Silverlight 3
for WPF & ASP.NET
Programmers

M. MacDonald, Pro Silverlight 3 in C#, Apress, 2009
A. Michail, Essential Silverlight 3, Addison-Wesley, 2009
MSDN
http://silverlight.net
Contents

- Introduction
- Architecture
- WPF compatibility
- Application
- Integration with a Web page
- Features
- Performance
Introduction
Microsoft Silverlight

- Silverlight is Microsoft’s implementation of a cross-browser, cross-platform client framework that allows designers and developers to deliver Rich Internet Applications (RIA) embedded in Web pages.
- Silverlight-based applications run in the browser.
- Silverlight applications can be created using any .NET Framework language.
## Supported Systems and Browsers

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Browser</th>
<th>IE 8</th>
<th>IE 7</th>
<th>IE 6</th>
<th>Firefox 2, 3</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Windows Server 2008 R2</td>
<td></td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Windows Vista</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Windows Server 2008</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Windows XP SP2, SP3</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Windows 2000 SP4 + all post-SP4 hotfixes, updates, and security updates</td>
<td></td>
<td>-</td>
<td>-</td>
<td>Yes (SL 2, 3 only)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Windows Server 2003 (excluding IA-64)</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Mac OS 10.4.8+ (PowerPC)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes (SL 1.0 only)</td>
<td>Yes (SL 1.0 only)</td>
</tr>
<tr>
<td>Mac OS 10.4.8+ (Intel-based)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Linux, FreeBSD, SolarisOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Firefox 3 recommended (Moonlight 2.0 Beta - SL 2 support)</td>
<td></td>
</tr>
</tbody>
</table>

Minimum requirements:
- PC with Windows - X86 or x64 500 MHz or higher processor with 128 MB of RAM
- Mac OS 10.4.8+ (PowerPC) - PowerPC G4 800 MHz or higher processor with 128 MB of RAM
- Mac OS 10.4.8+ (Intel-based) - Intel Core Duo 1.83 GHz or higher processor with 128 MB of RAM
Silverlight Features

- WPF and XAML
- Extensions to JavaScript
- Cross-browser, cross-platform support
- .NET Framework programming model and associated tools
  - Base class library
  - CLR (memory management, garbage collection, type safety checking, and exception handling)
- Networking support
- Data (XML, LINQ, LINQ to XML)
- DLR (dynamic compilation and execution of scripting languages such as JavaScript and IronPython)
Silverlight Tools

- Visual Studio 2008 or Visual Web Developer Express Edition
- Silverlight 3 Tools for Visual Studio 2008 SP1 (free)
  - Developer version of the Silverlight plug-in
  - Silverlight 3 SDK
  - C# and VB project templates, XAML IntelliSense and code generators, debugging support
  - Still no graphical design-time features (also no preview)
- Visual Studio 2010
  - Visual designer based on Blend (with live preview)
- Expression Blend 3
  - Rich support for creating Silverlight user interface
  - A coding editor included with limited functionality
Silverlight Tools cont.

- Silverlight Toolkit (free)  [http://silverlight.codeplex.com]
  - A set of controls and components that extend the features of Silverlight
  - Silverlight Unit Testing Framework included
- Deep Zoom Composer (free)
  - It allows the preparation of images for use with the Deep Zoom feature
- Silverlight Spy  [http://silverlightspy.com]
- Expression Design
  - Imports Adobe Photoshop and Illustrator files, exports XAML
- Expression Media Encoder
  - Converts video and audio
Silverlight History

- **Silverlight 1.0** – September, 2007
  - Developed as WPF/E – Windows Presentation Foundation/Everywhere
  - No CLR, coding only using JavaScript (no compilation)
  - No visual controls

- **Silverlight 2** – October, 2008
  - Developed as Silverlight 1.1, renumbered in March, 2007
  - Cross-platform CLR
  - A set of standard controls

- **Silverlight 3** – July, 2009
  - Improvements and extensions
  - Out of browser applications

- **Silverlight 4 Beta 1** – November, 2009
Silverlight 3 New Features

- 3-D effects (perspective transforms)
- Bitmap effects and pixel shaders
- WriteableBitmap
- Bitmap caching
- New media supported (raw video/audio and H.264)
- New and improved controls for Silverlight 3
  - ListBox, SaveFileDialog, HyperlinkButton
  - TreeView, TreeViewItem, HierarchicalDataTemplate, AutoCompleteBox
  - ChildWindow, DataPager, DescriptionViewer, Label
- Changing styles at run time
- BasedOn styles
- Animation easing functions
- FontSource support in glyphs
- Out-of-browser support
- Communication between local Silverlight-based applications (Local Messaging)
- Application library caching
- Application extension services
- Data binding improvements
- Browser zoom support
- Navigation
- Networking
Architecture
Silverlight Architecture
Silverlight Plug-In

