Silverlight 5 Beta

Technical Features

Microsoft®
Silverlight™

Contents

About Silverlight 5

Silverlight delivers the richest set of capabilities available to developers today through a Web-browser plug-in. Announced last December and shipping later this year, Silverlight 5 makes further advances in media, application development and user experience, adding over 40 new features.

The Silverlight 5 Beta provides developers their first opportunity to start exploring these new capabilities. More information on the Silverlight 5 announcements is available at http://www.microsoft.com/silverlight/future.

Features included in the Beta

The Silverlight 5 Beta is a major step toward the final release and includes many of the significant features already announced. The Beta download link and full list of features is available at http://www.silverlight.net.

Visual Studio and Expression Blend tooling support for the Silverlight 5 Beta is also available.

Multiple Window Support

Multiple window support enables a trusted application to dynamically create additional top level windows.

```
Window tearOffWindow = new Window();
tearOffWindow.Height = 400;
teaOffWindow.Width = 600;
tearOffWindow.Top = 24;
tearOffWindow.Left = 30;
tearOffWindow.Title = “Stock Console”;
tearOffWindow.Content = myUIElement; //Set content to someFrameworkElement
tearOffWindow.Visibility = Visibility.Visible; //Display the Window
```

Ancestor RelativeSource Binding

Ancestor RelativeSource Binding enables a DataTemplate to bind to a property on the control that contains it, equivalent to FindAncestor and AncestorType, AncestorLevel bindings in WPF.

```
<UserControl x:Class=”WpfApplication1.UserControl1”
                    xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
                    xmlns:x=”http://schemas.microsoft.com/winfx/2006/xaml”>
      <ContentControl Tag=”SomeValue”>
            <HeaderdContentControl>
                  <HeaderedContentControl.Header>
                        <TextBlock Text=”{Binding Tag, RelativeSource=
                              {RelativeSource,AncestorType=ContentControl,
                              AncestorLevel=2}}” />
                  </HeaderedContentControl.Header>
                  <Button>Click Me!</Button>
            </HeaderdContentControl>
      </ContentControl>
   </UserControl>
```

Implicit DataTemplates

This feature enables the following capabilities:

- ContentPresenter DataTemplates can be selected based upon the content type.
- Implicit definition of DataTemplates
- Dynamically update the ContentPresenter DataTemplate when its content changes
- Enable proper scoping of DataTemplates

```
<DataTemplate DataType="local:Book">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="60" />
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="20" />
                </Grid.ColumnDefinitions>
                <Rectangle Fill="Transparent" Grid.Column="2"/>
                <Image Source="{Binding ImageSource}"
                      HorizontalAlignment="Left" Grid.RowSpan="2" Width="60"/>
                <TextBlock Text="{Binding Title}" Grid.Row="0" Grid.Column="1" />
                <Grid Grid.Row="1" Grid.Column="1">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="100" />
                        <ColumnDefinition Width="100" />
                        <ColumnDefinition Width="Auto" />
                    </Grid.ColumnDefinitions>
                    <TextBlock Text="{Binding Author, StringFormat='by {0}'}"/>
                    <Button Content="Edit" Margin="4,0" Grid.Column="2"
                        Height="26" VerticalAlignment="Top"
                        Style="{StaticResource StandardButtonStyle}"
                        Command="{Binding DataContext.EditBookCommand,
                        RelativeSource={RelativeSource FindAncestor,
                             AncestorType=UserControl}}"
                        IsEnabled="{Binding IsSelected,
                             RelativeSource={RelativeSource FindAncestor,
                             AncestorType=ListBoxItem}}"
                        />
                </Grid>
            </Grid>
        </DataTemplate>

        <DataTemplate DataType="local:ClearanceBook">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="60" />
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="20" />
                </Grid.ColumnDefinitions>
                <Rectangle Fill="Red" Grid.Column="2"/>
```

```
        <Image Source="{Binding ImageSource}"
               HorizontalAlignment="Left" Grid.RowSpan="2" Width="60"/>
        <TextBlock Text="{Binding Title}" Grid.Row="0" Grid.Column="1" />
        <Grid Grid.Row="2" Grid.Column="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="100" />
                <ColumnDefinition Width="100" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <TextBlock Text="{Binding Author, StringFormat='by {0}'}"/>
            <ComboBox VerticalAlignment="Top" Margin="4,0"
                 Width="90" Grid.Column="1"
                 SelectedItem="{Binding Deal, Mode=TwoWay}"
                 ItemTemplate="{StaticResource DealItemTemplate}"
                 IsEnabled="{Binding IsSelected,
                     RelativeSource={RelativeSource FindAncestor,
                            AncestorType=ListBoxItem}}"
                     ItemsSource="{Binding DataContext.Deals,
                            RelativeSource={RelativeSource Mode=FindAncestor,
                            AncestorType=UserControl}}">
            </ComboBox>
            <Button Content="Edit" Margin="4,0" Grid.Column="2"
                 Height="26" VerticalAlignment="Top"
                 Style="{StaticResource StandardButtonStyle}"
                 Command="{Binding DataContext.EditBookCommand,
                        RelativeSource={RelativeSource FindAncestor,
                            AncestorType=UserControl}}"
                  IsEnabled="{Binding IsSelected,
                        RelativeSource={RelativeSource FindAncestor,
                            AncestorType=ListBoxItem}}"
                />
        </Grid>
    </Grid>
</DataTemplate>

<DataTemplate DataType="local:BestSellerBook">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="60" />
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="20" />
        </Grid.ColumnDefinitions>
        <Rectangle Fill="Yellow" Grid.Column="2"/>
        <Image Source="{Binding ImageSource}" HorizontalAlignment="Left"
              Grid.RowSpan="2" Width="60"/>
        <TextBlock Text="{Binding Title}" Grid.Row="0" Grid.Column="1" />
        <Grid Grid.Row="1" Grid.Column="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="100" />
                <ColumnDefinition Width="100" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <TextBlock Text="{Binding Author, StringFormat='by {0}'}"/>
            <TextBlock Text="   BEST SELLER" FontWeight="Bold" Grid.Column="1"/>
            <Button Content="Edit" Margin="4,0" Grid.Column="2"  Height="26"
                 VerticalAlignment="Top"
                 Style="{StaticResource StandardButtonStyle}"
```

```
                    Command="{Binding DataContext.EditBookCommand,
                        RelativeSource={RelativeSource FindAncestor,
                                AncestorType=UserControl}}"
                        IsEnabled="{Binding IsSelected,
                                RelativeSource={RelativeSource FindAncestor,
                                        AncestorType=ListBoxItem}}"
                    />
                </Grid>
            </Grid>
        </DataTemplate>
```

ClickCount

Enables Multi-click input on left and right mouse button.

```
    private void OnMouseDownClickCount(object sender, MouseButtonEventArgs e)
    {
        // Checks the number of clicks.
        if (e.ClickCount == 1)
        {
            // Single Click occurred.
        }

        if (e.ClickCount == 2)
        {
            // Double Click occurred.
        }

        if (e.ClickCount >= 3)
        {
            // Triple Click occurred.
        }
    }
```

Binding on Style Setter

Binding in style setters allows bindings to be used within styles to reference other properties.

```
<Style x:Key=”TextBlockStyle2” TargetType=”TextBlock”>
        <Setter Property=”FontFamily”
                Value=”/SL5;Component/Fonts/Fonts/.zip#Segoe UI/>
        <Setter Property=”FontSize” Value=”0,3,0,0”/>
                <Setter Property=”Foreground” Value=
                        ”{Binding Source={StaticResource SysColors},Path=ControltextBrush}”/>
</Style>
```

Trusted applications in-browser

Silverlight 5 enables trusted applications to run in-browser for the first time. The Beta includes support for two features: Elevated In-Browser via Group Policy & In-Browser HTML support w/Group Policy.

```
<Deployment.OutOfBrowserSettings>
        ::
        <OutOfBrowserSettings.SecuritySettings>
                <SecuritySettings ElevatedPermissions=”Required” />
        <OutOfBrowserSettings.SecuritySettings>
        ::
</Deployment.OutOfBrowserSettings>
```

Additional information

- The Xap must be signed and interest must be present in the Trusted Publishers Certificate store.
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Silverlight\ : add the following DWORD key "AllowElevatedTrustAppsInBrowser" with 0x00000001.

AllowInstallOfElevatedTrustApps

With respect to trusted in-browser apps, disabling the AllowInstallOfElevatedApps regkey will disable install trusted apps but not prevent enabled apps to consume trusted functionality in-browser. Specifically, if an app has been enabled to run trusted in-browser but the AllowInstallOfElevatedTrustApps has been disabled, then the app will run as a trusted app in browser but not be installable.

