* 1. 

Hands-On Lab

Building Applications in Silverlight 4

Module 5 Migrating Existing Applications to Out-of-Browser

**Contents**

[Lab 5 – Building an Out-of-Browser Application 3](#_Toc276644024)

[Exercise 1: Enabling Out of Browser 6](#_Toc276644025)

[Exercise 2: Detect if running out of Browser 8](#_Toc276644026)

[Exercise 3: Customizing the Window and enabling Elevated Privileges 15](#_Toc276644027)

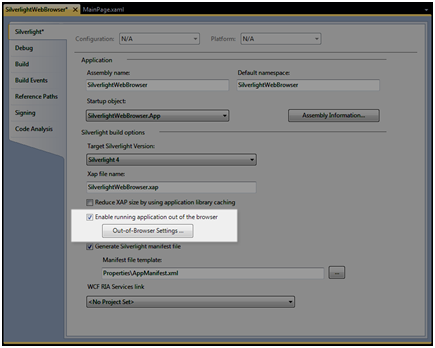
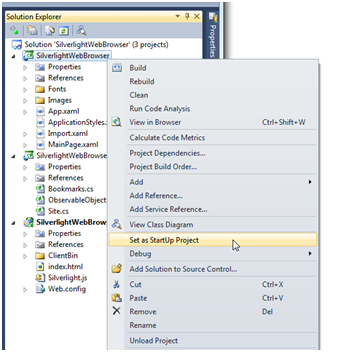
[Exercise 4: Export Data to Excel, Word, and more 23](#_Toc276644028)

[Summary 32](#_Toc276644029)

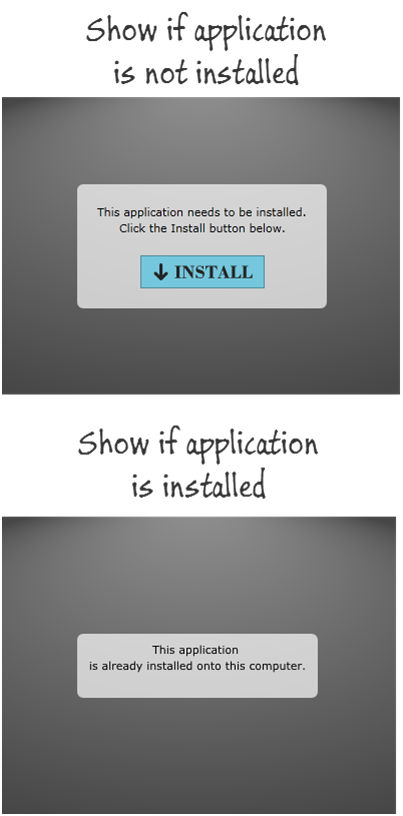
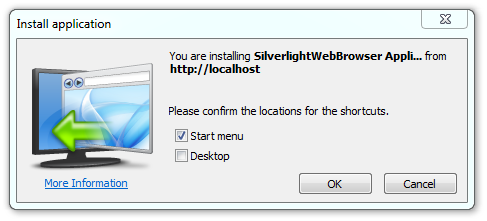
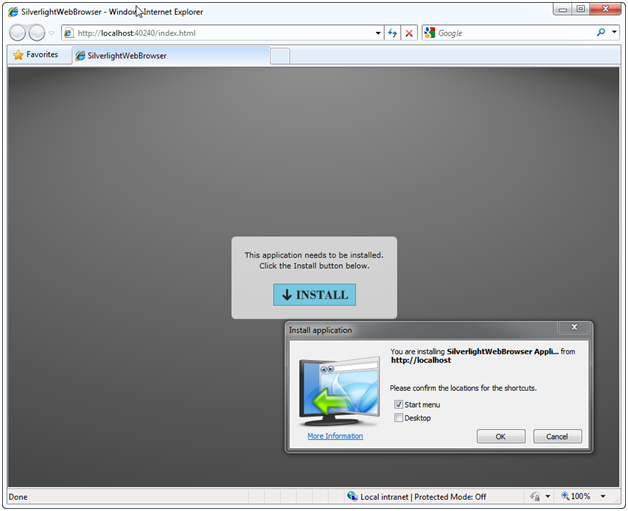
Lab 5 – Building an Out-of-Browser Application