- It is the primary entry point
- It represents an instance of the application and all services run for the lifetime of the plug-in
  - Each Silverlight instance has its own AppDomain, even if two or more instances are placed on the same page
- Silverlight responds to draw requests, input events, and timer events
  - In response to these events, Silverlight invokes event handlers (in the same thread)
  - If the application code takes significant time in response to an event, it is possible to cause the Web browser to become unresponsive
Downloader

- Early during the startup process, the Silverlight runtime downloads the application XAP
  - If that XAP contains references to other files on your Web server or there are explicit download requests, Silverlight also initiates those downloads

- All download requests are made through the hosting browser
  - Using the browser’s cache and proxy settings
  - There is limitation of two simultaneous downloads per domain
  - The same security context as the hosting page is used
XAML Parser

- As the application loads, the XAML parser parses the XAML files and instantiates and initializes those objects
  - This construction process is semantically equivalent to programmatically constructing the objects in the application startup event
  - Initializing large numbers of objects is often faster when done through XAML than through code due to a number of optimizations made within the Silverlight runtime
The Silverlight CoreCLR is based on the version 2.0 of the CLR

- The execution engine and virtual machine are the same (the type system, metadata, the garbage collection, the JIT compiler, and the thread pool)

- There is no Code Access Security in the Silverlight CLR

- Silverlight is oriented at single user interactive applications
  - It impacts on optimizations (e.g. the garbage collector)

- There is no dependence on the desktop .NET Framework which may be installed on the host OS
Silverlight BCL restrictions:
- Limited System.Security namespace
- No non-generic collections (e.g. ArrayList)
- No own globalization data, using the globalization functionality provided by the host OS instead

Silverlight BCL improvements:
- Isolated storage (existing in full BCL, but not used on such scale)
- Some inspiration and guidance was taken from .NET Compact Framework
Element Tree

- All Silverlight display and user interface features operate on the concept of building a tree and manipulating it.

- The basic model to display content is to:
  - Construct a tree of elements through code or XAML.
  - Write code to respond to events that can manipulate that tree of elements.
Rendering

- Silverlight includes its own rendering system rather than using the one available on the target operating system
  - It gives consistent visual quality and performance independent of operating system and browser
  - It also supports more advanced rendering features than are available in HTML
    - e.g. anti-aliased vector graphics primitives, video, effects, 3D,
WPF Compatibility
WPF Features Not in Silverlight

- Commanding
  - The interface `ICommand` exists in Silverlight, but there is no `CommandManager` implementation that does actual commanding

- Simple inline XAML content

  `<Button>My button content</Button>
  <Button Content="My button content"/>`

- Printing support or flow document support

- Dynamic resources
  - In Silverlight, all resource references to keyed resources in XAML are static
WPF Features Not in Silverlight cont.

- **BitmapImage** as the only available **ImageSource** type
- **TileMode** property for **TileBrush** objects
- **ValueConversion** attribute
  - In WPF, it allows the author of a value converter to specify the data types involved in the implementation of the converter
- **ITypedList, InstanceFactory, or BindingList(T)**
- **MarkupExtension** base class
  - In WPF, it allows to create custom markup extensions
- **Freezable** class (and its functionality)
WPF Controls Not in Silverlight

- Menu, ContextMenu, Ribbon
- ListBoxItem, ListView, ListViewItem
- ToolBar, ToolBarOverflowPanel, ToolBarPanel, ToolBarTray
- Window, ResizeGrip, NavigationWindow, RibbonWindow, StatusBar, StatusBarItem
- WindowsFormsHost
- GroupBox, InkCanvas, RichTextBox, Separator, TabPanel, UniformGrid, Viewport3D, WebBrowser
WPF Features with Limited Support in SL

- Data binding
  - No ADO.NET
  - No ICustomTypeDescriptor, IListSource, or IBindingList
  - No binding directly to XML data
  - No BindingMode.OneWayToSource binding
  - No notification for source and target updates
  - No XPath binding
  - No CollectionViewCollectionViewSource or CollectionViewSource class

- Styles
  - No support for implicit styles applied using the TargetType attribute value
WPF Features with Limited Support in SL

- **Triggers**
  - Silverlight support for triggers is limited to using `EventTrigger` for the `Loaded` event and the `BeginStoryboard` action.
  - `VisualStateManager` introduced in Silverlight allows to accomplish some of the missing functionality.

