

Silverlight 4.0: What's new

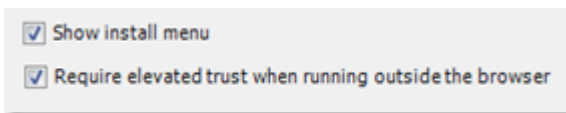
Corrado Cavalli –MVP - corrado.cavalli@manageddesigns.it - blogs.ugidotnet.org/corrado

Just a few months after Silverlight version 3.0, the team is ready with the first public beta of version 4.0 which, as you will read, it presents a wealth of interesting news.

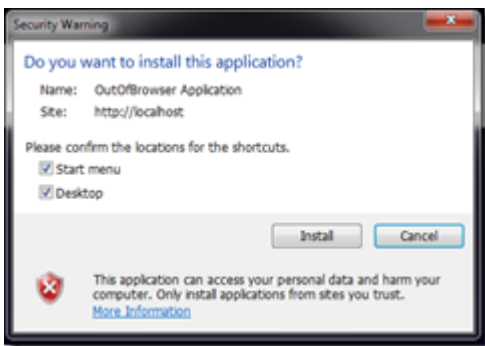
Here is a quick overview of what you'll get.

Permission elevation in Out-Of-Browser (OOB) applications

Out-of-browser property windows now list two new interesting entries:



- **Show Install Menu:** Allows you to remove *Install application* option menu when you right-click an OOB enabled application.
- **Require elevated trust when running outside the browser:** Requests elevated privileges when an application is first run locally, user is prompted at installation time to authorize elevation in a ClickOnce like style.



OOB authorized operations are:

- Safe communication with cross-domain services.
- An elevated OOB application can automatically start in full screen mode without user intervention.

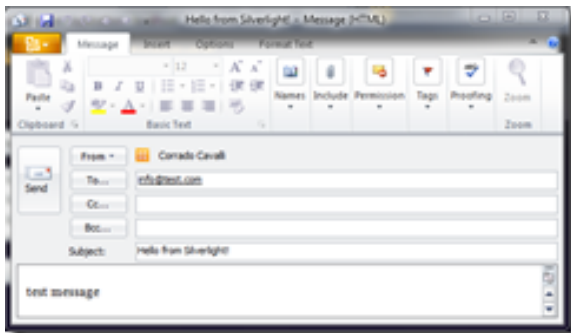
- In full screen mode ESC key doesn't automatically change view to windowed mode, you must explicitly invoke: `Application.Current.Host.Content.IsFullScreen = false`
- It's possible to determinate if an OOB application is running with elevated permissions checking the value of `Application.Current.HasElevatedPermissions` property.
- Direct access to following folders: **MyDocument**, **MyPictures**, **MyVideos** e **MyMusic**
- COM interoperability is also supported, e.g. a Silverlight4 application can use local Outlook to send an email using this few lines of code:

```

if (ComAutomationFactory.IsAvailable)
{
    dynamic outlook = ComAutomationFactory.CreateObject("Outlook.Application");
    dynamic mailItem = outlook.CreateItem(0);
    mailItem.To = "info@test.com";
    mailItem.Subject = "Hello from Silverlight!";
    mailItem.HTMLBody = "<P>test message</P>";
    mailItem.Display();
}

```

Here's the final output:



As you can see Silverlight 4 fully supports late binding via `dynamic` (C#) and `Object` (VB) keywords.

Right to Left support

FrameworkElement now exposes a `FlowDirection` property so that right to left languages like Arabic, Hebrew etc can be properly handled inside a Silverlight app.

Webbrowser control per le applicazioni OOB

It is now possible to render HTML content inside an out-of-browser application by mean of `WebBrowser` control and related `Source` property and/or `NavigateTo` method.

