IBM ILOG Diagram for .NET V2.0

Tutorials

June 2009

Tutorials

Tutorials

The IBM® ILOG® Diagram for .NET tutorials provide step-by-step introductions to the fundamental areas of programming for the IBM ILOG Diagram for .NET framework. Each tutorial provides a detailed walkthrough of the steps required to write, build and run small sample applications.

In This Section

Creating an IBM ILOG Diagram for .NET Windows Forms Application and User Symbol

Walks you through the process of creating your first IBM ILOG Diagram for .NET Windows® Forms application.

Creating your First IBM ILOG Diagram for .NET AJAX Web Site

Walks you through the process of creating your first IBM ILOG Diagram for .NET AJAX Web Site.

Related Section

Getting Started

Provides a hands-on introduction and overview to the IBM ILOG Diagram for .NET framework.

Creating an IBM ILOG Diagram for .NET Windows Forms Application and User Symbol

This tutorial walks you through the creation of your first IBM® ILOG® Diagram for .NET Windows® Forms application and user symbol.

In This Section

Creating your First IBM ILOG Diagram for .NET Windows Forms Application

Introduces the basics of the Diagram Designer of IBM ILOG Diagram for .NET for creating a new diagram.

Creating a New User Symbol for Your Application

Walks you through the creation of a graphic object that represents a traffic light.

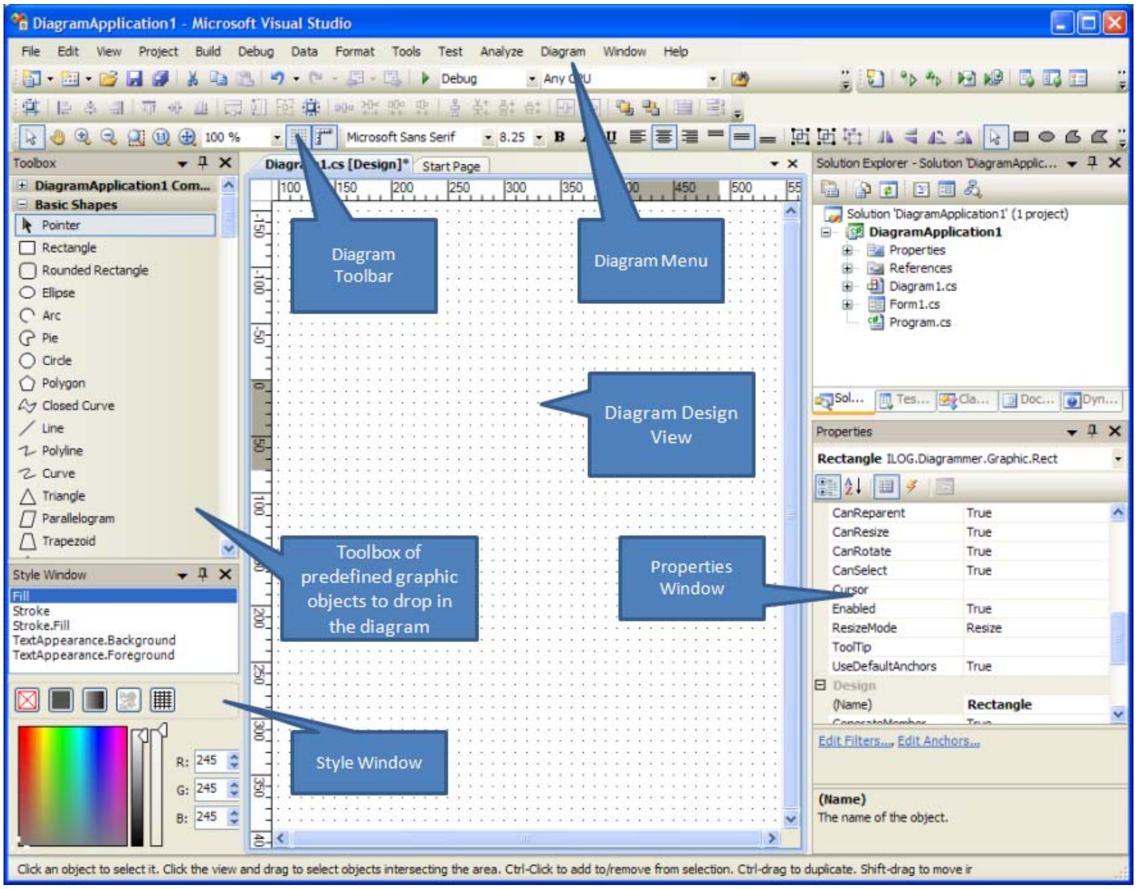
Creating your First IBM ILOG Diagram for .NET Windows Forms Application

Several elements in Visual Studio will help you create your diagram:

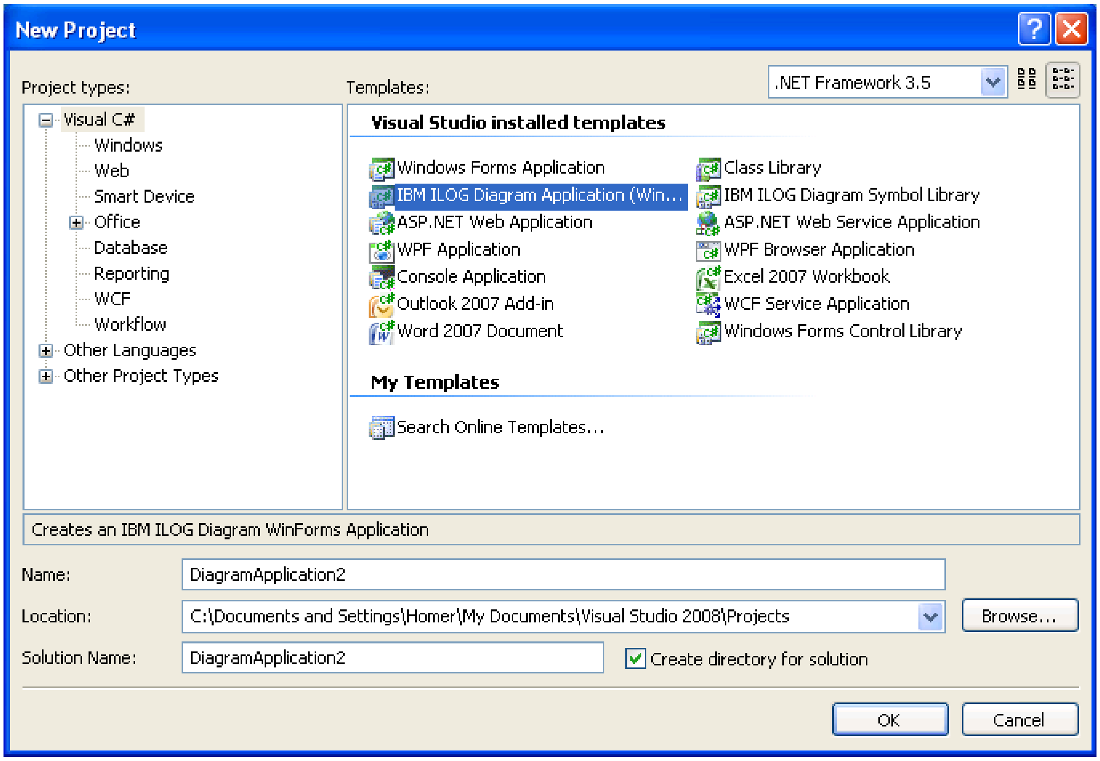
- Diagram Design view: is where you design your diagram or user symbol by dragging the graphic objects from the Toolbox.

- Toolbox: presents a set of predefined graphic objects that you can drag to the Diagram Design View.
- Properties Window: displays the properties of the graphic objects that you have selected in the Diagram Design view. From the Properties view you can modify all the properties of the graphic objects and access to the events fired by the graphic objects.
- Diagram Toolbar: is a Visual Studio toolbar that enables operations like controlling the zoom of the Diagram Design view, creating some graphic objects, grouping objects, changing the text properties of the selected graphic objects and more. You can open the Diagram Toolbar by selecting View>Toolbars>IBM ILOG Diagram from the menu bar.
- Diagram Menu: is a Visual Studio menu that presents several commands like for example zoom or import a file in your diagram.
- Style Window: is a view dedicated to changing the style of graphic objects. The Style Window presents the **Fill** and **Stroke** properties of the selected graphic objects and allows you to change them easily.

The following illustration shows the different elements you can use inside Visual Studio to create diagrams or symbols.


DiagramApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Tools Test Analyze Diagram Window Help
Diagram Toolbar
Diagram Menu
Diagram Design View
Toolbox of predefined graphic objects to drop in the diagram
Properties Window
Style Window
Toolbox
Basic Shapes
Pointer
Rectangle
Rounded Rectangle
Ellipse
Arc
Pie
Circle
Polygon
Closed Curve
Line
Polyline
Curve
Triangle
Parallelogram
Trapezoid
Style Window
Fill
Stroke
Stroke.Fill
TextAppearance.Background
TextAppearance.Foreground
Solution Explorer - Solution 'DiagramApplic...
Solution 'DiagramApplication1' (1 project)
DiagramApplication1
Properties
References
Diagram1.cs
Form1.cs
Program.cs
Properties
Rectangle ILOG.Diagrammer.Graphic.Rect
CanReparent True
CanResize True
CanRotate True
CanSelect True
Cursor
Enabled True
ResizeMode Resize
ToolTip
UseDefaultAnchors True
Design
(Name) Rectangle
Edit Filters... Edit Anchors...
(Name)
The name of the object.
Click an object to select it. Click the view and drag to select objects intersecting the area. Ctrl-Click to add to/remove from selection. Ctrl-drag to duplicate. Shift-drag to move it

1. Launch Visual Studio .NET
2. Select File>New Project from the main menu. The New Project window opens.
3. Select Visual C# under Project Types and IBM ILOG Diagram Application (WinForms) under Visual Studio installed templates.