* 1. Silverlight started off as a browser plug-in. Since then, Silverlight has emerged as a client platform technology that not only includes browser based applications, but is now a desktop platform as well. This lab is designed to guide Winforms or desktop developers through the process of developing a desktop application in Silverlight.
  2. In the lab you'll create a Silverlight web browser. It will truly be a desktop application. Along the way you'll learn how to enable Silverlight out-of-browser, detect the context the application is running in, customize the window, and how to interact with Microsoft Excel.
  3. You'll start by using an existing application and enabling the out-of-browser feature. Next, you’ll determine if the application is running on the desktop or in the browser. You’ll then customize the look of the application. Finally you’ll learn how to interact with Microsoft Excel. The Silverlight application that you'll create is shown next:
     1. 
  4. **You will benefit from this lab if:**
  + You are coming from a Winforms background and are familiar with developing for the desktop and not the web.
  + You’re looking to develop Desktop applications with Silverlight
  + You’re interested in knowing the capabilities of Silverlight Out-of-Browser.
  1. **You will learn:**
  + How to enable Out-of-Browser.
  + How to create custom windows.
  + How to manipulate the application window from code.
  + The possibilities of Elevated Privileges.
  + How to call external applications with the AutomationFactory.
  1. **Business Requirements for the Silverlight application include:**
  + Enabling Out of Browser
  + Display the correct interface based on the context the application is running under
  + Prompt the user to install the application from the web to their desktop
  + Enable elevated privileges
  + Customize the look of your application
  + Enable the exporting of data to Office products
  1. **Estimated Time 45 Minutes**

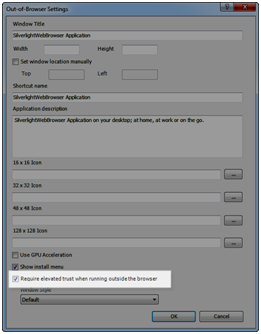
Exercise 1: Enabling Out of Browser

* + 1. The out-of-browser feature in Silverlight opens a new world of possibilities. Silverlight 4 extends out-of-browser functionality by enabling trusted applications. This makes it closer to programming for WinForms. This lab focuses on core out-of-browser functionality. To get started you must first enable your Silverlight project to run in an out-of-browser window, which this exercise outlines.
  1. Open the solution for this lab, and run the project. The application will run in a browser, and you’ll notice WebBrowser control saying “HTML is enabled only in Out-of-Browser mode”.
  2. To enable out of browser, stop debugging, and open the Properties page in Visual Studio. Here click the “Enable running application out of the browser”.
     1. 
     2. Figure 1
     3. Enable running application out of the browser
  3. Before running the project, right click the Silverlight project and select Set as Startup Project.
     1. 
     2. Figure 2
     3. Set as Startup Project
  4. Run the project. Your application is now running in Out-of-Browser mode. If you’re coming from WinForms you should feel at home now…no browser!
     1. **Note:** Notice you can see HTML, Flash, and Silverlight running inside the app!

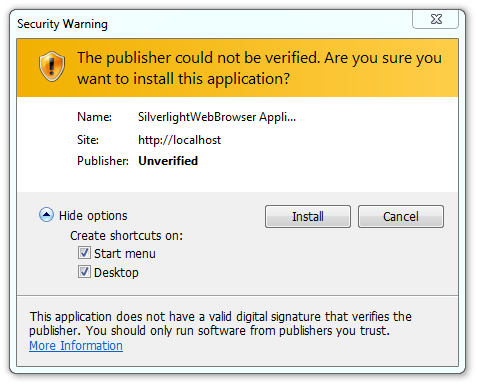
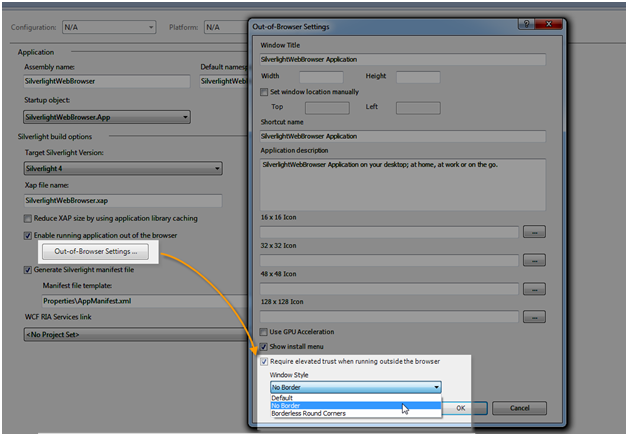
Exercise 2: Detect if running out of Browser

