

Silverlight 4.0: What's new

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

A distanza di pochi mesi dal rilascio della versione 3.0 il team di Silverlight è già pronto per regalarci la prima beta pubblica della versione 4.0 la quale, come avrete modo di leggere, si presenta ricchissima di interessanti novità.

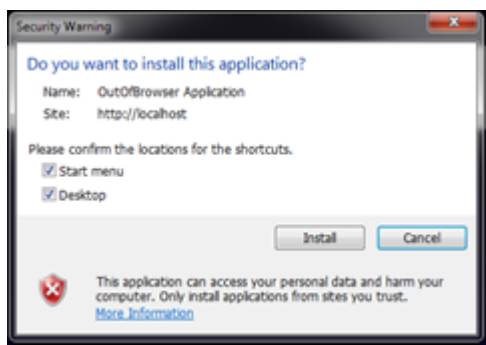
Ecco una breve carrellata delle principali novità.

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

La finestra di impostazione delle proprietà dell'applicazione out-of-browser si arricchisce di due interessanti opzioni:



- **Show Install Menu:** Permette di rimuovere l'opzione *Install application* quando si fa right-click di un applicazione OOB enabled
- **Require elevated trust when running outside the browser:** Richiede l'elevazione dei privilegi consentiti all'applicazione quando installata localmente, il tutto deve essere autorizzato dall'utente:



Tra le operazioni consentite alle applicazioni OOB abbiamo:

- Comunicazione con servizi esterni al dominio da cui è stata generata l'applicazione senza necessità del relativo file *clientaccesspolicy.xml* lato server.

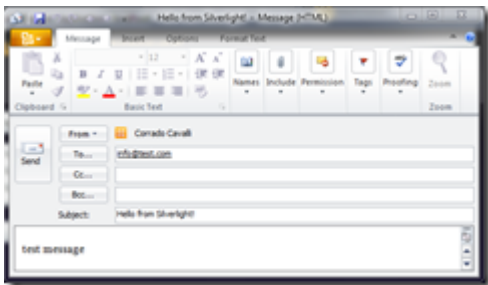
- Full screen mode senza che l'operazione venga iniziata da un operatore, ovvero una OOB application può partire anche in modalità full-screen.
- In full screen mode il tasto **ESC** non riporta l'applicazione nella modalità window, deve essere gestita in maniera esplicita usando `Application.Current.Host.Content.IsFullScreen = false`
- È possibile determinare se l'applicazione sta girando con permessi elevati usando `Application.Current.HasElevatedPermissions`
- Accesso diretto alle seguenti cartelle locali: **MyDocument**, **MyPictures**, **MyVideos** e **MyMusic**
- Supporto alla interoperabilità COM, ovvero un applicazione Silverlight4 può inviare una mail via Outlook in questo modo:

```

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();
}

```

Ottenendo come risultato l'apertura della finestra di Outlook



Notate come Silverlight4 supporta nativamente il late binding attraverso la keyword **dynamic** di C#, oppure **Object** di VB.

Supporto da parte dei controlli della direzione **Right to Left**

FrameworkElement ora espone una proprietà **FlowDirection** che consente alle applicazioni versione 4.0 di gestire correttamente linguaggi Right to Left come arabo, ebreo etc...

Controllo **WebBrowser** per le applicazioni OOB

Volendo visualizzare del contenuto HTML in un applicazione Out-Of-Browser è possibile utilizzare il nuovo controllo **WebBrowser** con la relativa proprietà **Source** o il metodo **NavigateTo**. Il controllo non è in grado di renderizzare contenuto proveniente da un dominio diverso dal quale è stata scaricata l'applicazione ed è visibile solo in modalità Out-Of-Browser:

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

Implicit Styles e ViewBox

Esattamente come in WPF è ora possibile definire stili impliciti senza ricorrere all'uso di `ImplicitStyleManager` e usare il controllo `ViewBox`:

```
<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>
```

ottenendo:

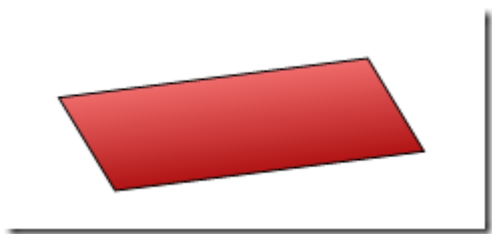


Commanding support

`CommandBase` espone le proprietà **Command** e **CommandParameter** che consentono di utilizzare lo stesso commanding system presente in Windows Presentation Foundation sin dalla versione 1.0, ulteriori info relative al commanding system sono disponibili a questo link: <http://msdn.microsoft.com/en-us/library/ms752308.aspx>