- **Routed events**
  - Silverlight includes the concept of routed events for a small subset of events.
  - It does not include Preview events or class-handling of routed events.
  - It does not enable to create a new routed event.
WPF Features with Limited Support in SL

- Visual and logical trees
  - Silverlight does not include the concept of separate visual and logical trees as in WPF
  - The **Visual** class does not exist in Silverlight
- 3-D Graphics – limitations
- Dependency properties – some important differences
Same Feature, Different Approach

- Subpixel rendering
  - WPF handles subpixel rendering issues by using a feature called pixel snapping, which snaps lines to the nearest device pixel
  - Silverlight introduces the concept of layout rounding, which rounds object dimensions to the nearest whole integer
    - This feature can be turned off by setting the `UseLayoutRounding` flag to false
- XAML processing
  - In some cases, the parsing behavior differs from the parsing behavior in WPF
Silverlight-Only Features

- **VisualStateManager**
  - It helps define and manage the visual behavior based on the control state
  - By using it, there can be defined all the control states and the transitions between states in your control templates

- **Deep Zoom**
  - It allows zooming in and out of high-resolution images or collections of images

- **DataGrid** control
Application
Available Application Models

- Silverlight provides the following two distinct models for application development:
  - The managed API for Silverlight
    - uses code running on the Silverlight CLR
    - the managed API with compiled languages such as Visual Basic and C# or dynamic languages such as IronPython and IronRuby can be used
  - The JavaScript API for Silverlight
    - uses JavaScript code interpreted by the browser
- In one application both models cannot be mixed
  - The only exception is the splash screen that uses the JavaScript API and then transitions to the managed API when the application has loaded
Silverlight Application Example

- The 001.xap file is a .zip file containing:
  - 001.dll – compiled application's assembly
  - AppManifest.xaml

```xml
<Deployment
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    EntryPointAssembly="FirstApp"
    EntryPointType="FirstApp.App"
    RuntimeVersion="3.0.40624.0"
>
    <Deployment.Parts>
        <AssemblyPart x:Name="FirstApp" Source="FirstApp.dll" />
    </Deployment.Parts>
</Deployment>
```

- The App.xaml source file:

```xml
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             x:Class="FirstApp.App">
    <Application.Resources>
    </Application.Resources>
</Application>
```
Integration with a Web Page
Minimum HTML Code

```html
<!DOCTYPE HTML PUBLIC "-/W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <body>
    <object width="100%" height="50"
      data="data:application/x-silverlight-2,"
      type="application/x-silverlight-2">
      <param name="source" value="WebPageIntegration.xap"/>
    </object>
  </body>
</html>
```

- The **width** and **height** attributes are required for cross-browser compatibility.
- The **data** attribute is recommended to avoid performance issues on some browsers.
- The **type** attribute is required.
- The **param** element named **source** is required.
Properties of the Object used by SL

- **type** – should be: application/x-silverlight-2
- **data** – should be: data:application/x-silverlight-2,
- **width, height** – recommended to use (some problems with Firefox when missed)
  - Can be set either as a pixel or a percentage value
- **altHtml** - content that provides fallback cues such as download and installation information for Silverlight
  - Not recommended, provide this value as inner HTML for the object element
- **tabIndex, id, style** – general behaviour of the object element
- **culture, uiculture** - the culture settings that are used to retrieve satellite assemblies
Adding Silverlight using JavaScript

```html
<script type="text/javascript" src="Silverlight.js"></script>

<script type="text/javascript">
    Silverlight.createObjectEx({
        source: "WebPageIntegration.xap",
        parentElement: document.getElementById("sl4Host"),
        id: "sl4",
        properties: {
            width: "200",
            height: "50",
            background: "white",
            alt: "<!--not installed-->",
            version: "3.0.40624.0"
        },
        events: {
            onError: onSLError,
            onLoad: onSLLoad
        },
        initParams: "param1=value1,param2=value2",
        context: "row4"
    });
</script>

<script type="text/javascript">
    Silverlight.createObject(
        "WebPageIntegration.xap",
        document.getElementById("sl3Host"),
        "sl3",
        {
            width: "200",
            height: "50",
            background: "white",
            alt: "<!--not installed-->",
            version: "3.0.40624.0"
        },
        {
            onError: onSLError,
            onLoad: onSLLoad
        },
        "param1=value1,param2=value2",
        "row3",
    );
</script>
```
Properties of the Silverlight Plug-in

- **Source** – the address of the XAP file
- **Background** – the colour used to paint the background
- **InitParams** – string that can be used to pass custom initializations information
- **SplashScreenSource** – the location of a XAML splash screen to show while the XAP file is downloading
- **Windowless** – whether the Silverlight control is rendered in windowed mode (the default) or windowless mode (useful when the control should be shaped)
- **MinRuntimeVersion** - sets a value that indicates the earliest Silverlight version that is required in order to run a Silverlight-based application
- **AutoUpgrade** - whether a Silverlight plug-in version earlier than the specified `minRuntimeVersion` will attempt to update automatically
Properties of the Silverlight Plug-in cont.

- **AllowHtmlPopupWindow** – if the `HtmlPage.PopupWindow` method can be used to display a new browser window
- **EnableAutoZoom** - whether the host (for certain platforms) can invoke zoom behavior that increases the DPI
- **FullScreen** - whether the Silverlight plug-in is displayed as a full-screen plug-in
- **EnableHtmlAccess** – whether the Silverlight application should have access to the HTML object model
- **EnableNavigation** - whether the hosted content in the Silverlight plug-in can use a HyperlinkButton to navigate to external URIs
Properties of the Silverlight Plug-in cont.