New Project

Project types:
- Visual C#
  - Windows
  - Web
  - Smart Device
  - Office
  - Database
  - Reporting
  - WCF
  - Workflow
- Other Languages
- Other Project Types

Templates: .NET Framework 3.5

Visual Studio installed templates

Windows Forms Application	Class Library
IBM ILOG Diagram Application (Win...	IBM ILOG Diagram Symbol Library
ASP.NET Web Application	ASP.NET Web Service Application
WPF Application	WPF Browser Application
Console Application	Excel 2007 Workbook
Outlook 2007 Add-in	WCF Service Application
Word 2007 Document	Windows Forms Control Library

My Templates

Search Online Templates...

Creates an IBM ILOG Diagram WinForms Application

Name: DiagramApplication2

Location: C:\Documents and Settings\Homer\My Documents\Visual Studio 2008\Projects

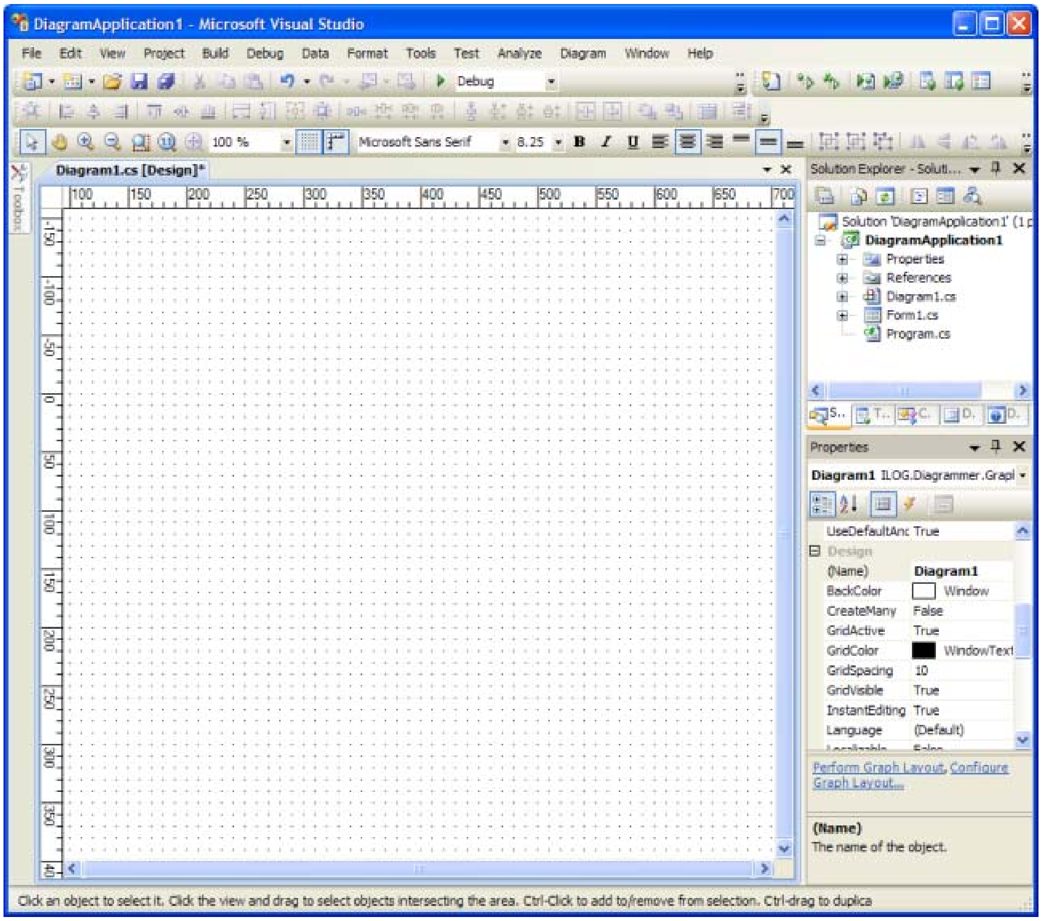
Solution Name: DiagramApplication2

[✓] Create directory for solution

This project template creates a fully functional IBM® ILOG® Diagram for .NET Windows® Forms application that contains the following files:

- `Form1.cs` that defines the main Form of the application. This Form contains a DiagramView object that is the IBM ILOG Diagram for .NET WinForm control to display a diagram. This diagram view is already configured to display the diagram that you will create.
- `Diagram1.cs` that defines the diagram to display in the **DiagramView** control. The `Diagram1.cs` file defines a new class named Diagram1 that is a subclass of the Group class of IBM ILOG Diagram for .NET. A **Group** object is a graphic object that simply displays its children.

When the project is created, the `Diagram1.cs` file is opened in Design mode. Now you are able to define the content of your diagram directly within Visual Studio by dragging graphic objects to the Diagram Designer.


DiagramApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Tools Test Analyze Diagram Window Help
Debug
100 %
Microsoft Sans Serif
8.25
Toolbox
Diagram1.cs [Design]*
Solution Explorer - Soluti...
Solution 'DiagramApplication1' (1 p
DiagramApplication1
Properties
References
Diagram1.cs
Form1.cs
Program.cs
Properties
Diagram1 ILOG.Diagrammer.Grapl
Perform Graph Layout, Configure Graph Layout...

UseDefaultAnc	True
Design	
(Name)	Diagram1
BackColor	Window
CreateMany	False
GridActive	True
GridColor	WindowText
GridSpacing	10
GridVisible	True
InstantEditing	True
Language	(Default)
Localizable	False

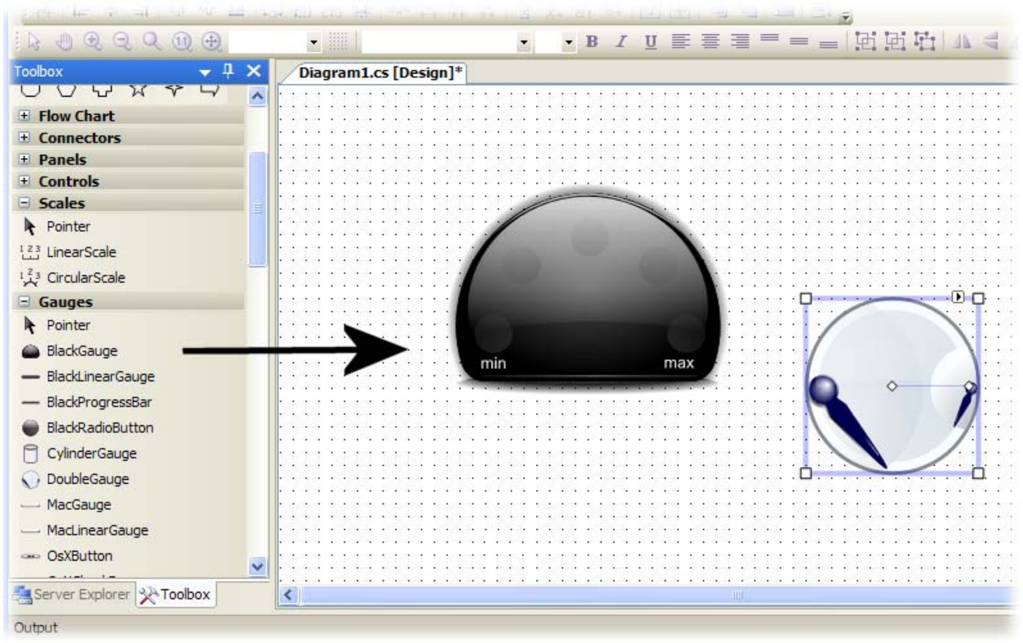
(Name)
The name of the object.

Click an object to select it. Click the view and drag to select objects intersecting the area. Ctrl-Click to add to/remove from selection. Ctrl-drag to duplica

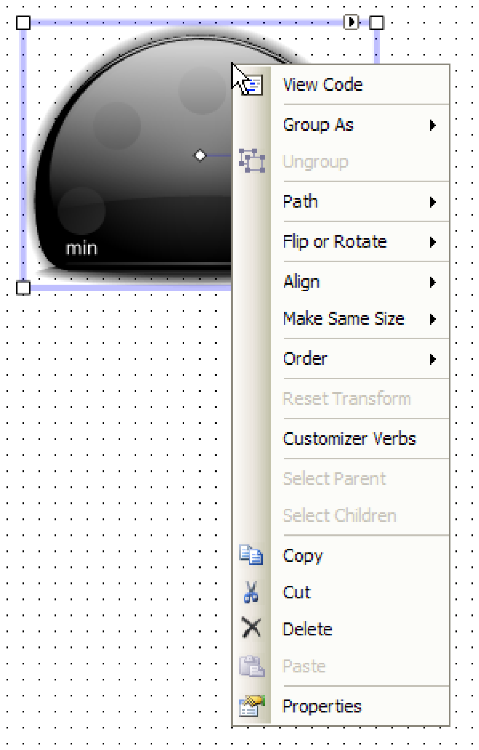
You can drag any graphic object from the Toolbox to the Diagram Design view. Once dropped, you can use the Properties Window to modify the properties of the selected object. At any time you can undo your work by using the Undo button on the toolbar.

To learn more about the Diagram Designer inside Visual Studio see *Building Diagrams and User Symbols Inside Visual Studio*.

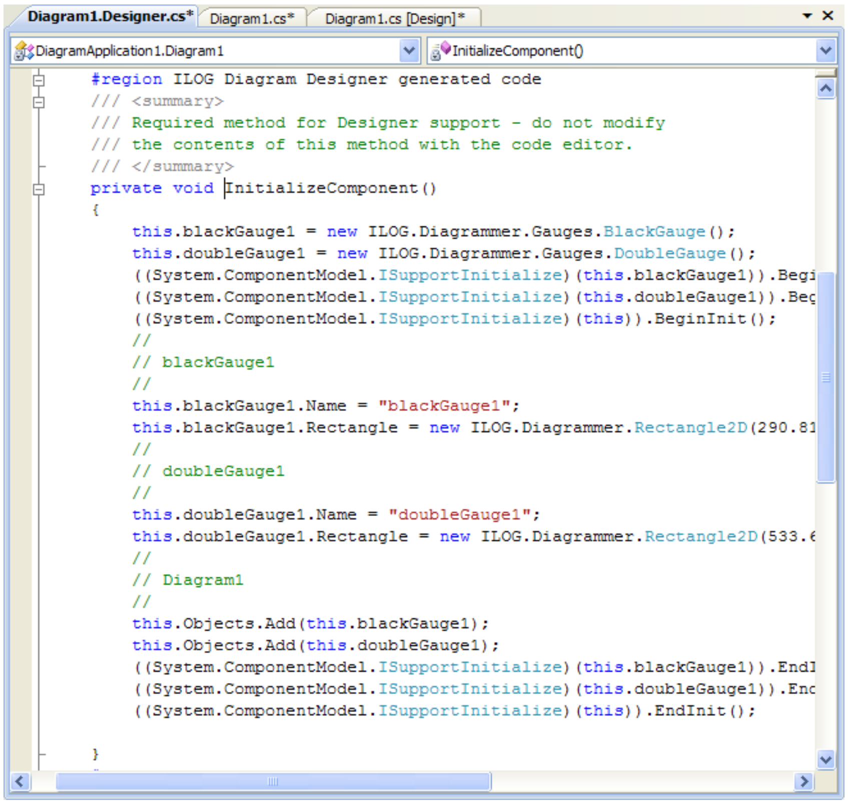
In this example you are going to drag two predefined graphic objects from the Toolbox: **BlackGauge** and **DoubleGauge**.