CompositeTransform

Supponendo di voler applicare delle trasformazioni ad un elemento come l'esempio che segue:



Oltre al classico approccio:

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

si può utilizzare il nuovo oggetto **CompositeTransform** evitando trasformazioni in sequenza e con dello xaml più contenuto:

```
<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>
```

Nota: *Blend4 non gestisce ancora CompositeTransform.*

RichTextArea

Il controllo RichTextBox di WPF arriva in Silverlight permettendo di visualizzare ed editare testo formattato.

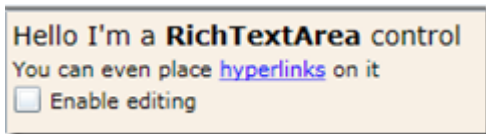
Ecco un esempio di come usare il controllo **RichTextArea**:

```

<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>

```

ottenendo ciò che segue:



creare un mini text editor diventa ormai una cosa banale anche in Silverlight, al momento causa un bug nel designer di Visual Studio 2010 gli spazi nei paragrafi non vengono correttamente renderizzati a design time.

PrintDocument

Finalmente la possibilità di stampare direttamente da Silverlight senza passare attraverso l'HTML Bridge, il tutto attraverso una classe familiare agli sviluppatori Windows Forms ovvero **PrintDocument**. Supponendo di avere questo frammento di xaml:

```

<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>

```

per stamparne il contenuto basta scrivere questo codice:

```

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;
}

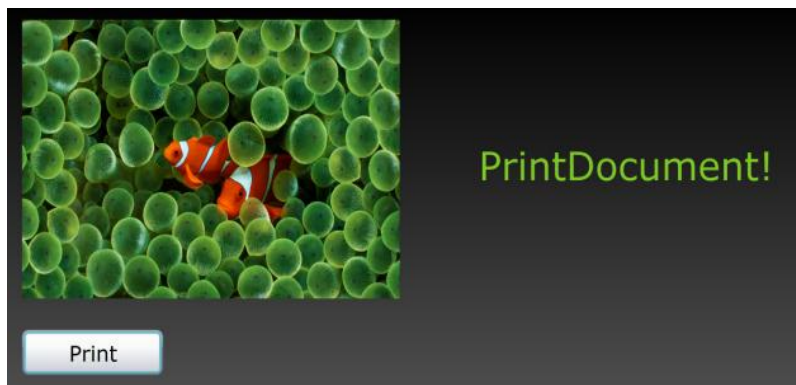
void OnEndPrinting(object sender, EndPrintEventArgs e)
{
    Debug.WriteLine("Print Completed");
}

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

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

```

Ottenendo in uscita:



Gestione WebCam e Microfono

Interagire con webcam è microfono significa solitamente recuperare l'elenco delle webcam installate nel sistema, il tutto si ottiene con questo snippet:

```

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

```

```
}
```

Oppure si può semplicemente ottenere la webcam predefinita usando:

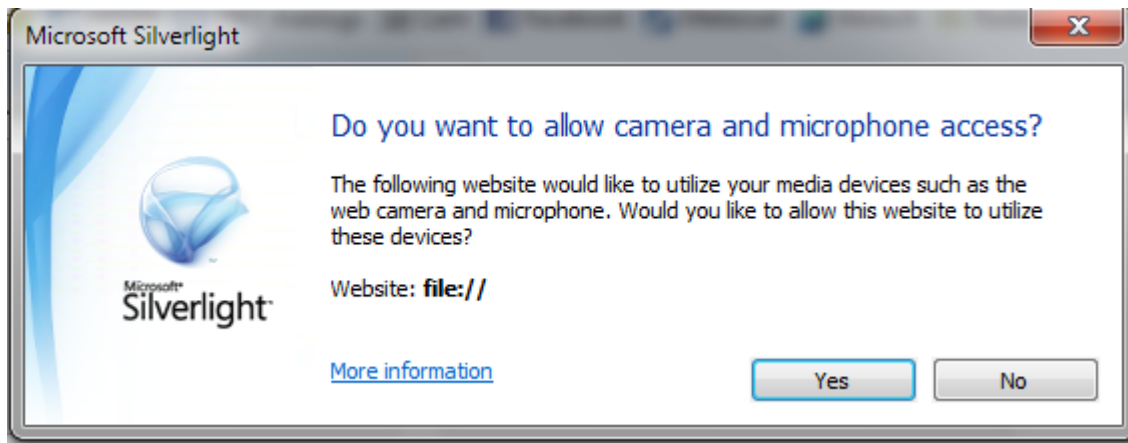
```
VideoCaptureDevice device = CaptureDeviceConfiguration.DefaultVideoCaptureDevice();
```

