAnimR);

    string offsetDirect = "LocalOffsetX";
    switch (direct)
    {
       case -1: offsetDirect = "LocalOffsetX"; break;
       case 0: offsetDirect = "LocalOffsetZ"; break;
       case 1: offsetDirect = "LocalOffsetY"; break;
    }

    DoubleAnimation dbAnimO = new DoubleAnimation() {
        To = offset, Duration = TimeSpan.FromSeconds(0.75), AutoReverse = false };
    Storyboard.SetTarget(dbAnimO, this.Projection);
    Storyboard.SetTargetProperty(dbAnimO, new PropertyPath(offsetDirect));
    sb.Children.Add(dbAnimO);
    this.Resources.Add(name, sb);
}
```