4. Right-click a graphic object to open a contextual menu that presents the common actions that you can apply on the selected graphic object. For example, Cut/Copy/Paste, alignment options, z-ordering options and so on.



As you drop graphic objects in your diagram, the Diagram Designer generates the code that corresponds to your diagram in the `Diagram1.Designer.cs` file.



```
#region ILOG Diagram Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.blackGauge1 = new ILOG.Diagrammer.Gauges.BlackGauge();
    this.doubleGauge1 = new ILOG.Diagrammer.Gauges.DoubleGauge();
    ((System.ComponentModel.ISupportInitialize)(this.blackGauge1)).Begi
    ((System.ComponentModel.ISupportInitialize)(this.doubleGauge1)).Beg
    ((System.ComponentModel.ISupportInitialize)(this)).BeginInit();
    //
    // blackGauge1
    //
    this.blackGauge1.Name = "blackGauge1";
    this.blackGauge1.Rectangle = new ILOG.Diagrammer.Rectangle2D(290.81
    //
    // doubleGauge1
    //
    this.doubleGauge1.Name = "doubleGauge1";
    this.doubleGauge1.Rectangle = new ILOG.Diagrammer.Rectangle2D(533.6
    //
    // Diagram1
    //
    this.Objects.Add(this.blackGauge1);
    this.Objects.Add(this.doubleGauge1);
    ((System.ComponentModel.ISupportInitialize)(this.blackGauge1)).EndI
    ((System.ComponentModel.ISupportInitialize)(this.doubleGauge1)).End
    ((System.ComponentModel.ISupportInitialize)(this)).EndInit();

}
```

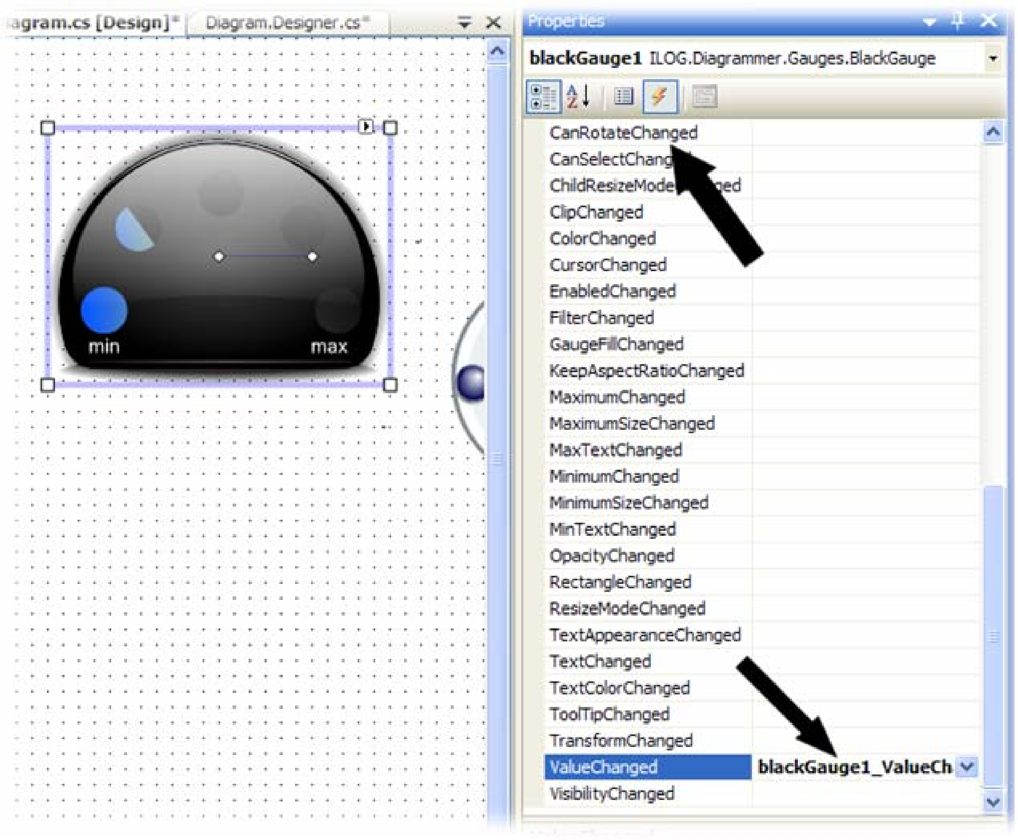
Now you have dropped two objects: BlackGauge and DoubleGauge. The **InitializeComponent** method, generated by the Diagram Designer, initializes the value of these two objects.

You may want to complement the application by implementing some event handler. The BlackGauge object defines a ValueChanged event fired when the Value property of the gauge changes.

5. Select the BlackGauge.

6. Select the event Tab in the Properties Window. All events defined by this graphic object appear.

7. Double click the **ValueChanged** event.



blackGauge1 ILOG.Diagrammer.Gauges.BlackGauge

CanRotateChanged	
CanSelectChanged	
ChildResizeModeChanged	
ClipChanged	
ColorChanged	
CursorChanged	
EnabledChanged	
FilterChanged	
GaugeFillChanged	
KeepAspectRatioChanged	
MaximumChanged	
MaximumSizeChanged	
MaxTextChanged	
MinimumChanged	
MinimumSizeChanged	
MinTextChanged	
OpacityChanged	
RectangleChanged	
ResizeModeChanged	
TextAppearanceChanged	
TextChanged	
TextColorChanged	
ToolTipChanged	
TransformChanged	
ValueChanged	blackGauge1_ValueCh
VisibilityChanged	

8. Code the event handler. Here you simply code the fact that the second gauge value will be modified when the BlackGauge changes.

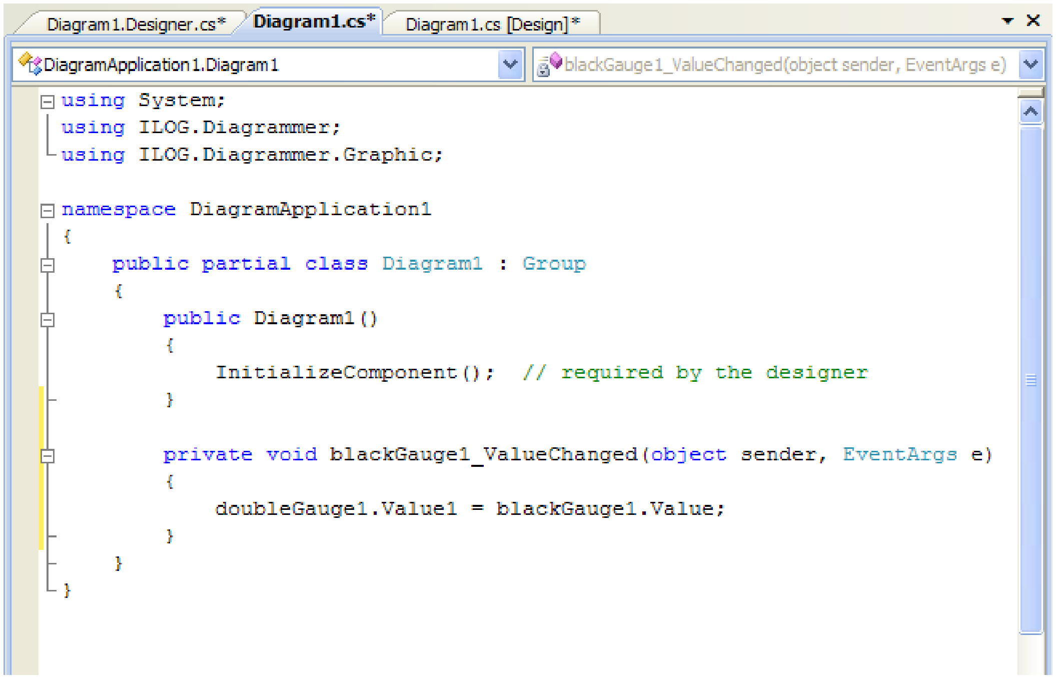
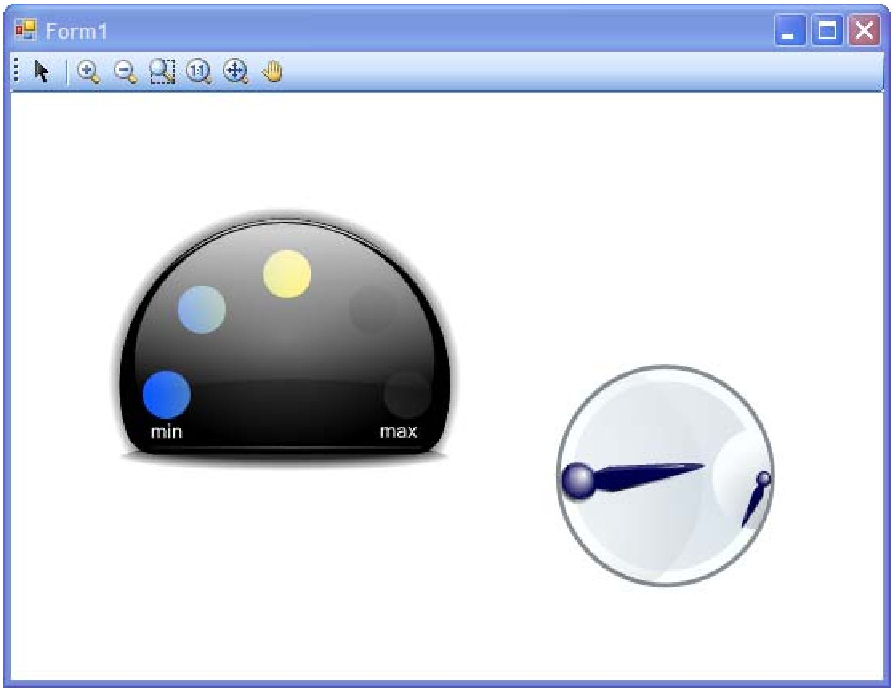


Diagram1.Designer.cs*
Diagram1.cs*
Diagram1.cs [Design]*
DiagramApplication1.Diagram1
blackGauge1_ValueChanged(object sender, EventArgs e)

```
using System;
using ILOG.Diagrammer;
using ILOG.Diagrammer.Graphic;

namespace DiagramApplication1
{
    public partial class Diagram1 : Group
    {
        public Diagram1()
        {
            InitializeComponent();  // required by the designer
        }

        private void blackGauge1_ValueChanged(object sender, EventArgs e)
        {
            doubleGauge1.Value1 = blackGauge1.Value;
        }
    }
}
```

9. Run the application. You will see that dragging the mouse over the BlackGauge will change the value of the needle of the second gauge.