AllowLaunchOfElevatedTrustApps

With respect to trusted in-browser apps, disabling the AllowLaunchOfElevatedApps regkey will disable execution trust feature consumption of any application (in-browser or OOB). Specifically, if an app has been enabled to run trusted in-browser but the AllowLaunchOfElevatedTrustApps has been disabled, then the app will run as a partial trust app in browser.

Realtime Sound (low-latency Audio)

Enables pre-loading of an audio source for precision timing of playback. Multiple playback instances are supported through creation of SoundEffectInstance objects.

```
using Microsoft.Xna.Framework.Audio;
...


var streamInfo = Application.GetResourceStream(new Uri("Windows_44100_16_Mono.wav",
UriKind.RelativeOrAbsolute));
SoundEffect soundEffect = SoundEffect.FromStream(streamInfo.Stream);
soundEffect.Play(); // Default playback mode
soundEffect.Play(0.8f, -0.5f, 1.0f); // Explicit volume, pitch, pan

// Using explicitly created SoundEffectInstance
{
       SoundEffectInstance inst = soundEffect.CreateInstance();
       inst.Volume = 0.85f;
       inst.Play();
}
```

Variable Speed Playback (“Trick Play”)

This API enables development of fundamental player experiences such as fast-forward and rewind and variable speed playback at common speeds (1.2x, 1.4x) – a very common scenario for enterprise training content. The MediaElement.Rate property supports the values -8.0, -4.0, 0.5, 1, 4.0, 8.0. Note : AudioPitch correction is not present in the Beta but will be added for the final release.

```
       private void Button_Click(object sender, RoutedEventArgs e)
       {
              media.PlaybackRate = 2.0;
       }
```

1.1. Known issue

Setting the PlaybackRate property in XAML resets the value to 1.0 before the media is opened.

Linked Text Containers

Enables RichTextBox controls to be associated and text to overflow from one into the next. multiple RichTextBoxOverflows can be chained (from an initial RichTextBox) to automatically spread text across a pre-defined free-form layout or multiple column layout.

```
    <StackPanel Width="400">
        <RichTextBox x:Name="MasterRTB" Width="50" Height="50"
OverflowContentTarget={Binding ElementName="overflowContainer"}>
            <Paragraph>
                There is noooooooooo way all of this content is going to fit into a RTB
that is 50x50!
            </Paragraph>
        </RichTextBox>
        </RichTextBoxOverflow x:Name="overflowContainer">
    </StackPanel>
```

Text Tracking & Leading

Tracking/leading set more precisely how far apart each character is for extra creative control, Adding finer grained control over the spacing between letters in words in the RichTextbox control.

```
    <TextBlock FontSize="12" CharacterSpacing="300">
The intercharacter spacing of this text has been loosened.
    </TextBlock>
```

Custom Markup Extensions

Custom markup extensions allow you to run custom code from XAML. Markup extensions allow code to be run at XAML parse time for both properties and event handlers, enabling cutting-edge MVVM support.

First, put this namespace in your XAML

```
xmlns:local="clr-namespace:CMEInvokeMethodsSL"
```

Next add some Xaml

```
<Grid x:Name="LayoutRoot">
      <Canvas>
          <TextBox Text="Some Text"
                   GotFocus="{local:MethodInvoke Path=TextChanged}"
                   Width="102" Canvas.Left="35" Canvas.Top="42" />
          <ComboBox Height="34" Name="cbOptions"
                    SelectionChanged="{local:MethodInvoke Path=OptionSelected}"
                    Width="120" Canvas.Left="143" Canvas.Top="42">
              <ComboBoxItem Content="A" />
              <ComboBoxItem Content="B" />
              <ComboBoxItem Content="C" />
              <ComboBoxItem Content="D" />
          </ComboBox>
          <Button Name="InvokeButton"
                  Click="{local:MethodInvoke Path=SaveData}"
                  Content="Invoke Button"  Margin="128,108,119,115" Canvas.Left="141"
Canvas.Top="-64" />
      </Canvas>
  </Grid>
```