* + 1. Since there are many features that aren’t available in browser, such as the WebBrowser control, you will need to be able to detect which context, desktop or browser, the application is currently running under. This exercise shows how to detect if your application is installed, displays a custom prompt asking the user to install you app, and finally how to install from a Button.
  1. Locate and open the InstallPrompt.xaml file. This UserControl will load if the application is run in the Browser. Before determining the context of where the browser is being run, we are going to toggle between two states based on if the application is already installed. The below screenshot shows the two states:
     1. 
     2. Figure 1
     3. Screenshot Example
  2. In InstallPrompt.xaml.cs we are going to toggle the Visibility of both the InstallPanel and AlreadyInstalledPanel, previously defined in the XAML. The property Application.Current.InstallState contains what state the application is in. If it’s installed, the value is InstallState.Installed. Other possible values could be InstallState.NotInstalled, InstallState.Installing, or InstallState.Failed.
  3. In the constructor register the loaded event. In the Loaded event handler check to see if the InstallState is equal to InstallState.Installed. If true, then display the message indicating the application has already been installed, otherwise show the install button
     1. C#
     2. public partial class InstallPrompt : UserControl
     3. {
     4. public InstallPrompt()
     5. {
     6. InitializeComponent();
     7. this.Loaded += new RoutedEventHandler(InstallPrompt\_Loaded);
     8. }
     9. void InstallPrompt\_Loaded(object sender, RoutedEventArgs e)
     10. {
     11. if (Application.Current.InstallState == InstallState.Installed)
     12. {
     13. InstallPanel.Visibility = Visibility.Collapsed;
     14. AlreadyInstalledPanel.Visibility = Visibility.Visible;
     15. }
     16. else
     17. {
     18. InstallPanel.Visibility = Visibility.Visible;
     19. AlreadyInstalledPanel.Visibility = Visibility.Collapsed;
     20. }
     21. }
     22. }
     23. Visual Basic
     24. Partial Public Class InstallPrompt
     25. Inherits UserControl
     26. Public Sub New()
     27. InitializeComponent()
     28. AddHandler Loaded, AddressOf InstallPrompt\_Loaded
     29. AddHandler InstallButton.Click, AddressOf InstallButton\_Click
     30. End Sub
     31. Private Sub InstallPrompt\_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
     32. If Application.Current.InstallState = InstallState.Installed Then
     33. InstallPanel.Visibility = Visibility.Collapsed
     34. AlreadyInstalledPanel.Visibility = Visibility.Visible
     35. Else
     36. InstallPanel.Visibility = Visibility.Visible
     37. AlreadyInstalledPanel.Visibility = Visibility.Collapsed
     38. End If
     39. End Sub
     40. End Class
  4. In the InstallPrompt.xaml.cs, add an event handler for the InstallButton’s Click event. Here we’ll call Application.Current.Install() which brings up the installation prompt
     1. C#
     2. private void InstallButton\_Click(object sender, RoutedEventArgs e)
     3. {
     4. Application.Current.Install();
     5. }
     6. Visual Basic
     7. Private Sub InstallButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
     8. Application.Current.Install()
     9. End Sub
     10. 
     11. Figure 2
     12. Add event for install Application
     13. Now that we have the InstallPrompt.xaml.cs or InstallPrompt.xaml.vb checking whether or not to prompt the user to install the application and installing if not currently installed, let’s turn our attention to detecting what context the application is running in. What we want is if the application is in a web browser we want to show the InstallPrompt screen, otherwise we should show the MainPage. The way we’re going to do it is by changing the RootVisual based on if the Application is running out of the Browser.
  5. Open App.xaml.cs. Find the Startup event and replace the current method with the below code. Here we’re checking to see if the Application.Current.IsRunningOutOfBrowser is true. If it’s not then we set the RootVisual to the InstallPrompt.
     1. C#
     2. private void Application\_Startup(object sender, StartupEventArgs e)
     3. {
     4. if (Application.Current.IsRunningOutOfBrowser)
     5. {
     6. this.RootVisual = new MainPage();
     7. }
     8. else
     9. {
     10. this.RootVisual = new InstallPrompt();
     11. }
     12. }
     13. Visual Basic
     14. Private Sub Application\_Startup(ByVal sender As Object, ByVal e As StartupEventArgs)
     15. If Application.Current.IsRunningOutOfBrowser Then
     16. Me.RootVisual = New MainPage()
     17. Else
     18. Me.RootVisual = New InstallPrompt()
     19. End If
     20. End Sub
  6. We’re all set. Set the Web project to the Startup project and run the project. You should see the InstallPrompt asking you to install.
     1. 
     2. **Figure 3**
     3. Install appliction
  7. After clicking OK, you application will open in a new Window. Notice we see the MainPage and not the InstallPrompt? This is due to the logic we added in the Startup event of App.xaml.cs or App.xaml.vb.
     1. 
     2. Figure 4
     3. MainPage example