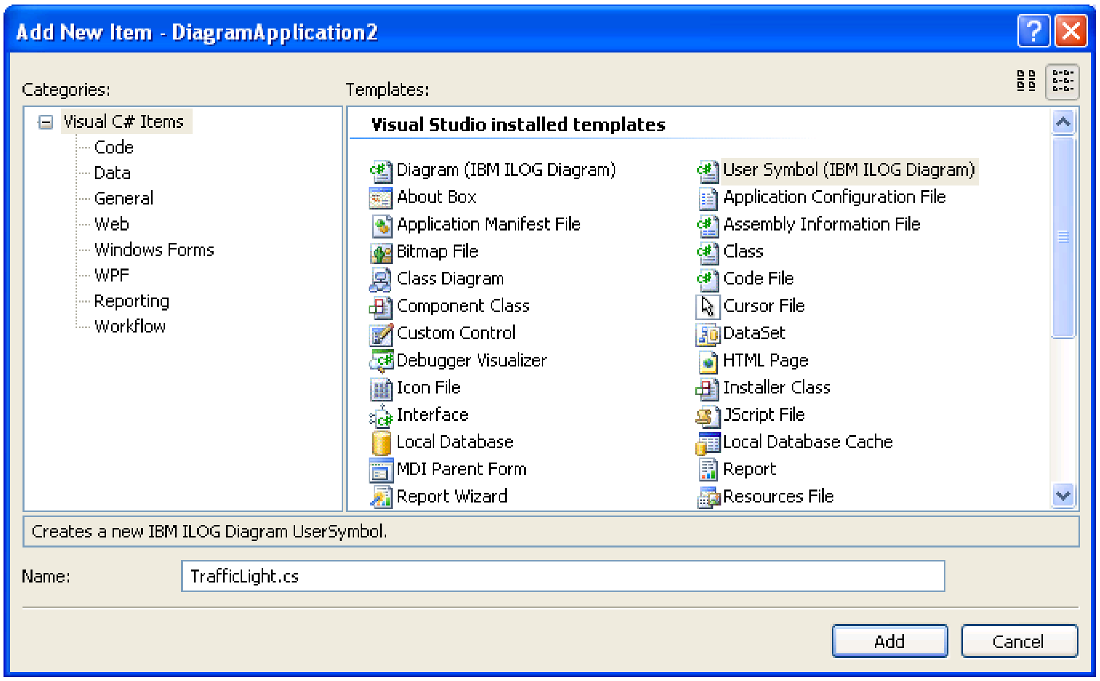
Now that you have created a basic IBM ILOG Diagram for .NET WinForms application, you can continue by connecting the gauges to real life data using the power of Visual Studio.NET to query data from a database or from a Web service.

Creating a New User Symbol for Your Application

The IBM® ILOG® Diagram for .NET Diagram Designer not only allows you to create a diagram using predefined graphic objects, it also allows you to create new graphic objects that you will use in your diagrams later on.

In this example you are going to create a graphic object that represents a traffic light.

1. In the Solution Explorer, right-click the project you have created in *Creating your First IBM ILOG Diagram for .NET Windows Forms Application*.
2. Choose Add>New Item.
3. In the Add New Item window choose the User Symbol (IBM ILOG Diagram) template.
4. In the Name field enter `TrafficLight.cs` as the name of the new item.



Creates a new IBM ILOG Diagram UserSymbol.

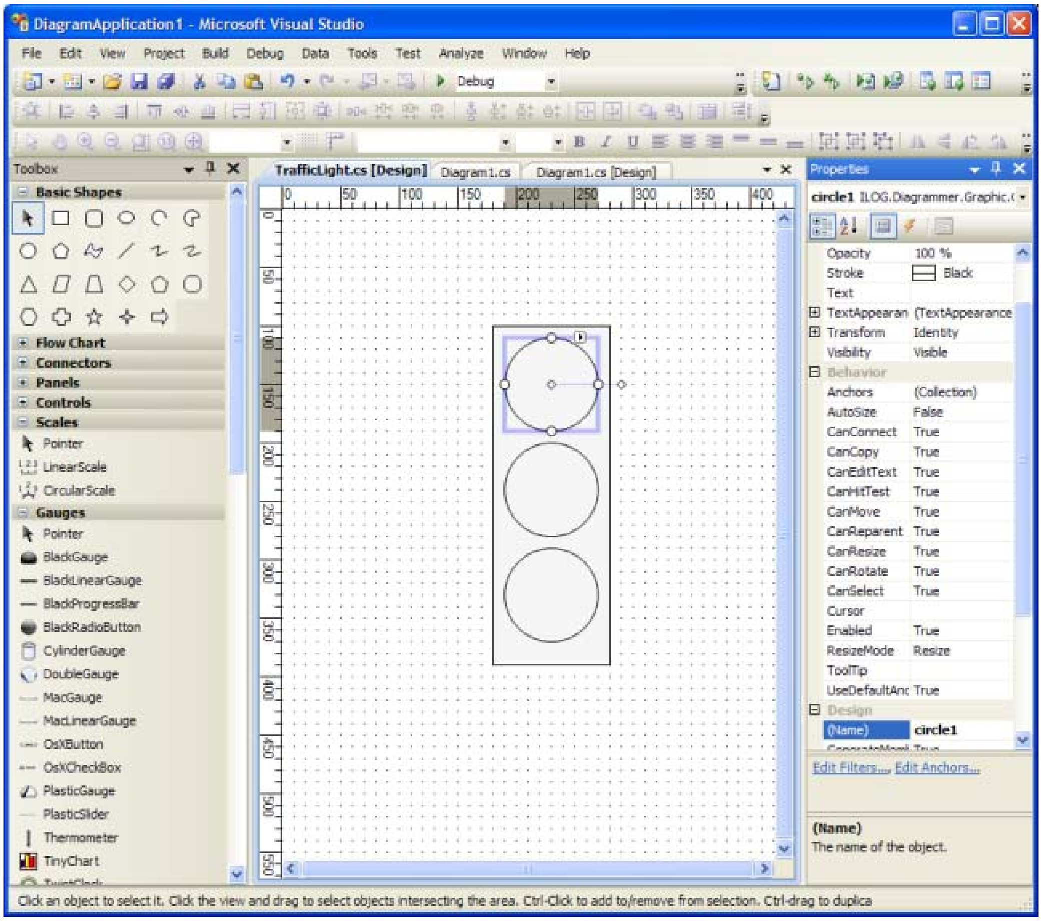
Name: TrafficLight.cs

This new graphical representation will be a subclass of the IBM ILOG Diagram for .NET class named UserSymbol. You will refer to this type of new graphical representation as "user symbols" or simply "symbols".

The template creates a file named `TrafficLight.cs` that defines a new TrafficLight class.

Once the file is created, the new symbol opens in Design view.

5. In Design view, drag four objects from the Toolbox: a rectangle and three ellipses on top of the rectangle.



Opacity	100 %
Stroke	Black
Text	
TextAppearan	(TextAppearance
Transform	Identity
Visibility	Visible
Behavior	
Anchors	(Collection)
AutoSize	False
CanConnect	True
CanCopy	True
CanEditText	True
CanHitTest	True
CanMove	True
CanReparent	True
CanResize	True
CanRotate	True
CanSelect	True
Cursor	
Enabled	True
ResizeMode	Resize
ToolTip	
UseDefaultAnc	True
Design	
(Name)	circle1

(Name)
The name of the object.

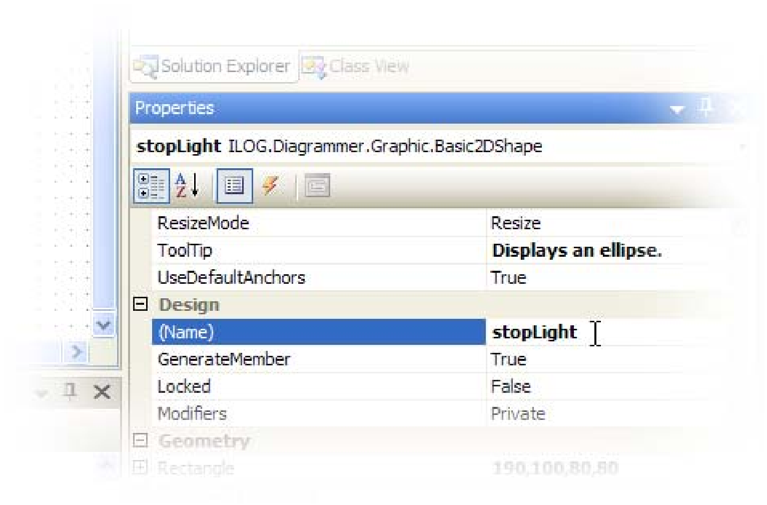
Click an object to select it. Click the view and drag to select objects intersecting the area. Ctrl-Click to add to/remove from selection. Ctrl-drag to duplica

This gives the beginning of a traffic light.