Ottenere un anteprima a video della webcam è veramente molto semplice:

```
CaptureSource cs = new CaptureSource();
```

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    VideoCaptureDevice device = CaptureDeviceConfiguration.DefaultVideoCaptureDevice();
    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();
        }
    }
}
```

Il metodo **RequestDeviceAccess** fa apparire all'utente una finestra di conferma di accesso alla webcam



Per memorizzare in un controllo image il contenuto della webcam è sufficiente usare il metodo **AsyncCaptureImage**:

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

La gestione del microfono avviene in maniera analoga, il metodo **GetAvailableAudioCaptureDevices** ritorna l'elenco degli ingressi audio disponibili:

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

Volendo usare il dispositivo predefinito è invece possibile usare:

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

Per ottenere l'insieme dei campioni audio è necessario creare una classe che eredita da **AudioSink** e implementarne i relativi metodi astratti:

```
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...  
    }  
}
```

A questo punto non ci rimane che associare alla relativa proprietà CaptureSource la fonte

```
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();  
    }  
}
```

Ricordatevi di invocare il metodo **Stop** quando non desiderate più interagire con la webcam o il microfono.

DataBinding

E' ora possibile usare binding con elementi che ereditano direttamente da DependencyObject ciò significa che questo codice, non applicabile in Silverlight3, ora funziona correttamente:

```

<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>

```

La classe Binding ora supporta **StringFormat**, **IDataErrorInfo** e **INotifyDataErrorInfo**

```

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

```

Value:350.000

NotificationWindow

Se la vostra applicazione sta girando in modalità out-of-browser è possibile visualizzare una finestra di notifica (a.k.a 'Toast' window) per un determinato intervallo di tempo nell'angolo in basso a destra.

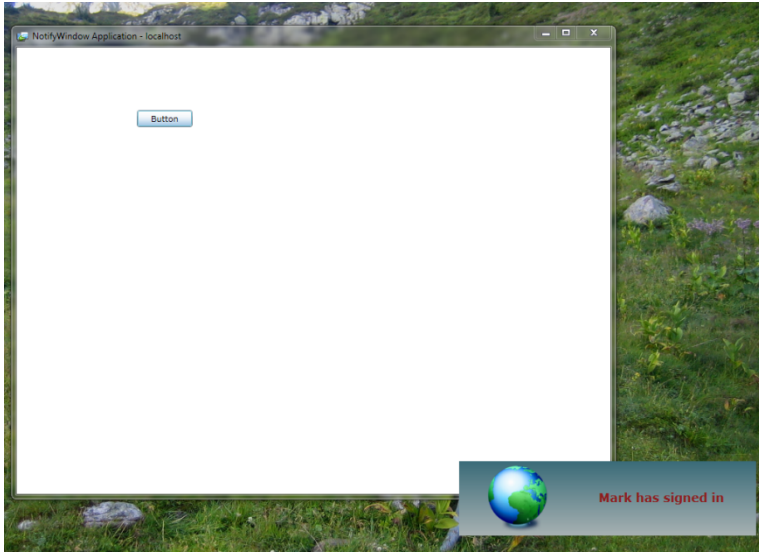
Ecco il codice per visualizzare una notifica per 4 secondi:

```

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 è uno UserControl che verrà utilizzato come contenuto della NotificationWindow la cui dimensione non può superare i 400x100 pixels. Il codice produce il seguente risultato:



HTML Brush

Il contenuto del controllo WebBrowser può essere usato in sostituzione di una qualsiasi brush grazie alla classe **HTMLBrush** a patto che l'applicazione sia in modalità Out-Of-Browser. Un esempio di utilizzo è riportato nell'esempio che segue:

```
<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>
```