Control can only render contents coming from application's site or origin and it visible only in OOB mode.

```
<WebBrowser Margin="0,3,0,0" Source="http://localhost:1832/MyPage.htm" Grid.Row="1"/>
```

Implicit Styles e ViewBox

Like in WPF is it now possible to define implicit styles without resorting to Toolkit's ImplicitStyleManager and also scale client area contents using the **ViewBox** control:

```
<UserControl x:Class="ImplicitStyleAndViewBox.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  d:DesignHeight="300"
  d:DesignWidth="400">

  <UserControl.Resources>
    <Style TargetType="TextBox">
      <Setter Property="Background" Value="Yellow" />
      <Setter Property="FontSize" Value="14" />
    </Style>
  </UserControl.Resources>

  <Viewbox>
    <Grid x:Name="LayoutRoot" Background="White">
      <TextBox Height="23" HorizontalAlignment="Left" Text="Hello" />
    </Grid>
  </Viewbox>
</UserControl>
```

Below you can see xaml fragment output

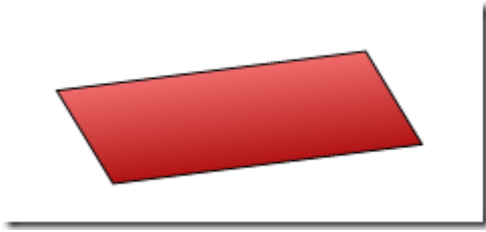


Commanding support

CommandBase class now exposes a **Command** and **CommandParameter** properties adding the same commanding system available since v1.0 on Window Presentation Foundation to Silverlight, more infos about commanding system are available here: <http://msdn.microsoft.com/en-us/library/ms752308.aspx>.

CompositeTransform

Let's suppose we're asked to draw a rectangle as the one below:



In addition to classic approach:

```
<Rectangle.RenderTransform>
  <TransformGroup>
    <ScaleTransform ScaleX="2" ScaleY="2" />
    <SkewTransform AngleX="12" AngleY="12" />
    <RotateTransform Angle="-20" />
  </TransformGroup>
</Rectangle.RenderTransform>
```

We can now use a **CompositeTransform** object avoiding multiple object transformations and with a more compact xaml.

```
<Grid x:Name="LayoutRoot" Background="White">
  <Rectangle Height="54" Margin="114,83,133,0" Stroke="Black" VerticalAlignment="Top"
  RenderTransformOrigin="0.5,0.5">
    <Rectangle.RenderTransform>
      <CompositeTransform ScaleX="2"
                          ScaleY="2"
                          Rotation="-20"
                          SkewX="12"
                          SkewY="12" />
    </Rectangle.RenderTransform>
    <Rectangle.Fill>
      <LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox"
  StartPoint="0.5,0">
        <GradientStop Color="#FFB11313" Offset="1"/>
        <GradientStop Color="#FFF16B6B" Offset="0.013"/>
      </LinearGradientBrush>
    </Rectangle.Fill>
  </Rectangle>
</Grid>
```

Note: *Blend4 doesn't use CompositeTranform yet.*

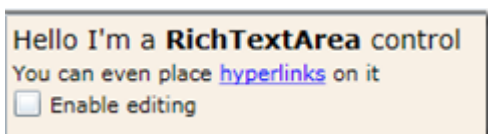
RichTextArea

WPF's RichTextBox comes, with a different name, to Silverlight platform and allows you to display and edit formatted text.

Here's a small fragment of code demonstrating how to use the new RichTextArea control:

```
<RichTextArea Background="AntiqueWhite" IsReadOnly="True">
  <Paragraph FontSize="14"> Hello I'm a <Bold>RichTextArea</Bold> control</Paragraph>
  <Paragraph>You can even place <Hyperlink
NavigateUri="http://3.ly/K0G">hyperlinks</Hyperlink> on it</Paragraph>
  <Paragraph>
    <InlineUIContainer>
      <CheckBox Content="Enable editing" />
    </InlineUIContainer>
  </Paragraph>
</RichTextArea>
```

What we finally get is this:



Due to a bug in Visual Studio 2010 Beta2, designer white spaces inside paragraph aren't properly handled at design time.

PrintDocument

Silverlight 4 finally adds one of the most requested features: Printing!

If you're a Windows forms developer you'll probably recognize the class involved in printing mechanism: PrintDocument.