- **EnableGPUAcceleration** - whether to use graphics processor unit (GPU) hardware acceleration for cached compositions
  - Must be set at startup of the application

- **EnableCacheVisualization** - whether to use a non-production analysis visualization mode, which shows areas of a page that are being GPU accelerated with a colored overlay
  - For development only

- **EnableFramerateCounter** - whether to display the current frame rate in the hosting browser’s status bar

- **EnableRedrawRegions** - whether to visually indicate the areas of the plug-in that are being redrawn with each frame
  - For development only

- **MaxFrameRate** - the maximum number of frames that Silverlight can render per second
Events of the Silverlight Plug-in

- **OnSourceDownloadProgressChanged** – when a piece of the XAP file has been downloaded
- **OnSourceDownloadComplete** – when the entire XAP file has been downloaded
- **OnLoad** – when the markup in the XAP file has been processed
- **OnResize** – when the size of a Silverlight content region has changed
- **OnError** – when a unhandled error occurs in the Silverlight plug-in (e.g. an exception in the application's code)
- **OnFullScreenChanged** – when the FullScreen property of the Silverlight plug-in changes
- **OnZoom** – when the Silverlight plug-in content area receives a host-generated zoom event
Read-Only Properties of the Plug-in

- **ActualWidth, ActualHeight** – size of the rendering area of the Silverlight plug-in
- **GetSystemGlyphTypefaces** - returns a collection of typefaces that are installed on the user’s computer
  - For JavaScript only
- **IsLoaded** - gets a value that indicates whether the Silverlight plug-in is loaded and has loaded all initial content
Custom Initialization Parameters

```javascript
<script type="text/javascript">
Silverlight.createObjectEx({
  source: "WebPageIntegration.xap",
  parentElement: document.getElementById("sl4Host"),
  id: "sl4",
  properties:
  {
    width: "200",
    height: "50",
    background: "white",
    alt: "<!--not installed-->
  },
  events:
  {
    onError: onSLError,
    onLoad: onSLLoad
  },
  initParams: "myParam=something",
  context: "row4"
});

private void Application_Startup(
  object sender, StartupEventArgs e)
{
  const string paramKey = "myParam";
  string paramValue = "";
  if (e.InitParams.ContainsKey(paramKey)) {
    paramValue = e.InitParams[paramKey];
  }
  this.RootVisual = new MainPage(paramValue);
}
</script>
```
Features
Controls Included in Runtime

- Button, HyperlinkButton, RepeatButton
- CheckBox, RadioButton, ComboBox, ListBox, Slider
- TextBlock, ProgressBar
- TextBox, PasswordBox
- Image, MultiScaleImage, MediaElement,
- InkPresenter
- Border, Canvas, ContentControl, Grid, StackPanel, VirtualizingStackPanel
- ScrollBar, ScrollViewer
- ToolTip
- OpenFileDialog, SaveFileDialog
- ChildWindow
Controls Included in Silverlight SDK

- Calendar, DatePicker
- AutoCompleteBox
- DataGrid, DataPager
- TreeView
- GridSplitter
- TabControl
- DescriptionViewer, Label
- ValidationSummary
- Frame, Page
- Popup
- DockPanel, WrapPanel
Mouse Support

- Silverlight provides the following set of mouse events:
  - MouseMove, MouseEnter, MouseLeave, MouseLeftButtonDown, MouseLeftButtonUp
- Silverlight supports the concept of a routed event for certain events, such as some of the mouse events
  - The bubbling routing strategy used in Silverlight means that an event originates from a child object and is then routed up to successive parent objects in the object tree
  - MouseMove, MouseLeftButtonDown, and MouseLeftButtonUp events bubble up to parent objects
- The CaptureMouse and ReleaseMouseCapture methods to enable and disable mouse capture
  - When an object has captured the mouse, it receives mouse input regardless of the mouse position
HTML Bridge

In Silverlight, the HTML Bridge is an integrated set of types and methods that enable to do the following:

- Expose complete managed types to JavaScript for scripting
- Expose individual methods of managed types to JavaScript for scripting
- Pass managed types as parameters to JavaScript functions and objects
- Return managed types from JavaScript
- Assign managed types as event handlers, which are callable from JavaScript
- Call JavaScript event handlers from managed types
- Control various security aspects of your Silverlight-based application
Calling JavaScript Code From Silverlight

```html
<script type="text/javascript">
    function MyJSFunction(param) {
        alert(param);
    }
</script>

private void SendButton_Click(object sender, RoutedEventArgs e) {
    HtmlPage.Window.Invoke("MyJSFunction", SendTextBox.Text);
}
```
Calling Silverlight Code From JavaScript

```csharp
public partial class MainPage : UserControl {
    public MainPage()
    {
        InitializeComponent();
        HtmlPage.RegisterScriptableObject("MainPage", this);
    }

    [ScriptableMember]
    public void SetText(string text)
    {
        MainTextBlock.Text = text;
    }
}
```

```html
<script type="text/javascript">
    function MyJSFunction(param) {
        var slControl = document.getElementById("sl1");
        slControl.Content.MainPage.SetText(param);
    }
</script>
```
Splash Screen