6. Change the name of the three circles.

Select one graphic object at a time and open the Properties Window. In the Name property field enter the names for the three objects: stopLight, amberLight and goLight.

The names of the objects define the names of the class members that will be generated in the code. The comprehensive names are necessary when you will write code.



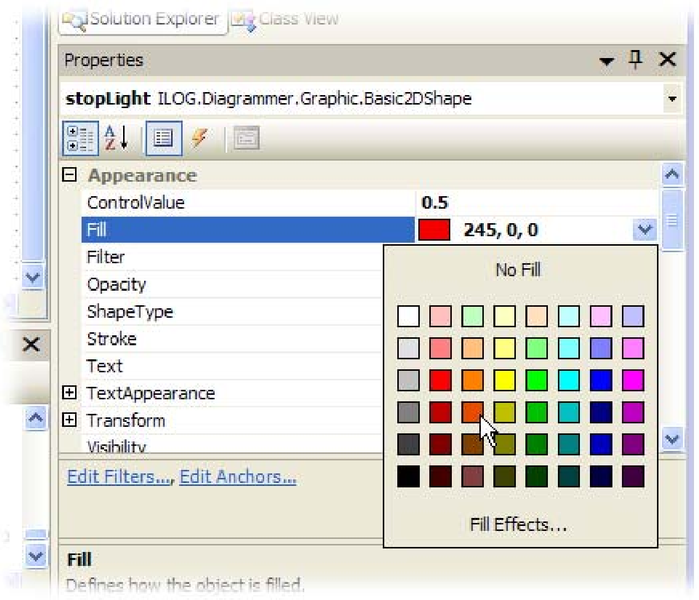
stopLight ILOG.Diagrammer.Graphic.Basic2DShape

ResizeMode	Resize
ToolTip	**Displays an ellipse.**
UseDefaultAnchors	True
Design	
(Name)	**stopLight**
GenerateMember	True
Locked	False
Modifiers	Private
Geometry	
Rectangle	190,100,80,80

Now you can improve the appearance of the symbol by changing the color for each object.

7. With the object selected, go to the Properties Window and edit the **Fill** property of each circle to specify a new color. A popup menu opens.

   Choose red for the top ellipse, yellow for the middle ellipse and green for the bottom ellipse.



Properties

stopLight ILOG.Diagrammer.Graphic.Basic2DShape

Appearance	
ControlValue	0.5
Fill	245, 0, 0
Filter	
Opacity	
ShapeType	
Stroke	
Text	
TextAppearance	
Transform	
Visibility	

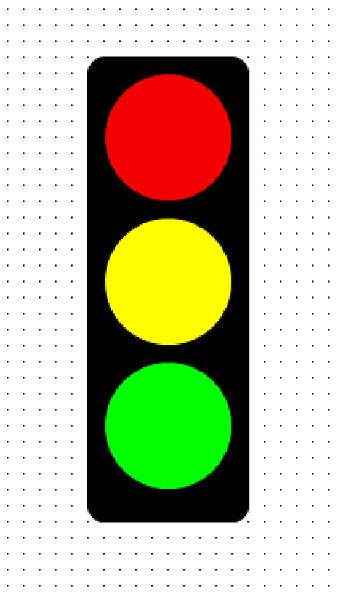
Edit Filters..., Edit Anchors...

Fill

Defines how the object is filled.

8. Change the **Fill** property of the rectangle to black.
9. Add corners to the rectangle by setting the Radius property to 10.

The following illustration shows the resulting symbol:



The graphical part of the symbol is now completed. Now you need to code a C# property that defines the traffic light state.

10. To open the code corresponding to the symbol, right-click in the diagram Design view and select View Code.

11. Add a new C# enumeration that defines the 3 states of the traffic light.

```
public enum TrafficLightState
{
    Stop,
    Go,
    Amber
}
```

12. And a new C# property named State that represents the state of the traffic light for the `TrafficLight` class.

```
private TrafficLightState state = TrafficLightState.Go;

public TrafficLightState State
{
    get
    {
       return state;
    }

    set
    {
            if (state != value)
            {
               state = value;
               stopLight.Fill  = new SolidFill(
                     value == TrafficLightState.Stop ? Color.Red :
   Color.LightGray);
               goLight.Fill    = new SolidFill(
                     value == TrafficLightState.Go ? Color.Green :
   Color.LightGray);
               amberLight.Fill = new SolidFill(
                     value == TrafficLightState.Amber ? Color.Yellow :
   Color.LightGray);


             }
         }
     }
```

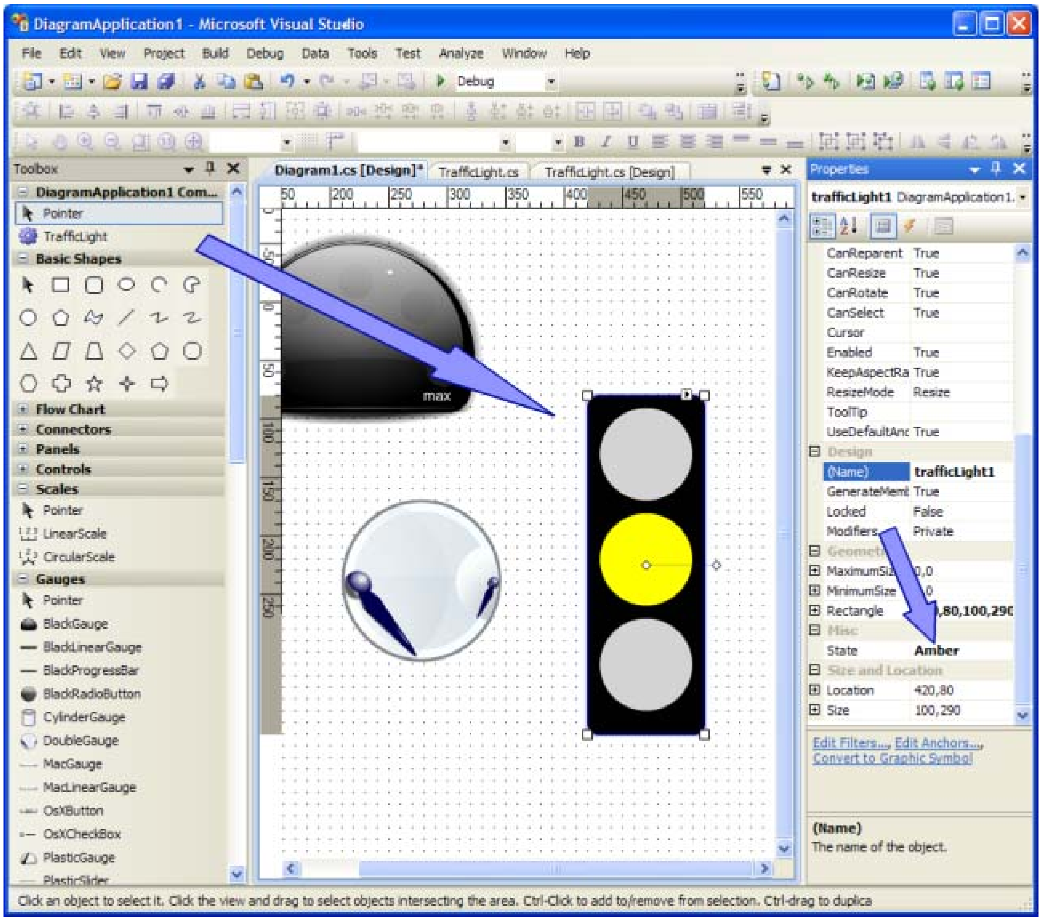
You simply change the Fill property of the ellipses depending on the state of the traffic light. For this, use a SolidFill object that allows you to fill an object with a basic color.

The full code in the `TrafficLight.cs` file is now:

```
using System;
using ILOG.Diagrammer;
using ILOG.Diagrammer.Graphic;
using System.Drawing;

namespace DiagramApplication1
{
    public enum TrafficLightState
    {
        Stop,
        Go,
        Amber
    }

    public partial class TrafficLight : UserSymbol
    {
        TrafficLightState state = TrafficLightState.Go;

        public TrafficLight()
        {
            InitializeComponent();                 // required by the designer

            // Initialize the default value.
            State = TrafficLightState.Stop;
        }

        public TrafficLightState State
        {
            get
            {
                return state;
            }

            set
            {
                if (state != value)
                {
                    state = value;

                    stopLight.Fill  = new SolidFill(
                        value == TrafficLightState.Stop ? Color.Red : 
Color.LightGray);
                    goLight.Fill    = new SolidFill(
                        value == TrafficLightState.Go ? Color.Green : 
Color.LightGray);
                    amberLight.Fill = new SolidFill(
                        value == TrafficLightState.Amber ? Color.Yellow : 
Color.LightGray);


                }
            }
        }
    }

}
```

13. Compile the new symbol that is fully functional.

Once your symbol is compiled, you can re-open the `Diagram1.cs` file. The new TrafficLight symbol will appear in the Toolbox. You can now drop a traffic light in your diagram and test the new State property.



CanReparent	True
CanResize	True
CanRotate	True
CanSelect	True
Cursor	
Enabled	True
KeepAspectRa	True
ResizeMode	Resize
ToolTip	
UseDefaultAnc	True
Design	
(Name)	trafficLight1
GenerateMemt	True
Locked	False
Modifiers	Private
Geometry	
MaximumSiz	0,0
MinimumSize	0
Rectangle	,80,100,290
Misc	
State	Amber
Size and Location	
Location	420,80
Size	100,290

(Name)
The name of the object.

Click an object to select it. Click the view and drag to select objects intersecting the area. Ctrl-Click to add to/remove from selection. Ctrl-drag to duplica

You have learned how to create a basic user symbol. Thanks to the large number of predefined graphic objects of IBM ILOG Diagram for .NET you will be able to create very complex user symbols.

All the symbols that you create through the IBM ILOG Diagram for .NET Designer can be re-used later on in your application and combined with all the features of IBM ILOG Diagram for .NET. For example, you can use a symbol as the node of a graph, attaching links to a symbol. Note that the predefined gauges in the product have been created with the designer as well as the BPMN or the UML objects that you can see in the samples.