Let's say we need to print what's rendered by this small xaml fragment:

```
<Grid x:Name="LayoutRoot">
  <Button Content="Print" Height="23" HorizontalAlignment="Left" Margin="33,170,0,0"
    VerticalAlignment="Top" Width="75" Click="button1_Click" />
  <Image Height="142" HorizontalAlignment="Left" Margin="33,12,0,0" Stretch="Fill"
VerticalAlignment="Top" Width="201"
    Source="/Printing;component/Clown_Fish.jpg" />
  <TextBlock Height="36" HorizontalAlignment="Left" Margin="276,72,0,0" Text="PrintDocument!"
VerticalAlignment="Top"
    Width="164"
    FontSize="20" Foreground="#FF7ECB27" />
  <Grid.Background>
    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
      <GradientStop Color="Black" Offset="0" />
      <GradientStop Color="White" Offset="1" />
    </LinearGradientBrush>
  </Grid.Background>
</Grid>
```

All we need to do is use this code, that's probably reminding you the old WinForms era: ☺

```

public partial class MainPage : UserControl
{
    PrintDocument pd = new PrintDocument();
    public MainPage()
    {
        InitializeComponent();
        pd.StartPrint += OnStartPrinting;
        pd.EndPrint += new EventHandler<EndPrintEventArgs>(OnEndPrinting);
        pd.PrintPage += new EventHandler<PrintPageEventArgs>(OnPrintPage);
    }

    void OnPrintPage(object sender, PrintPageEventArgs e)
    {
        e.PageVisual = LayoutRoot;
        e.HasMorePages = false;
    }

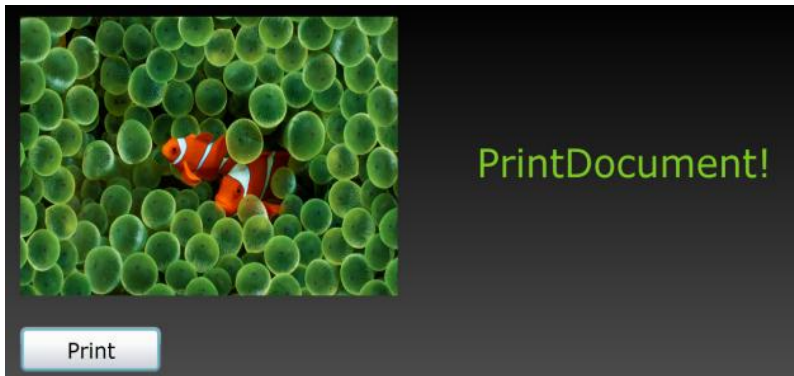
    void OnEndPrinting(object sender, EndPrintEventArgs e)
    {
        Debug.WriteLine("Print Completed");
    }

    void OnStartPrinting(object sender, StartPrintEventArgs e)
    {
        Debug.WriteLine("Print started");
    }

    private void button1_Click(object sender, RoutedEventArgs e)
    {
        pd.Print();
    }
}

```

Here's what gets printed:



WebCam e Microphone

Usually interaction with webcam and microphone begins with retrieval of available sources, to get available webcams we can use following snippet:

```

public MainPage()
{

```

```

InitializeComponent();
//lbWebcams is a listbox...
lbWebcams.DisplayMemberPath = "FriendlyName";
lbWebcams.ItemsSource = CaptureDeviceConfiguration.GetAvailableVideoCaptureDevices();
}

```

Or just use default one by simply using this line of code:

```
VideoCaptureDevice device = CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();
```

Following code renders the webcam live contents inside a border element:

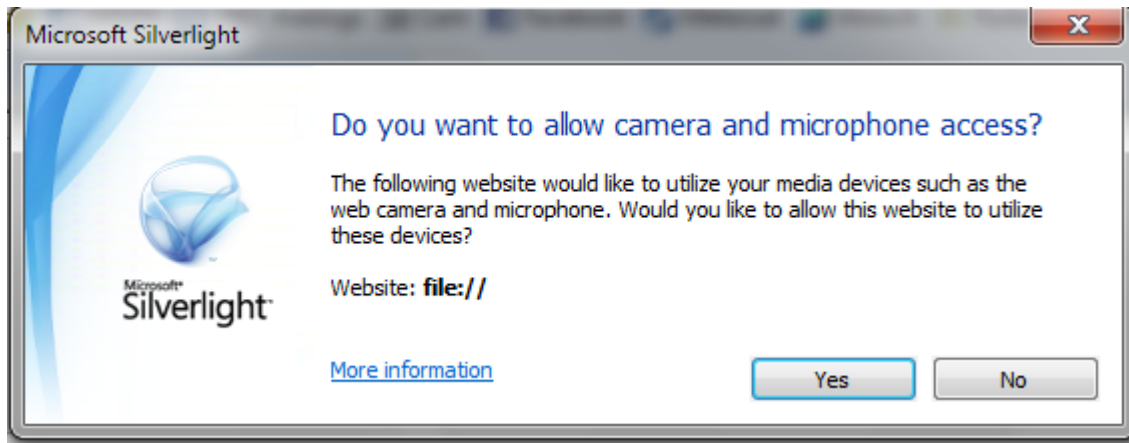
```

CaptureSource cs = new CaptureSource();

private void button1_Click(object sender, RoutedEventArgs e)
{
    VideoCaptureDevice device = CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();
    if (device != null)
    {
        bool authorized=CaptureDeviceConfiguration.RequestDeviceAccess();
        if (authorized)
        {
            cs.VideoCaptureDevice = device;
            VideoBrush vb = new VideoBrush();
            vb.SetSource(cs);
            border1.Background = vb; //using border as capture area
            cs.Start();
        }
    }
}