- The splash screen occupies the Silverlight client area while the package download completes in the background.
- The splash screen itself can only use the JavaScript API.
- The Silverlight splash screen model relies on three properties that are exposed by the Silverlight plug-in:
  - `splashscreensource` - the URI of a XAML page that displays while the primary package (source) is being downloaded.
  - `onsourcedownloadprogresschanged` - a JavaScript event handler that will be invoked continuously while the source is being downloaded.
  - `onsourcedownloadcomplete` - a JavaScript event handler that will be invoked once, when the source download is complete.
Custom Splash Screen

- Create a XAML file, do not add it to the .xap file, put it in place accessible from the hosting HTML page

```xml
// CustomSplash.xaml

<Canvas xmlns="http://schemas.microsoft.com/client/2007"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <Rectangle x:Name="Border"
                StrokeThickness="1" Stroke="LightGray"
                Height="20" Width="200" />
    <Rectangle x:Name="Bar"
                Fill="LightGray"
                Height="20" Width="0" />
    <TextBlock x:Name="Text"
               Canvas.Left="0" Canvas.Top="2" Width="200"
               TextAlignment="Center" />
</Canvas>
```
Custom Splash Screen cont.

- Use the **id**, **onSourceDownloadProgressChanged**, and **splashscreensource** parameters of the object tag

```
<-- TestHTMLPage.html -->

<object width="100%" height="80%" id="theApp"
    data="data:application/x-silverlight-2,"
    type="application/x-silverlight-2">
    <param name="source" value="CustomSplashScreen.xap" />
    <param name="splashscreensource" value="CustomSplash.xaml" />
    <param name="onSourceDownloadProgressChanged" value="onSourceDownloadProgressChanged" />
</object>
```
Custom Splash Screen cont.

- Write the JavaScript function for the `onSourceDownloadProgressChanged` event

```html
<!-- TestHTMLPage.html -->

<script language=javascript>
    function onSourceDownloadProgressChanged(sender, eventArgs)
    {
        var myHost = document.getElementById("theApp");
        var bar = myHost.content.findName("Bar");
        var border = myHost.content.findName("Border");
        var text = myHost.content.findName("Text");
        var perc = 0;
        if (eventArgs.progress) {
            perc = eventArgs.progress;
        } else {
            perc = eventArgs.get_progress();
        }
        bar.Width = perc * border.Width;
        text.Text = parseInt(perc * 100) + "%";
    }
</script>
```
Full-Screen

- The Silverlight plug-in can display in either embedded mode or in full-screen mode:
  - In embedded mode, the plug-in displays within the Web browser
  - In full-screen mode, the plug-in resizes to the current resolution of the screen and displays on top of all other applications, including the browser
- Silverlight application returns to embedded mode when the user presses Esc or the plug-in loses focus
- The **Content.FullScreenChanged** event occurs whenever the **IsFullScreen** property changes
Full-Screen Limitations

- A Silverlight plug-in can enable full-screen mode only in response to a user-initiated action.
- The Silverlight plug-in does not support `OpenFileDialog` and `SaveFileDialog` in full-screen mode.
  - In most cases, displaying one of these dialog boxes in full-screen mode will cause the plug-in to revert to embedded mode.
  - To avoid issues on some browsers, exit full-screen mode before using these classes.
- In full-screen mode, the only input allowed is through the following keys: spacebar, enter, tab, arrows, page up, page down, home, end.
- Multitouch input is not supported in full-screen mode.
Full-Screen Example

```csharp
public partial class MainPage : UserControl {
    private readonly Content hostContent;
    public MainPage() {
        InitializeComponent();
        hostContent = Application.Current.Host.Content;
        hostContent.Resized += HostContent_SizeChanged;
        hostContent.FullScreenChanged += HostContent_SizeChanged;
        DisplayInformation();
    }
    void HostContent_SizeChanged(object sender, System.EventArgs e) {
        DisplayInformation();
    }
    private void DisplayInformation() {
        HeightTextBlock.Text = hostContent.ActualHeight.ToString();
        WidthTextBlock.Text = hostContent.ActualWidth.ToString();
        IsFullTextBlock.Text = hostContent.IsFullScreen.ToString();
    }
    private void ToggleFullScreenButton_Click(object sender, RoutedEventArgs e) {
        hostContent.IsFullScreen = !hostContent.IsFullScreen;
    }
}
```
Out-of-Browser Applications