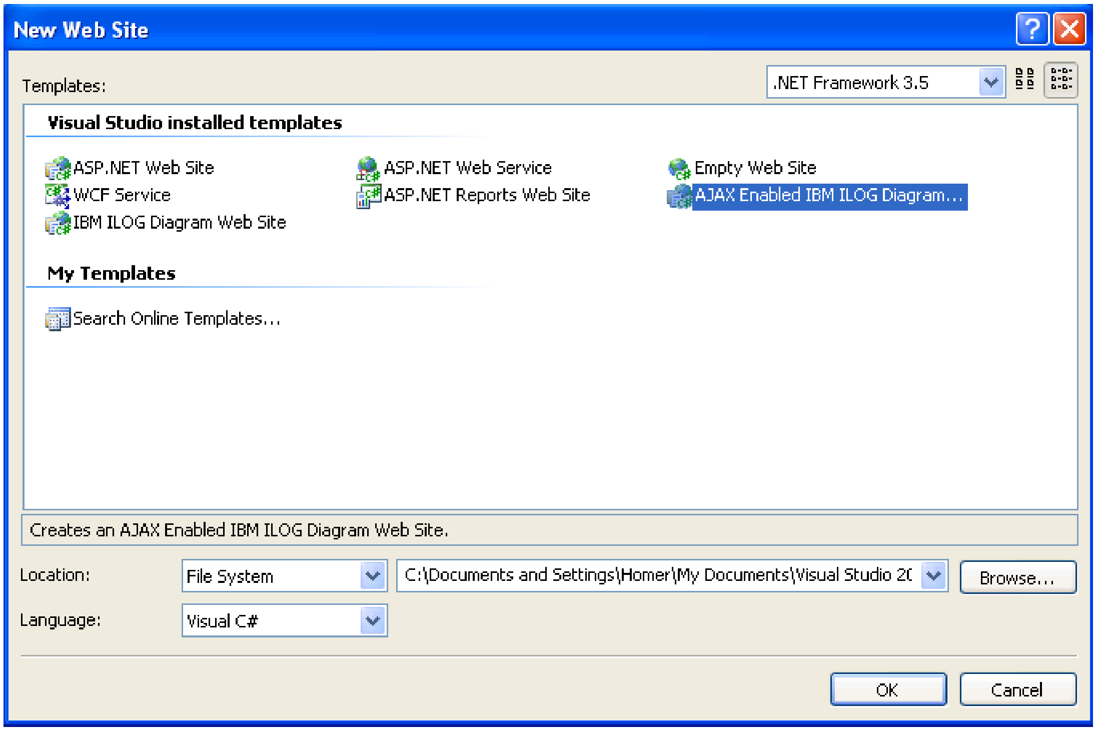
Creating your First IBM ILOG Diagram for .NET AJAX Web Site

This tutorial walks you through the creation of your first IBM® ILOG® Diagram for .NET AJAX Web site.

Through this tutorial you will learn the basics of the IBM ILOG Diagram for .NET API for creating a new Web site supporting rich client-side interactions.

Note: *To build AJAX enabled applications, IBM ILOG Diagram for .NET 2.0 requires by default the .NET Framework 3.5 SDK. In case you want to target a .NET Framework 2.0 Web application and ASP.NET AJAX 1.0, IBM ILOG Diagram for .NET 2.0 provides the* ***ILOG.Diagrammer.Web.Ajax10.dll*** *assembly which has a dependency with ASP.NET AJAX 1.0. Note that in this case, the IBM ILOG Diagram for .NET 2.0 Project Templates cannot be used and the web controls must be added manually to the Visual Studio Toolbox.*

1. Launch Visual Studio.
2. Select File>New Web Site from the menu bar. The New Web Site window opens.
3. Select AJAX Enabled IBM ILOG Diagram for .NET Web Site in Visual Studio installed templates and Visual C# in the Language combo box.



Creates an AJAX Enabled IBM ILOG Diagram Web Site.

Location: File System — C:\Documents and Settings\Homer\My Documents\Visual Studio 2(

Language: Visual C#

***Note:** In addition to the Web site templates, IBM ILOG Diagram for .NET 2.0 provides templates for creating Visual Studio 2008 Web Application projects. Select File>New Project, select the language and the Web project type, then the corresponding IBM ILOG Diagram template (IBM ILOG Diagram Web Application or IBM ILOG Diagram Ajax Web Application).*

This project template creates a fully functional AJAX-enabled IBM ILOG Diagram for .NET Web site that contains the following files:

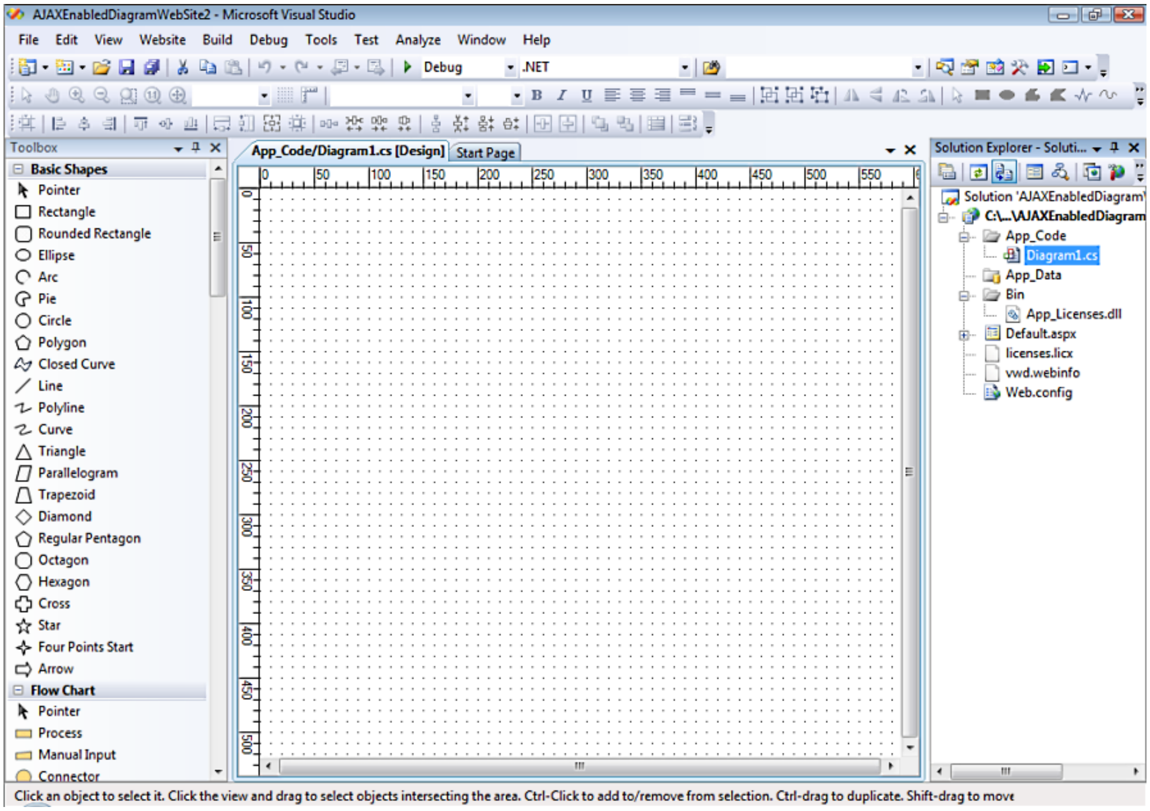
◆ `Default.aspx` that defines the main page of the Web site. This page contains an AjaxDiagramView object that is the IBM ILOG Diagram for .NET WebForm control to display a diagram. The diagram view is already configured to display the diagram that you will create.

◆ `App_Code\Diagram1.cs` that defines the diagram to display in the **AjaxDiagramView** control.

The `Diagram1.cs` file defines a new class named Diagram1 that is a subclass of the Group class of IBM ILOG Diagram for .NET. A **Group** object is a graphic object that simply displays its children.

4. Edit the content of your diagram. Open `Diagram1.cs` in Design mode. You can either select View>Designer from the menu bar or right-click `Diagram1.cs` in the Solution Explorer and select View Designer.

   Now you are able to define the content of your diagram directly within Visual Studio by dropping the graphic objects to the Diagram Designer.



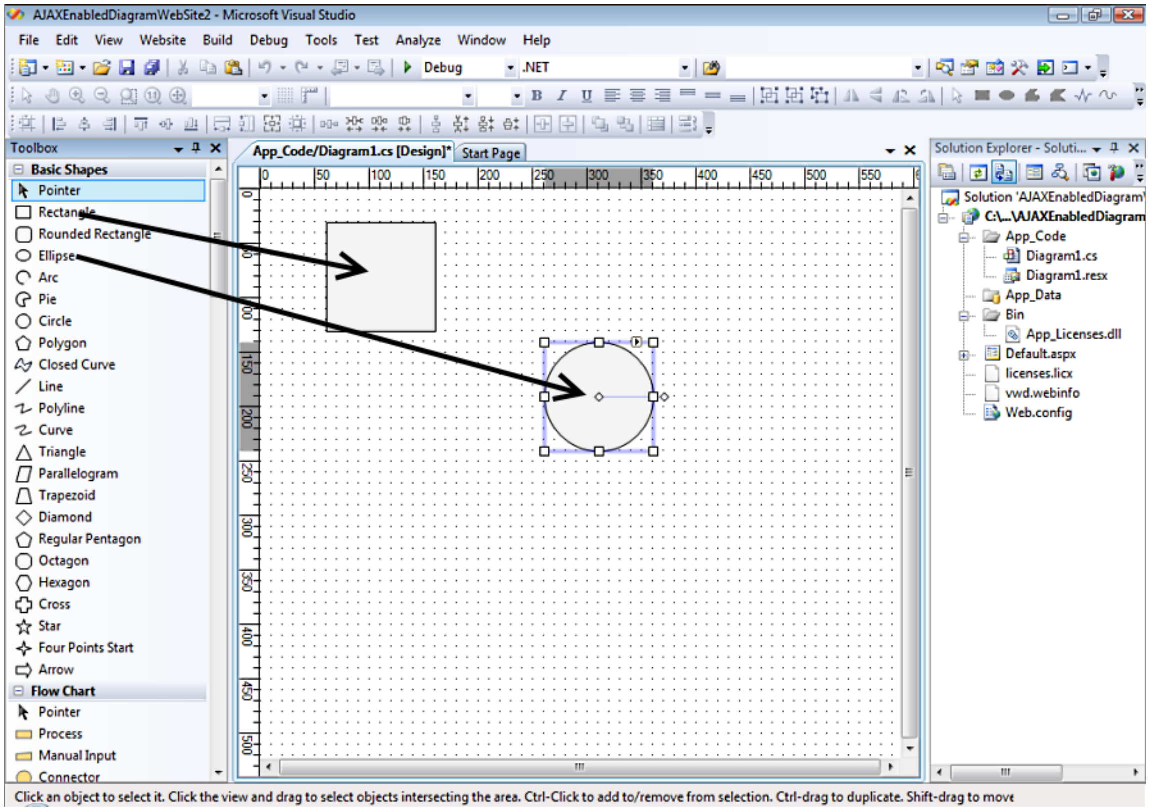
Click an object to select it. Click the view and drag to select objects intersecting the area. Ctrl-Click to add/remove from selection. Ctrl-drag to duplicate. Shift-drag to move