Add the following class to your code. This implements a custom markup extension using the IMarkupExtension interface.

```
using System;
using System.Reflection;
using System.Windows;
using System.Windows.Markup;
using System.Xaml;

namespace BookShelf
{
    public class MethodInvokeExtension : IMarkupExtension<object>
    {
        public string Method { get; set; }

        private object _target;
        private MethodInfo _targetMethod;

        public object ProvideValue(IServiceProvider serviceProvider)
        {
            try
            {
```

```
Delegate del = null;

// IProvideValueTarget - Provides information about the target object
IProvideValueTarget targetProvider =
            serviceProvider.GetService(typeof(IProvideValueTarget))
            as IProvideValueTarget;
// IXamlTypeResolver - Handles types that use xml prefixes
IXamlTypeResolver xamlResolver =
            serviceProvider.GetService(typeof(IXamlTypeResolver))
            as IXamlTypeResolver;
// IRootObjectProvider -
//Provides access to the root object (the Framework Element)
IRootObjectProvider rootObject =
            serviceProvider.GetService(typeof(IRootObjectProvider))
            as IRootObjectProvider;

if (targetProvider != null)
{
    // Get the target element (ex: ListBox)
    var targetObject = targetProvider.TargetObject;
    // Get the target element's event info
    EventInfo targetEvent = targetProvider.TargetProperty as EventInfo;

    if (targetEvent != null)
    {
        // Get the event's parameters and types for the target element
        ParameterInfo[] pars = targetEvent.GetAddMethod().GetParameters();
        Type delegateType = pars[0].ParameterType;
        MethodInfo invoke = delegateType.GetMethod("Invoke");
        pars = invoke.GetParameters();

        // Create the function call (to invoke the event handler)
        Type customType = typeof(MethodInvokeExtension);
        var nonGenericMethod = customType.GetMethod("PrivateHandlerGeneric");
         MethodInfo genericMethod =
            nonGenericMethod.MakeGenericMethod(pars[1].ParameterType);

        // Grab the root FrameworkElement
        if (rootObject != null)
        {
            FrameworkElement rootObjFE = rootObject.RootObject
                     as FrameworkElement;

            if (rootObjFE != null)
            {
                // Grab the FrameworkElement's DataContext
                //and the method name exposed by it
                _target = rootObjFE.DataContext;
                _targetMethod = _target.GetType().GetMethod(Method);
                // Make sure the FE's DataContext has the Method name
                //or get out.
                if (_target == null) return null;
                if (_targetMethod == null) return null;
                // Create the event handler and attach it from
                //the target element to the DataContext's method name
                del = Delegate.CreateDelegate(
                     delegateType, this, genericMethod);
            }
        }
    }
}
```

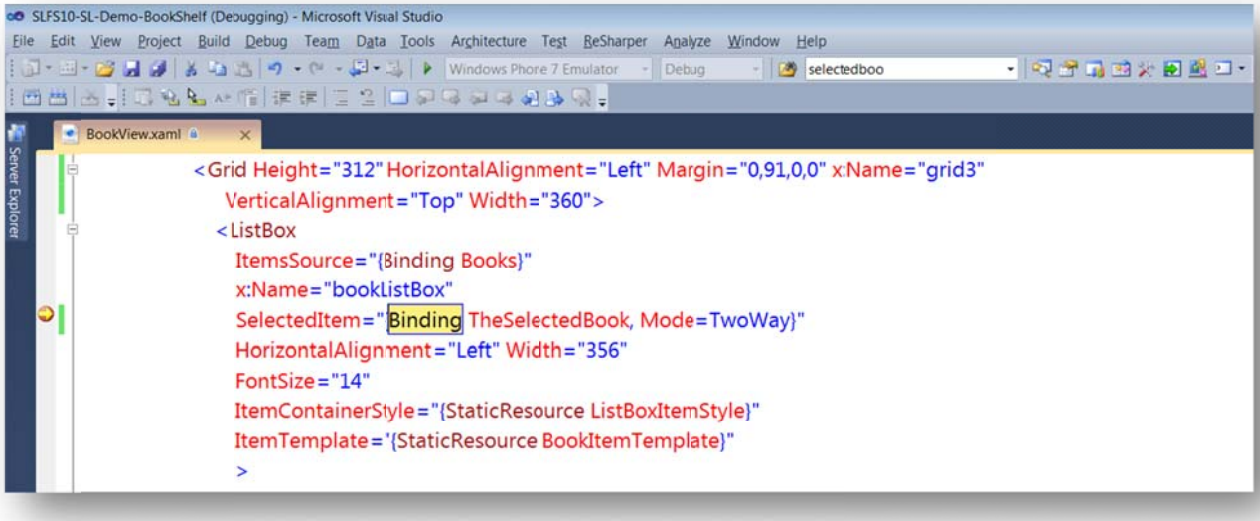
```
                return del;
            }
            catch (Exception ex)
            {
                string innerex = ex.InnerException.ToString();
                return null;
            }
        }

        // Invoke the generic handler
        public void PrivateHandlerGeneric<T>(object sender, T e)
        {
            _targetMethod.Invoke(_target, null);
        }
    }
}
```