- Silverlight-based applications can be installed from their host Web pages and run outside the browser
  - Configuration is a simple matter of providing additional information about an application
  - It is possible to enable out-of-browser support in an existing application without rebuilding
- When an application is properly configured, Silverlight displays an install option on its right-click menu
  - Custom UI can be provided for installation to supplement or replace the right-click option
- An out-of-browser application can run without a network connection
Out-of-Browser Configuration

<OutOfBrowserSettings ShortName="MyApp"
            EnableGPUAcceleration="False"
            ShowInstallMenuItem="True">

<OutOfBrowserSettings.Blurb>
    Example of out-of-browser support.
</OutOfBrowserSettings.Blurb>
<OutOfBrowserSettings.WindowSettings>
    <WindowSettings Title="My Great Application"
                     Height="200"
                     Width="200" />
</OutOfBrowserSettings.WindowSettings>
<OutOfBrowserSettings.Icons>
    <Icon Size="16,16">
        images/16.png</Icon>
    <Icon Size="32,32">
        images/32.png</Icon>
    <Icon Size="48,48">
        images/48.png</Icon>
    <Icon Size="128,128">
        images/128.png</Icon>
</OutOfBrowserSettings.Icons>
Out-of-Browser Installation

```csharp
Application.Current.Install();
```
Out-of-Browser Update

```csharp
public MainPage() {
    InitializeComponent();
    Application.Current.CheckAndDownloadUpdateCompleted +=
        CheckAndDownloadUpdateCompleted;
}

void CheckAndDownloadUpdateCompleted(object sender,
    CheckAndDownloadUpdateCompletedEventArgs e) {
    string info = "";
    if (e.UpdateAvailable) {
        info = "A new version has been installed, restart the app."
    } else if (e.Error == null) {
        info = "There is no new version";
    } else if (e.Error is PlatformNotSupportedException) {
        info = "A new version of Silverlight is required"
    } else {
        info = string.Format("Error: {0}" , e.Error.ToString());
    }
    MessageBox.Show(info);
}

private void CheckUpdateButton_Click(object s, RoutedEventArgs e) {
    Application.Current.CheckAndDownloadUpdateAsync();
}
```
Isolated Storage

- Isolated storage enables applications to create and maintain a safe client-side virtual file system for partial trust applications
  - It uses a virtual file system that can be just one file in the root directory or a tree of directories and files

- In Silverlight, all I/O operations are restricted to isolated storage and do not use the file system of the operating system

- The isolated storage data compartment consists of one or more isolated storage files, called stores
  - Stores usually reside on the client but an application can also use isolated storage on a server
  - Silverlight does not support roaming profiles that enable information to travel with a user
Isolated Storage – Quota Groups

- A quota group is the allocated space that the host domain can use on a particular client’s computer
  - An application on a domain shares the quota group with all the other applications on that domain
- A quota group's default size is 1 MB
  - An application can request to increase the quota group size with the `IsolatedStorageFile.IncreaseQuotaTo` method
  - If the user approves the increased size, that size is the new quota group size that all applications on that domain can share

```csharp
const int requiredFreeSpace = 10 * 1024 * 1024;
using (var store = IsolatedStorageFile.GetUserStoreForApplication()) {
    if (store.AvailableFreeSpace < requiredFreeSpace) {
        store.IncreaseQuotaTo(store.Quota + requiredFreeSpace - store.AvailableFreeSpace);
    }
}
```
Isolated Storage & OpenFileDialog

```csharp
var openFileDialog = new OpenFileDialog();
if (openFileDialog.ShowDialog() == true)
{
    FileInfo fileInfo = openFileDialog.File;
    using (IsolatedStorageFile store =
        IsolatedStorageFile.GetUserStoreForApplication())
    {
        Stream loadStream = fileInfo.OpenRead();

        string filePath = Path.Combine(dirPath, fileInfo.Name);
        IsolatedStorageFileStream saveStream =
            store.CreateFile(filePath);

        // copy data from the loadStream to the saveStream

        saveStream.Close();
        loadStream.Close();
    }
    ListFiles(dirPath);
}
```
Application Navigation

- The application navigation in a Silverlight application can be implemented by using the **Frame** and **Page** controls.
  - Page controls represent discrete sections of content.
  - The frame acts as a container for page controls, and facilitates navigation to pages.
- Within a frame, it can be specified that a certain URI pattern maps to a particular page.
- Application navigation can be integrated with the browser navigation by setting the **JournalOwnership** property of the frame.
  - The default value is 'Automatic' – it means that the frame will integrate with the browser journal if it is a top-level frame.
  - The Web page that contains the Silverlight object must include an iframe named `_sl_historyFrame`.
Application Navigation Example

```xml
<navigation:Frame x:Name="ContentFrame" Source="/Home"
    Navigated="ContentFrame_Navigated"
    NavigationFailed="ContentFrame_NavigationFailed">
    <navigation:Frame.UriMapper>
        <uriMapper:UriMapper>
            <uriMapper:UriMapping Uri="" MappedUri="/Views/Home.xaml"/>
            <uriMapper:UriMapping Uri="/{{pageName}}" MappedUri="/Views/{{pageName}}.xaml"/>
        </uriMapper:UriMapper>
    </navigation:Frame.UriMapper>
</navigation:Frame>
```
Resources