```

RequestDeviceAccess method requires user to confirm access to webcam or microphone by Silverlight 4 application.



Grabbing webcam's current frame it's just a matter of invoking **AsyncCaptureImage** method:

```

cs.AsyncCaptureImage((snapImage) =>
{
    image1.Source = snapImage; //snapImage is a writeablebitmap
});

```

Microphone interaction is quite similar to what shown before; **GetAvailableAudioCaptureDevices** return the list of available audio sources:

```
Collection<AudioCaptureDevice> inputLines =  
CaptureDeviceConfiguration.GetAvailableAudioCaptureDevices();
```

As before, if you wish to use predefined input just use following line:

```
AudioCaptureDevice device = CaptureDeviceConfiguration.GetDefaultAudioCaptureDevice();
```

Getting audio samples requires developer to create a class that inherits and implements abstract methods of **AudioSink** class:

```
public class MyAudioSync:AudioSink  
{  
    protected override void OnCaptureStarted()  
    {  
    }  
  
    protected override void OnCaptureStopped()  
    {  
    }  
  
    protected override void OnFormatChange(AudioFormat audioFormat)  
    {  
    }  
  
    protected override void OnSamples(long sampleTime, long sampleDuration, byte[] sampleData)  
    {  
        //Process samples here...  
    }  
}
```

What's missing now is to connect our custom AudioSink with audio capture device:

```
CaptureSource cs = new CaptureSource();  
  
AudioCaptureDevice device = CaptureDeviceConfiguration.GetDefaultAudioCaptureDevice();  
if (device != null)  
{  
    bool authorized =CaptureDeviceConfiguration.RequestDeviceAccess();  
    if (authorized)  
    {  
        cs.AudioCaptureDevice = device;  
        MyAudioSync mas = new MyAudioSync();  
        mas.CaptureSource = cs;  
        cs.Start();  
    }  
}
```

Don't forget to invoke **Stop** method when you no longer need to interact with webcam or microphone.

DataBinding

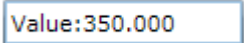
Binding is no longer limited to elements deriving from FrameworkElement, you can now virtually bind to any element.

This code, failing in Silverlight 3.0 due to PlaneProjection not deriving from FrameworkElement, runs fine in v4.0

```
<Slider Maximum="180" Name="slider1" Width="224" />
    <Image Source="desert.jpg" RenderTransformOrigin="0.5,0.5">
        <Image.Projection>
            <PlaneProjection RotationX="{Binding ElementName=slider1, Path=Value}" />
        </Image.Projection>
    </Image>
</Slider>
```

Binding class now supports **StringFormat**, **IDataErrorInfo** e **INotifyDataErrorInfo**

```
<TextBox Text="{Binding ElementName=window1, Path=Width, StringFormat=Value:0.000}" />
```