See Also *Creating your First IBM ILOG Diagram for .NET Windows Forms Application*

Building Diagrams and User Symbols Inside Visual Studio

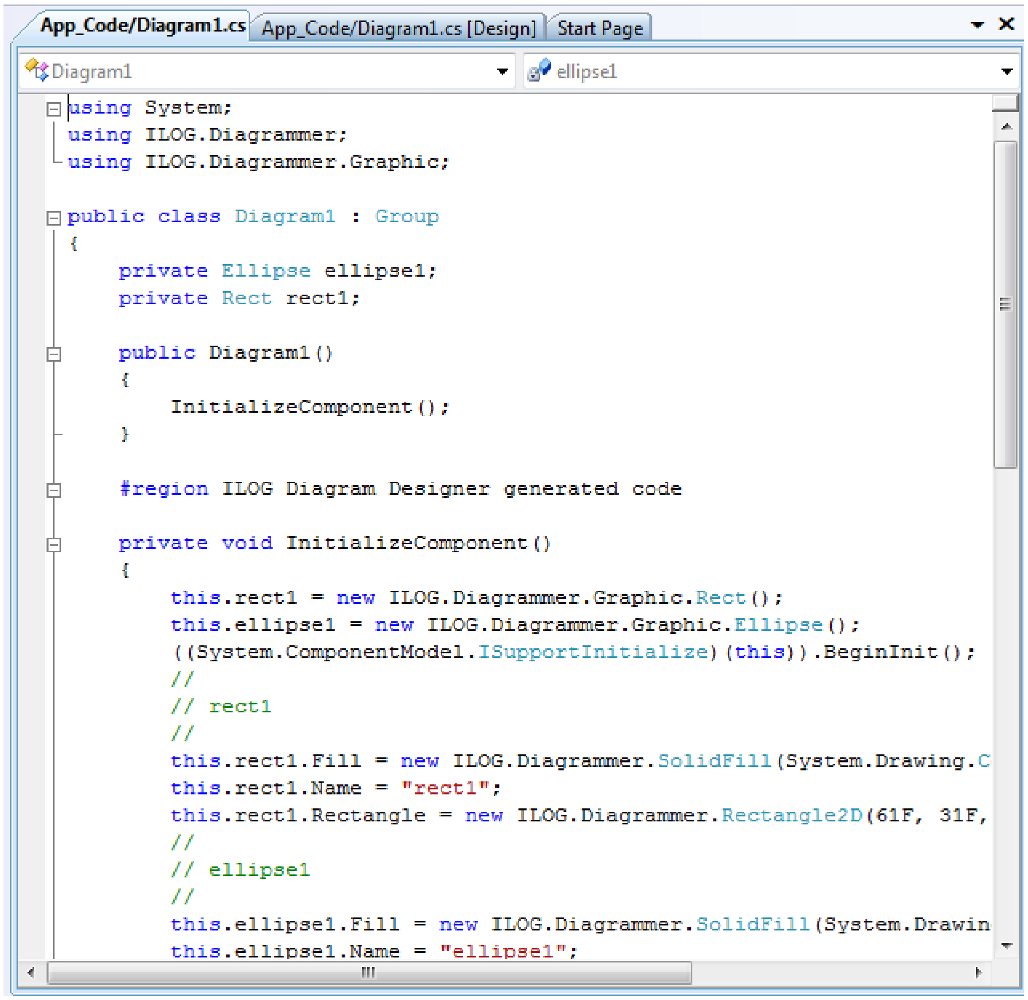
5. From the Toolbox, drag the Rectangle graphic object to the Designer.

6. Repeat the same operation with the Ellipse graphic object.



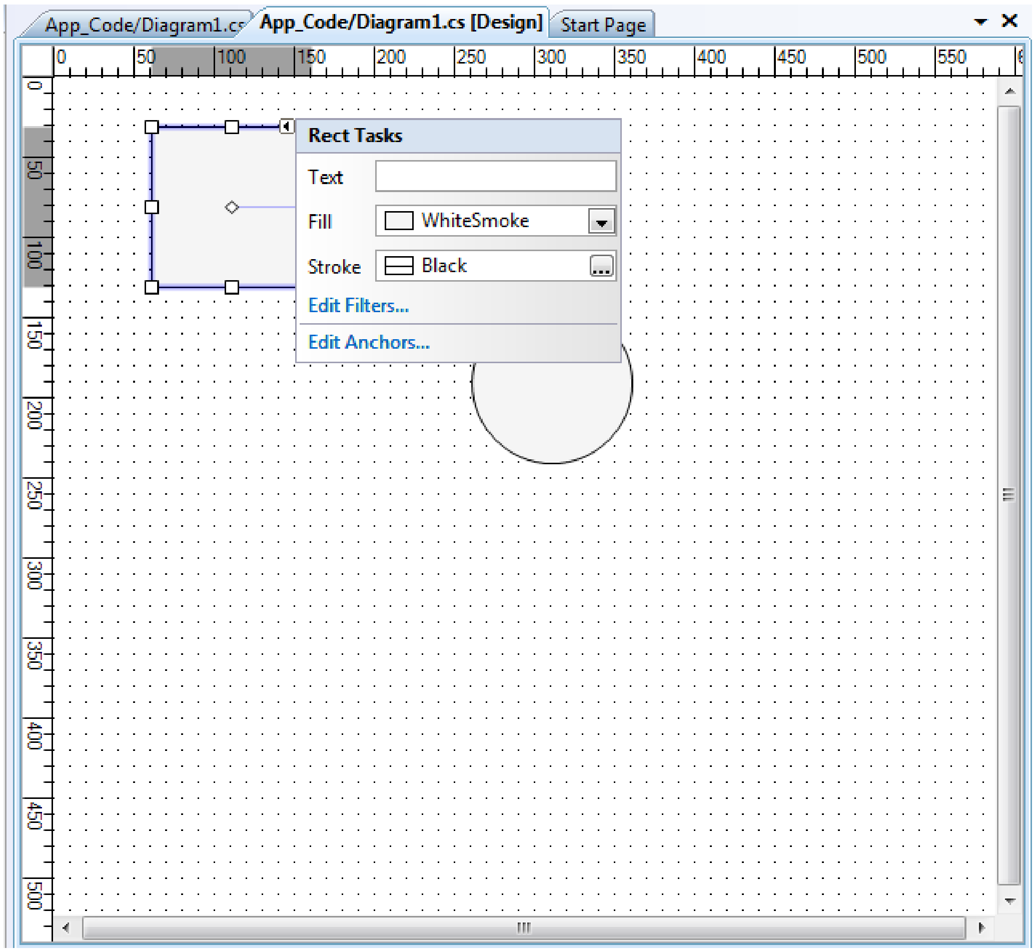
Click an object to select it. Click the view and drag to select objects intersecting the area. Ctrl-Click to add/remove from selection. Ctrl-drag to duplicate. Shift-drag to move

As you drag graphic objects to your diagram, the Diagram Designer generates the code that corresponds to your diagram in the `Diagram1.cs` file.



```
using System;
using ILOG.Diagrammer;
using ILOG.Diagrammer.Graphic;

public class Diagram1 : Group
{
    private Ellipse ellipse1;
    private Rect rect1;

    public Diagram1()
    {
        InitializeComponent();
    }

    #region ILOG Diagram Designer generated code

    private void InitializeComponent()
    {
        this.rect1 = new ILOG.Diagrammer.Graphic.Rect();
        this.ellipse1 = new ILOG.Diagrammer.Graphic.Ellipse();
        ((System.ComponentModel.ISupportInitialize)(this)).BeginInit();
        //
        // rect1
        //
        this.rect1.Fill = new ILOG.Diagrammer.SolidFill(System.Drawing.C
        this.rect1.Name = "rect1";
        this.rect1.Rectangle = new ILOG.Diagrammer.Rectangle2D(61F, 31F,
        //
        // ellipse1
        //
        this.ellipse1.Fill = new ILOG.Diagrammer.SolidFill(System.Drawin
        this.ellipse1.Name = "ellipse1";
```

7. Click the smart tag icon of the selected graphic object to open the smart tag panel which lists the common design-time actions specific to this object.