Silverlight supports the following kinds of resources:

- XAML resources, such as styles and templates that can be shared among multiple user interface elements
- Resource files, such as images and videos that can be referred to by URI
  - Resource files can be embedded in assemblies, included separately in the application package, or retrieved them from the network
- Resource strings and other values embedded in the assemblies or provided through localized satellite assemblies

The `AssemblyPart` class can be used to load assemblies

Using the `ResourceManager` class, localized resources can be loaded and used
Localization

- Globally aware applications
  - As for other .NET solutions, Silverlight applications use two properties:
    - `CultureInfo.CurrentCulture` – for formatting dates and numbers, text comparisons, etc.
    - `CultureInfo.CurrentUICulture` – for UI resources
  - Both these properties are read-only in Silverlight, to change the current culture, modify the `Thread.CurrentCulture` property

- Localized applications
  - Localizing an application involves creating a separate set of resources
  - Visual Studio can be used to create a localized Silverlight-based application that contains a designated subset of the application's satellite assemblies
    - in the project file add a `SupportedCultures` property to the `PropertyGroup` section
Media

The following media formats are supported by Silverlight:

- Video: WMV, MP4
- Audio: WMA, MP3
- Server-side playlist (SSPL)
- Client side playlist (ASX)

In addition to progressive downloads, **MediaElement** supports live and on-demand streaming from a Windows Media Server (using the **mms** protocol)

Integrating Digital Rights Management (DRM) into the Silverlight applications allows to protect and securely deliver streaming or progressive download content for cross-platform playback
Networking and Communication

- Silverlight runtime has built-in support for network communication using HTTP/HTTPS and TCP
  - Communication can be done also at sockets level (using the `System.Net.Sockets`)
- For security reasons, by default, Silverlight supports calls to Web services on the same domain or site of origin
  - Same domain means that calls must use the same sub domain, protocol, and port
- When a Silverlight-based application detects that its request is a cross-domain request, it will first look for the Silverlight cross-domain policy file (`clientaccesspolicy.xml`) at the application root of the Web service
  - If there is an error, the application will look for the Flash cross-domain policy file (`crossdomain.xml`) at the same place
Making HTTP Calls

- HTTP calls can be done by using a client-side proxy class or by constructing the calls manually
  - The proxy class for an XML Web Service or WCF service can be automatically generated
  - Silverlight uses WCF capabilities to create proxies and send SOAP 1.1 messages over HTTP

- For manual calling, the following classes can be used:
  - **WebClient** – provides a simple event-based model that enables you to download and upload streams and strings
  - **HttpWebRequest, HttpWebResponse** – support more complex communication scenarios
public App() {
    this.UnhandledException += this.Application_UnhandledException;
}
private void Application_UnhandledException(object sender,
                                             ApplicationUnhandledExceptionEventArgs e) {
    if (!System.Diagnostics.Debugger.IsAttached) {
        e.Handled = true;
            () => ReportErrorToDOM(e));
    }
}
private void ReportErrorToDOM(ApplicationUnhandledExceptionEventArgs e) {
    try {
        string msg = e.ExceptionObject.Message +
                     e.ExceptionObject.StackTrace;
        msg = errorMsg.Replace('"', '"').Replace("\n", @"\n");
        Error("Unhandled Error in Silverlight Application " +
            msg + ");");
    }
    catch (Exception) {}
Application Exceptions cont.

- If the **Handled** property of the event data is not set to true, then the exception might fall through to the unmanaged level of error handling
  - This can potentially be processed with an **onError** handler written in script, executed in the browser script host
  - However, if an exception reaches this point, the application's **AppDomain** is terminated, and managed application code can no longer execute
- The **UnhandledException** technique does not capture exceptions that originate from asynchronous operations, such as download requests that fail
  - These exceptions are typically handled by dedicated handlers on the object that invokes the original asynchronous operation
The OnError Parameter

The OnError handler can be used for the following errors:

- XAML parsing errors
- Errors that occur during load
- Run-time errors
- Errors in synchronous method calls that are not handled in a try/catch block
- Asynchronous error events that do not have an attached event listener

The Silverlight.js file provides a default event handler for the OnError handler parameter
Debugging Silverlight Applications