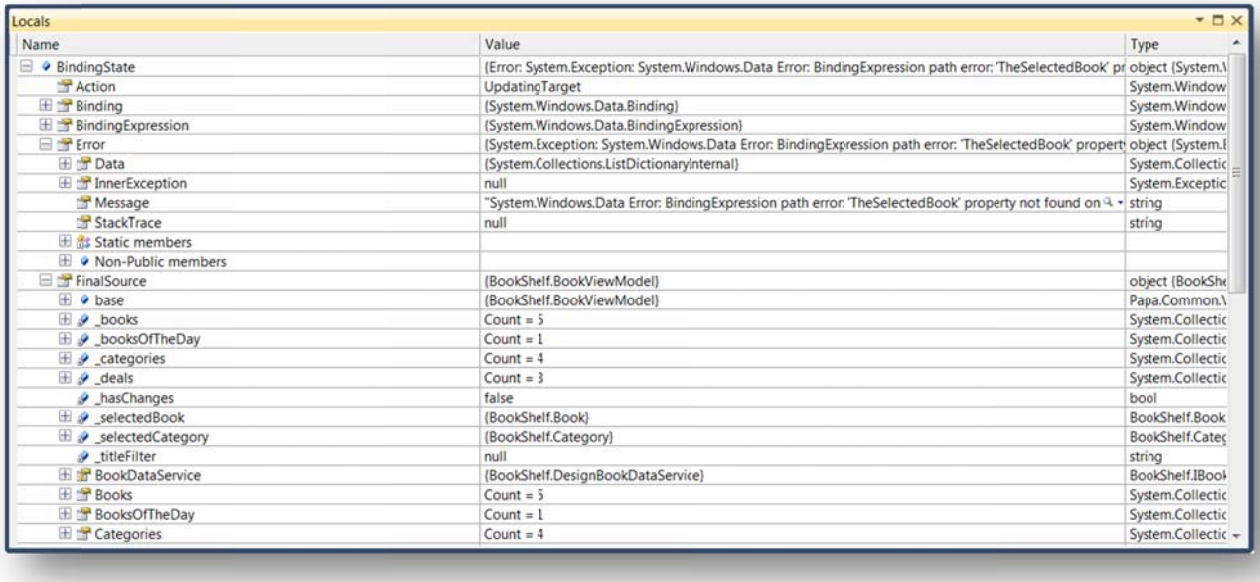
XAML Binding Debugging

Data binding debugging allows breakpoints to be set in XAML binding expressions so developers can step through binding issues, examine the Locals window, and create conditional breakpoints.

Here you can see Visual Studio stopped due to a binding error:

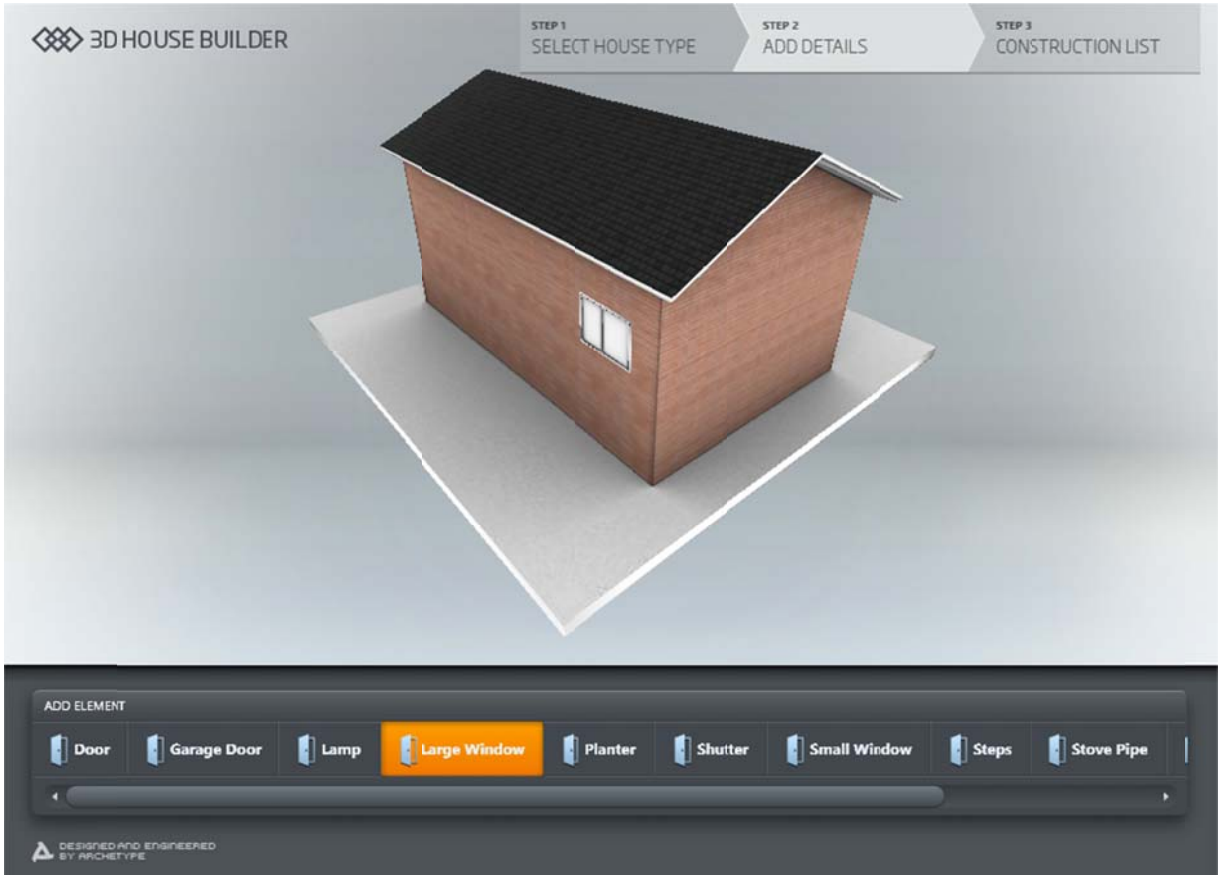


And the Locals window to assist with debugging the binding:



3D Graphics API

Silverlight 5 now has a built-in XNA 3D graphics API to enables you to build rich, GPU accelerated visualizations and rich user experiences. This includes a new 3D drawing surface control as well as immediate mode GPU access, shaders and effects for 2D and 3D use.



Additional Silverlight 5 Features Included in this Beta

- Hardware video decode for H.264 playback.
- Multi-core background JIT support for improved startup performance.
- ComboBox type ahead with text searching.
- Full keyboard support in full-screen for trusted in-browser applications, enables richer kiosk and media viewing applications in-browser.
- Default filename in SaveFileDialog – Specify a default filename when you launch the SaveFileDialog.
- Unrestricted filesystem access – trusted applications can Read write to files in any directory on the filesystem.
- Improved Graphics stack – The graphics stack has been re-architected to bring over improvements from WP7, such as Independent Animations.
- Performance optimizations –
- XAML Parser performance optimizations.
- Network Latency optimizations.
- Text layout performance improvements.
- Hardware acceleration is enabled in windowless mode with Internet Explorer 9.

Silverlight 5 Features not included in this Beta

The Beta is a step towards the final release of Silverlight 5. The full feature set planned for Silverlight 5 was announced at the Firestarter event in December 2011 and also includes the following features not shipped in this Beta:

- Support for Postscript vector printing enables users to create reports and documents, including the ability to create a virtual print view different from what is shown on the screen.
- Improved power awareness prevents the screen saver from being shown while watching video and allows the computer to sleep when video is not active.
- Remote control support, allowing users to control media playback
- DRM advancements that allow seamless switching between DRM media sources.
- Enhanced OpenType support.
- Fluid user interface enables smoother animation within the UI. Inter-Layout Transitions allow developers to specify animations to apply when elements are added, removed or re-ordered within a layout. This provides smoother user experiences when, for example, items are inserted into a list.
- Text clarity is improved with Pixel Snapping.
- The DataContextChanged event is being introduced.
- WS-Trust support: Security Assertion Markup Language authentication token.
- 64 bit support
- COM Interop for trusted applications running in-browser.
- Calling unmanaged code using P/Invoke from trusted applications in and out of browser.
- The Pivot viewer control enables a visual way to rapidly sort through large collections of graphical data, for example a catalog of movies represented by images of dvd covers, using intuitive transitions to show the effects of filters on the result set.

Legal Notice

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.