Text

Fill: WhiteSmoke

Stroke: Black

8. Change the Fill property to a color of your choice.

9. Close the smart tag panel by clicking anywhere in the Designer.

10. Select the Ellipse graphic object and repeat the same operation to change the Fill property.

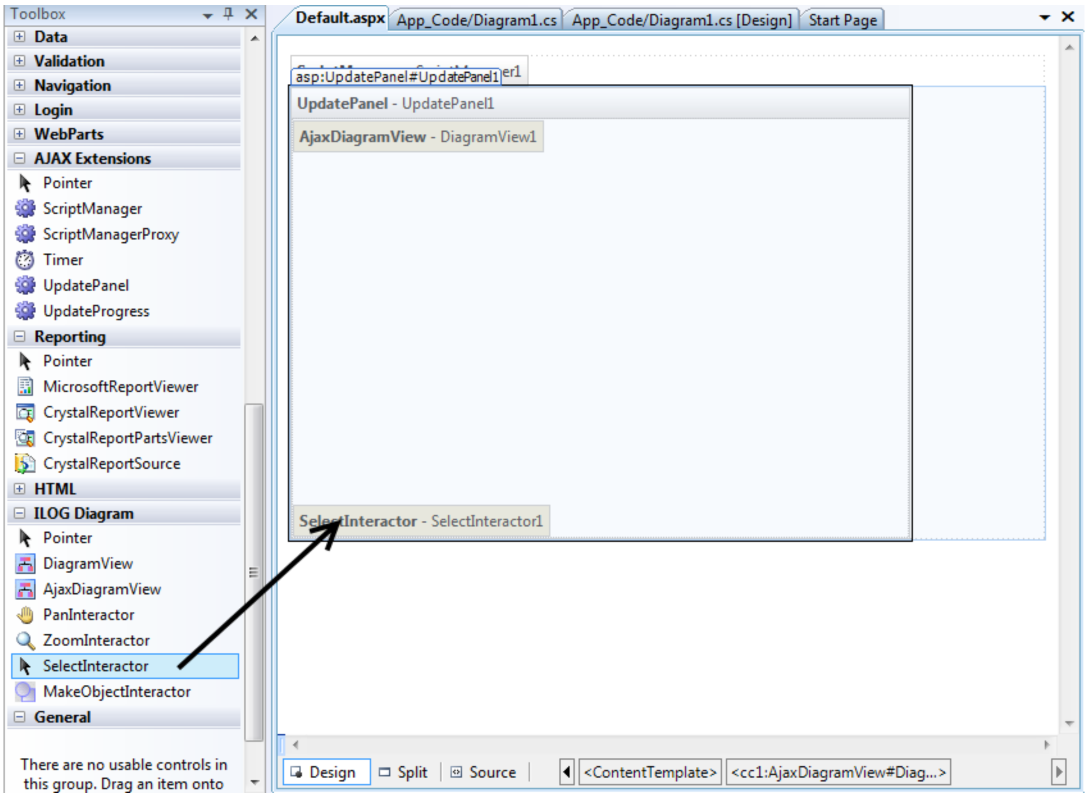
One of the benefits of the **AjaxDiagramView** control is that it supports rich client-side user interaction by means of ViewInteractor objects. A ViewInteractor is a Web control that adds client-side behavior to an **AjaxDiagramView** control.

In this example, you are going to add an interactor that enables the user to select graphic objects within its browser window, to move them or to request contextual information asynchronously (the latter is not covered in this tutorial).

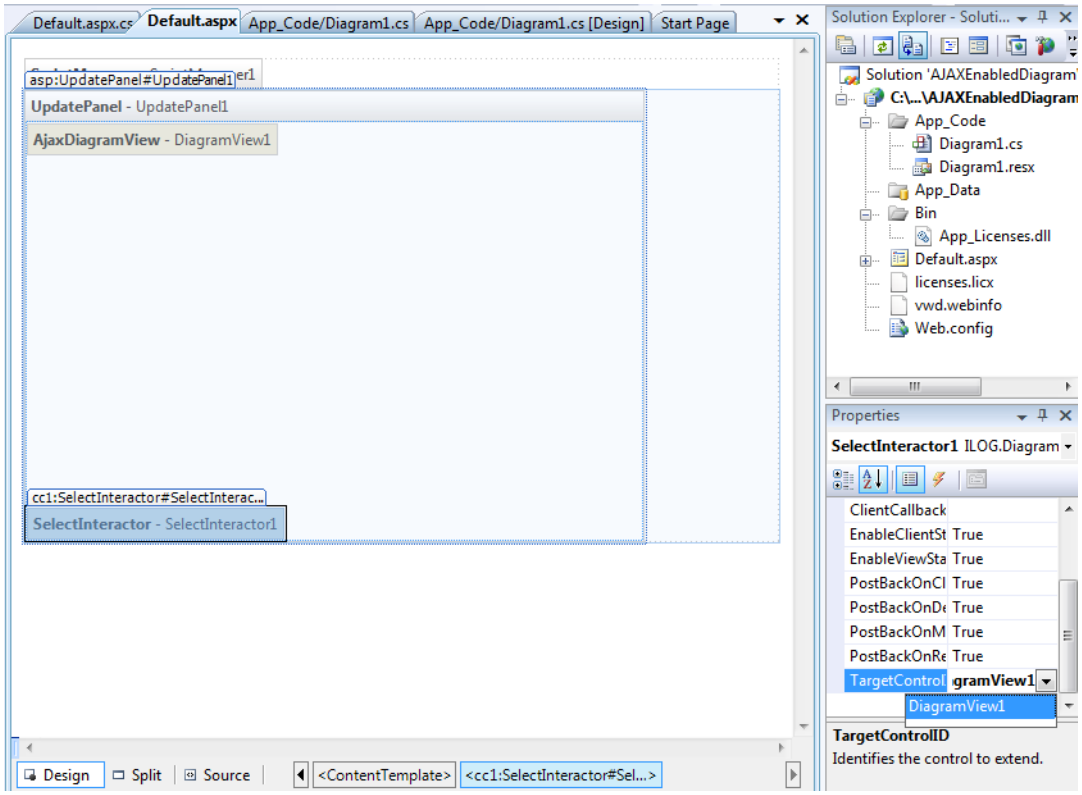
11. Open the `Default.aspx` page.

12. Switch to the Design view.

13. From the Toolbox, select the SelectInteractor control and drag it to the Designer in the UpdatePanel container.



14. Select the interactor control and in the Properties Window set its TargetControlID property to DiagramView1.


Default.aspx.cs
Default.aspx
App_Code/Diagram1.cs
App_Code/Diagram1.cs [Design]
Start Page
asp:UpdatePanel#UpdatePanel1
UpdatePanel - UpdatePanel1
AjaxDiagramView - DiagramView1
cc1:SelectInteractor#SelectInterac...
SelectInteractor - SelectInteractor1
Design
Split
Source
<ContentTemplate>
<cc1:SelectInteractor#Sel...>
Solution Explorer - Soluti...
Solution 'AJAXEnabledDiagram'
C:\...\AJAXEnabledDiagram
App_Code
Diagram1.cs
Diagram1.resx
App_Data
Bin
App_Licenses.dll
Default.aspx
licenses.licx
vwd.webinfo
Web.config
Properties
SelectInteractor1 ILOG.Diagram

ClientCallback	
EnableClientSt	True
EnableViewSta	True
PostBackOnCl	True
PostBackOnDe	True
PostBackOnM	True
PostBackOnRe	True
TargetControl	gramView1

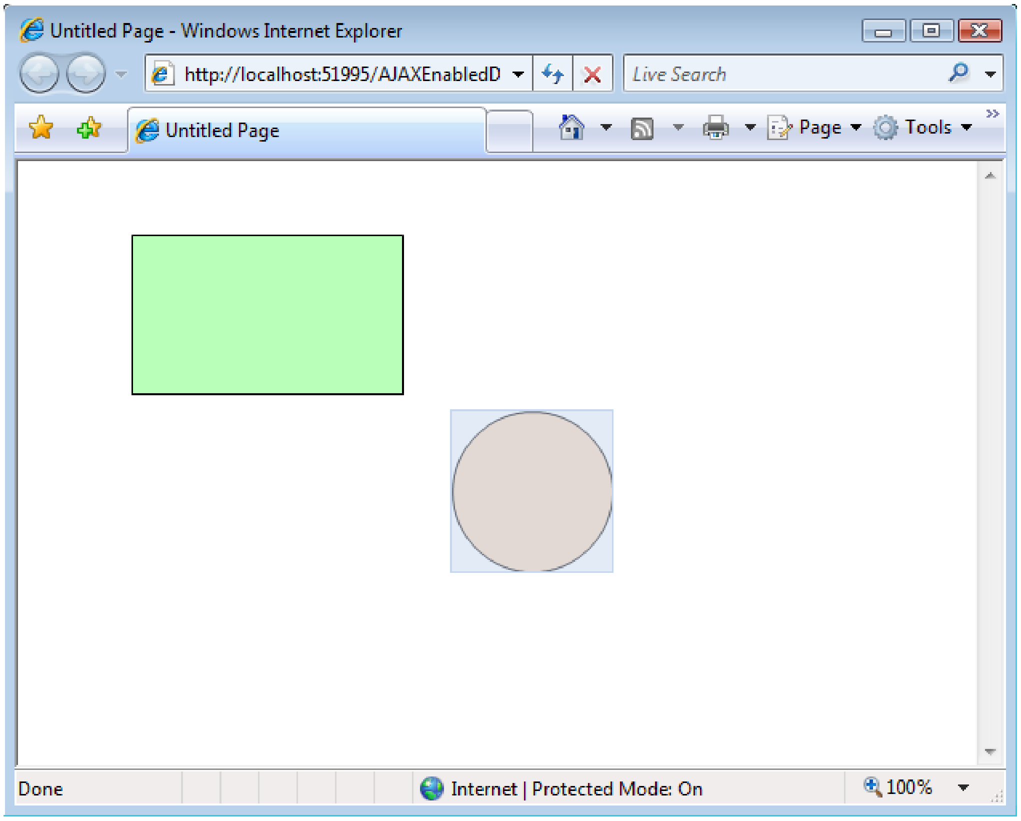
DiagramView1

TargetControlID
Identifies the control to extend.

Now you can browse the Web site.

15. In the Solution Explorer, right-click the project and select View in Browser.

You will see that when a graphic object is clicked, it becomes highlighted with a transparent blue rectangle which renders its selected state. You can also move a graphic object to another location by dragging it within the diagram view to the browser window (you can cancel the current interaction pressing the ESC key). Finally, to delete a graphic object select it and press the Delete key.


Untitled Page - Windows Internet Explorer
http://localhost:51995/AJAXEnabledD
Live Search
Untitled Page
Page
Tools
Done
Internet | Protected Mode: On
100%

Now that you have created a basic IBM ILOG Diagram for .NET Web site application, you can continue by populating the graphic container with your real diagram or add advanced interactions like graphic object creation, link connection, or display contextual information fetched asynchronously.

Index