Affinche il contenuto renderizzato da HTMLBrush sia sincronizzato con il contenuto di WebBrowser è necessario invocare il metodo **Redraw**:

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

Navigation Extensibility

Il sistema di navigazione introdotto con Silverlight3 può essere esteso associando alla proprietà **ContentLoader** un'istanza di una classe che implementa **INavigationContentLoader** (in System.Windows.Navigation) la quale può controllare la navigazione tra le varie pagine offrendo ad, ad esempio, la possibilità di scaricare alcune pagine on demand, oppure effettuare redirezione tra una pagina e l'altra. La navigazione attuale è gestita dalla classe **PageResourceContentLoader**.

L'implementazione di INavigationContentLoader è la seguente:

```
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 di files in applicazione Silverlight

E' possibile trascinare files da Windows Explorer direttamente in un applicazione Silverlight4: gestire questa operazione è semplice, basta impostare sull'elemento di destinazione la proprietà **AllowDrop=true** e gestirne l'evento **Drop**:

```
<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);
    }
}
```

Accesso alla ClipBoard

Sebbene attualmente limitato a testo Unicode Silverlight4 permette di scambiare informazioni con la **Clipboard** di sistema, nel caso l'applicazione non stia girando in elevation mode al primo accesso verrà chiesto all'utente di confermare l'accesso alla clipboard da parte dell'applicazione Silverlight.

L'utilizzo è basato sui due semplici metodi riportati di seguito:

```
//Copy
Clipboard.SetText(textBox1.Text);

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

Right-Click Mouse events

In Silverlight4 la classe UIElement si arricchisce di due interessanti eventi: **MouseRightButtonDown** e **MouseRightButtonUp**, eventi precedentemente gestiti dal plug-in per visualizzare il menu di configurazione. Il menu è tutt'ora presente ma lo si può rimuovere semplicemente gestendo l'evento MouseRightButtonDown:

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

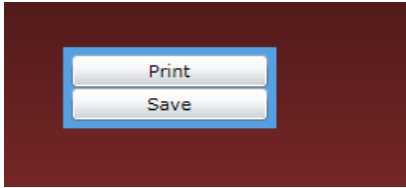
private void OnMouseRightButtonDown(object sender, MouseButtonEventArgs e)
{
    e.Handled = true;
}
```

Sebbene non sia previsto un controllo ContextMenu, attraverso l'uso di un popup si può ottenere un risultato analogo:

```
<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!

I .NET RIA Services cambiano nome e diventano ufficialmente **WCF RIA Services**, in questo rilascio è stato fatto parecchio refactoring ed è quindi probabile che le applicazioni basate sulla July 09 CTP non compilino più, nulla comunque che non si possa sistemare in breve tempo.

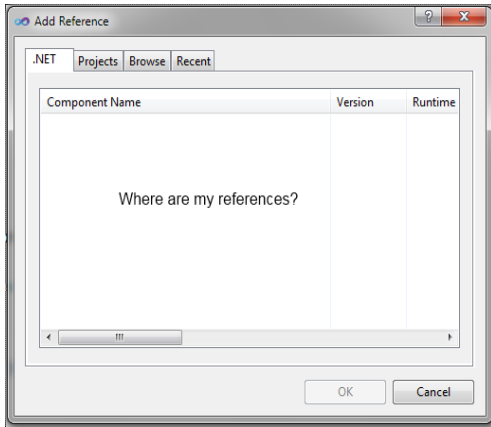
Le più importanti breaking changes riguardano invece gli attributi **[Custom]** e **[ServiceOperation]** che vengono sostituiti come indicato nella tabella che segue:

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

Nota: La release attuale è compatibile solo con Visual Studio 2010 Beta2.

Known Issues

Quando cercate di aggiungere un riferimento ad un assembly in un progetto Silverlight 4.0 la finestra dei riferimenti appare completamente vuota, e quindi per aggiungere il riferimento bisogna usare la tab Browse e aggiungere il riferimento manualmente.



Conclusion

Quelle elencate sono solo una parte di tutte le novità presenti in Silverlight 4.0, vi invito ad esplorarle tutte installando la beta e consultando i contenuti del file **silverlight.chm**