Value:350.000

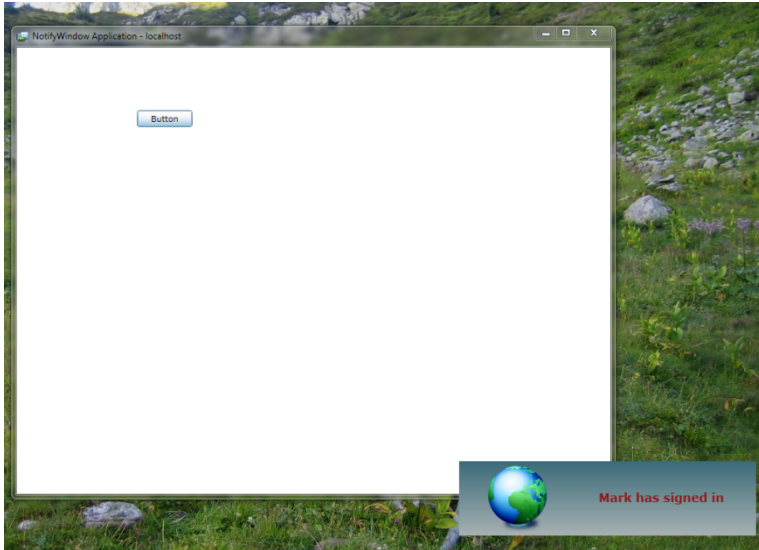
NotificationWindow

Out-of-browser application using can display a notification window in lower right corner (a.k.a 'Toast' Window) using **NotificationWindows** class.

Here's a small sample that displays a notification for 4 seconds:

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    if (App.Current.IsRunningOutOfBrowser)
    {
        if (win != null && win.Visible)
            win.Close();
        else
            win = new NotificationWindow();
        //Toast is notification window content
        Toast toast = new Toast();
        win.Width = toast.Width; //max width is 400
        win.Height = toast.Height; //ma height is 100
        win.Content = toast;
        win.Show(4000);
    }
}
```

Toast is a UserControl used as notification window contents whose maximum dimension cannot be greater than 400x100 pixels. Final output is:



HTML Brush

If your application is running out-of-browser, you can use WebBrowser control content as brush for other elements thanks to the new **HTMLBrush** as following snippet demonstrates:

```
<Grid x:Name="LayoutRoot">
  <Grid.RowDefinitions>
    <RowDefinition Height="23"/>
    <RowDefinition Height="0.5*"/>
    <RowDefinition Height="0.5*"/>
  </Grid.RowDefinitions>
  <Grid.Background>
    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
      <GradientStop Color="Black" Offset="0"/>
      <GradientStop Color="White" Offset="1"/>
    </LinearGradientBrush>
  </Grid.Background>
  <WebBrowser x:Name="wb" Margin="0,3,0,0" Source="http://localhost:1832/MyPage.htm"
Grid.Row="1" Grid.RowSpan="1"/>
  <Button Content="Navigate" HorizontalAlignment="Left" Height="23" VerticalAlignment="Bottom"
Width="73"
      Click="Button_Click" />
  <Rectangle x:Name="Rectangle1" Grid.Row="2" Stroke="Black">
    <Rectangle.Fill>
      <HtmlBrush x:Name="htmlBrush" SourceName="wb" />
    </Rectangle.Fill>
  </Rectangle>
</Grid>
```

To synchronize **HTMLBrush** with browser content when this changes, call the **Redraw** method.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    htmlBrush.Redraw();
}
```

Navigation Extensibility

Silverlight 3 navigation system can be extended to support special scenarios like redirection or on demand download of requested pages.

Extensibility is achieved by associating a custom **INavigationContentLoader** (in System.Windows.Navigation dll) implementation to Frame's ContentLoader property.

By default navigation is handled by a **PageResourceContentLoader** instance, that's actually the only implementation Silverlight natively offer.

Here's **INavigationContentLoader** definition:

```
public interface INavigationContentLoader
{
    IAsyncResult BeginLoad(Uri targetUri, Uri currentUri, AsyncCallback userCallback, object
asyncState);
    void CancelLoad(IAsyncResult asyncResult);
    bool CanLoad(Uri targetUri, Uri currentUri);
    LoadResult EndLoad(IAsyncResult asyncResult);
}
```

Drag & Drop support

Handling files dragged from Windows Explorer inside a Silverlight 4.0 application is just a two quick easy steps: set element's **AllowDrop** =True then handle **Drop** event as shown below:

```
<Image AllowDrop="True" />
```

```
private void OnFileDrop(object sender, System.Windows.DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.FileDrop))
    {
        FileInfo[] droppedFiles=(FileInfo[])e.Data.GetData(DataFormats.FileDrop);
        FileInfo fileInfo=droppedFiles[0];
        System.IO.Stream stream = fileInfo.OpenRead();

        string fileName = string.Format("{0}{1}",
                                        Guid.NewGuid().ToString(),
                                        fileInfo.Extension);
        UploadFile(fileName, stream);
    }
}
```

Clipboard access

While limited to Unicode text for now, Silverlight 4 offers system Clipboard access.