- With the Silverlight Tools installed, the Silverlight projects can be debugged as any other project type.
- It is possible to attach with a debugger to a running Silverlight application.
  - Use the 'Attach to Process' option in Visual Studio and choose the process with the 'Silverlight' type.
- It is also possible to debug remotely applications run on a Macintosh computer.
Debugging JS in a Silverlight Project

To debug JavaScript using Internet Explorer, set the following options in the Advanced tab in Internet Options group of the Tools menu in Internet Explorer:

- Clear 'Disable Script Debugging (Internet Explorer)'
- Clear 'Disable Script Debugging (Other)'
- Select 'Display a notification about every script error'
- Clear 'Show friendly HTTP error messages'

The debugger can be attached to a process of 'Script' type.
Deep Zoom

- Deep Zoom provides the ability to zoom almost arbitrarily large images in Silverlight in a performant manner
  - The Deep Zoom Composer application can be used to create an image pyramid

- To use Deep Zoom in application:
  - Create an image pyramid
  - Add a Deep Zoom object like `MultiScaleImage` or `MultiScaleSubImage` to the application
  - Hook up events to add interactivity (zoom and panning) to the Deep Zoom object(s).
Deployment

- Deploying a Silverlight-based application is often just a matter of uploading an application package file and a Web page to the Web server.

- Silverlight applications can also use other resources deployed separately on the server and retrieved only when they are needed.
  - The application can refer to these files the same way it would if they were within the application package.
  - Also library assemblies can be stored on the server and retrieved when they are needed.
ADO.NET Data Services

Applications for Silverlight can read and modify data deployed by ADO.NET Data Services

- The ADO.NET Data Services Framework defines data as entities and relationships according to the specifications of the Entity Data Model (EDM) and deploys data in the style of representational state transfer (REST) resources

- Applications for Silverlight can access the data through standard HTTP protocol

- LINQ queries can be used from the application for Silverlight to communicate with data services, execute queries, and update data on the server

- ADO.NET Data Services use data transport formats, currently ATOM and JSON, that are text based and can be used across a firewall
Performance
Performance Tips

- Test on multiple platforms and browsers
  - The differences in platform or browser behavior, and the code Silverlight uses to handle the behavior, can affect application performance

- Set `EnableFrameRateCounter` to true during development
  - This setting displays the frames-per-second (fps) of the rendered Silverlight content in the browser's status bar

- Use transparent background for a Silverlight plug-in sparingly
  - Using a transparent background for a Silverlight plug-in can be very useful but it has a significant effect on performance
Performance Tips cont.

- When animating the **Opacity** or **Transform** of a **UIElement**, set its **CacheMode**
- While animating text, set **TextRenderingMode** to **RenderForAnimation**
  - By default, Silverlight optimizes text for readability
  - Changing size or rotation results in a recalculating how the glyph is most legible at that font size and position
  - While animating the rotation or scale of text, set the **TextHintingMode** to **Animated** and set it back to **Fixed** when the animation ends
- Cache visual elements when blending layers using **Opacity** and rotating or stretching objects
  - Set the **CacheMode** property on a **UIElement**
Performance Tips cont.

- Avoid using windowless mode
  - Performance is seriously impacted when in windowless mode; media playback is not recommended at all in windowless mode
- Use **Visibility** instead of **Opacity** whenever possible
  - **Opacity** is associated with higher costs because the object is still hit tested and technically rendered
- In full-screen mode, hide unused objects
- Do not use **Width** and **Height** with **MediaElement** objects
  - Let the media element display at its natural size (re-encode the media to the desired size if necessary)
- Do not use **Width** and **Height** with **Path** objects
  - Setting these properties will result in additional stretching, which affects performance
Performance Tips cont.

- Break up CPU-intensive work into smaller tasks
  - When procedural code is running (for example, C# or Visual Basic), the plug-in stops drawing

- Break up large application packages
  - In some cases, the Silverlight plug-in fails to load very large application package (.xap) files

- Use `Double.ToString(CultureInfo.InvariantCulture)` rather than `Double.ToString()`
  - The version for `InvariantCulture` is optimized for performance

- Use `Stretch="Fill"` when rendering a lot of images
  - With this setting, there is no potential clipping happening which adds extra edges
Hardware Acceleration

- Caching visual elements as bitmaps allows to take advantage of hardware acceleration
  - Once an object or tree of objects has been cached as a bitmap, it no longer goes through the render phase as the application refreshes, rather, the cached bitmap is rendered

- Hardware acceleration can benefit performance for the following scenarios:
  - Blending two static layers using opacity
  - Transforming objects (e.g. stretching and rotating)

```xml
<param name="EnableGPUAcceleration" value="true"/>
<StackPanel CacheMode="BitmapCache" ... />
```
Hardware Acceleration cont.

- Hardware acceleration is only enabled on Windows Vista, Windows 7, and Windows XP
  - On Windows XP, NVidia, ATI, Intel cards with a driver date post November 2004 is required for hardware acceleration