If application is not running in elevated mode, at first ClipBoard access for any Silverlight session, user is

prompted to confirm its use.

Using Clipboard is just a matter of using following methods:

```
//Copy  
Clipboard.SetText(textBox1.Text);  
  
//Paste  
textBox1.Text = Clipboard.GetText();
```

Right-Click Mouse events

Silverlight 4.0 adds UIElements with two new interesting events: **MouseRightButtonDown** and **MouseRightButtonUp**, in previous version these elements were handled by plug-in to show Silverlight's configuration menu.

Current behavior remains untouched but you can easily change it by handling MouseRightButtonDown event this way:

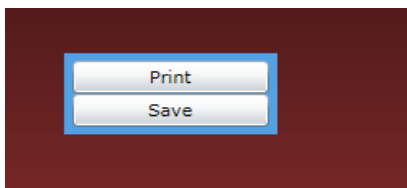
```
<Grid x:Name="LayoutRoot" MouseRightButtonDown="OnMouseRightButtonDown" />

private void OnMouseRightButtonDown(object sender, MouseButtonEventArgs e)
{
    e.Handled = true;
}
```

While still missing a ContextMenu, limitation can be replaced by using a Popup element:

```
<Grid x:Name="LayoutRoot" MouseRightButtonDown="OnMouseRightButtonDown">
    <Grid.Background>
        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="Black" Offset="0" />
            <GradientStop Color="#FFBA3C3C" Offset="1" />
        </LinearGradientBrush>
    </Grid.Background>
    <Popup x:Name="popup">
        <Border Background="#FF52A1E5" Padding="5" >
            <StackPanel Width="120">
                <Button Content="Print" />
                <Button Content="Save" />
            </StackPanel>
        </Border>
    </Popup>
</Grid>

private void OnMouseRightButtonDown(object sender, MouseButtonEventArgs e)
{
    popup.HorizontalOffset = e.GetPosition(null).X + 2;
    popup.VerticalOffset = e.GetPosition(null).Y + 2;
    popup.IsOpen = true;
    e.Handled = true;
}
```



Goodbye .NET RIA Services, Hello WCF RIA Services!

Technology formerly known as **.NET RIA Services** has been renamed to **WCF RIA Services**, the new release contains refactor aspects that probably will broke your app based on July 09 drop, but nothing that can't be easily fixed.

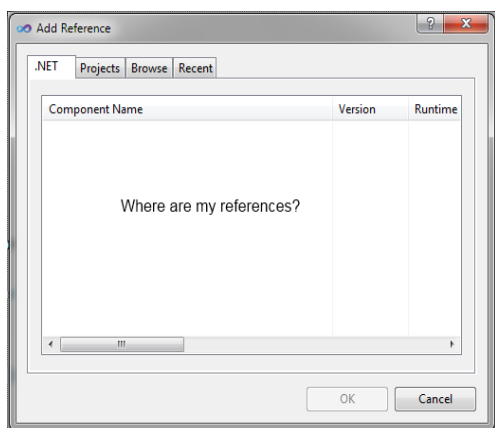
Probably the most important breaking changes regards **[Custom]** and **[ServiceOperation]** attributes that have been changed as depicted in following table.

<i>.Net RIA Services</i>	<i>WCF RIA Services</i>
Custom	Update (bool useCustomMethod)
ServiceOperation	Invoke

Note: Actual release targets Visual Studio 2010 beta2 and Fx 4.0 only.

Known Issues

Trying to add a new Silverlight reference in a Silverlight 4.0 application results in an empty .NET assembly list inside **Add Reference** window, for the moment the only option is to browse the assembly and manually add it.



The last word

I've just mentioned some from the long list of news added to Silverlight 4.0, I encourage you to install the bits and start exploring the **silverlight.chm** guide for a complete and more detailed analysis.