Exercise 3: Customizing the Window and enabling Elevated Privileges

* + 1. Silverlight out of Browser enables you to ask the user to install your application with Elevated Privileges (EP). By running Silverlight in an EP mode you have a number of power pieces of functionality at your fingertips: Clipboard access, relaxed cross domain, direct access to the user folder, COM interop, and custom chrome. This exercise covers how to enable Elevated Privileges and then how to create a custom Chrome (a feature that’s crucial for kiosks or brand driving applications).
  1. To enable Elevated Privileges, bring up the Out-of-Browser Settings dialog (Right click the Silverlight project, click Properties, find the Out-of-Browser Settings button towards the bottom.)
  2. Check the Check box labeled “Require elevated trust when running outside the browser”.
     1. 
     2. Figure 1

Enable Check box

* 1. Click OK and run the project. If you get to the Install Prompt and it indicates you already have the application installed, Right click, select Remove this application, then refreshes the page. Click the Install button when prompted. Notice the Install prompt looks different than the previous step. This is asking the user for Elevated Privileges.
     1. 
     2. Figure 2
     3. New Install window
  2. Now that Elevated Privileges are enabled, we can have custom chrome on our window. Go back to the Out-of-Browser Settings dialog box and choose No Border from the drop down list labeled Window Style.
     1. 
     2. Figure 3
     3. Out-of-Browser Settings
  3. Run the project to show the chrome is gone. This is the desired effect; however we don’t have a way to close the window. For now, go back to Visual Studio and stop the debugger.
  4. Open up MainPage.xaml and uncomment lines 210-231. This is the WindowControlsPanel that contains buttons that control our window: always on top, minimize, maximize, and close. Below is the XAML uncommented followed by what you should see in Visual Studio.
     1. XAML
     2. <StackPanel x:Name="WindowControlsPanel" Grid.Column="1" HorizontalAlignment="Right" Margin="0,2,4,0" Orientation="Horizontal" VerticalAlignment="Top">
     3. <Button x:Name="OnTopButton" Margin="0,0,1,0" Style="{StaticResource
     4. TopButtonStyle}" Width="16" Height="16" ToolTipService.ToolTip="Pin to the topmost window">
     5. <Path Data="F1 M477.23441,395.96191 L473.29312,399.9386 L463.59912,
     6. 392.70959 L466.6774,389.59689 C467.56339,388.7099467.67639,387.38589
     7. 466.93039,386.63989C466.18439,385.89389464.86139,386.0069463.9744,
     8. 386.89389 L450.7294,400.13989C449.84241,401.02591 449.7294,402.34891
     9. 450.4754,403.09491C451.22141,403.84091 452.54639,403.72791453.4314,
     10. 402.8429L455.93942,400.3111 L463.16742,410.00711 L459.79639,413.3999
     11. C458.9104,414.28589 458.79639,415.60791 459.5434,416.35489C460.29041,
     12. 417.10089 461.6134,416.98691 462.49939,416.10089L469.23541,409.36591
     13. L480.4234,420.55591 C481.17041,421.2999482.49341,421.18689483.3804,
     14. 420.30191 C484.26541,419.41489484.37939,418.09091 483.63339,417.34491
     15. L472.4444,406.15689L479.9364,398.66489 C480.82339,397.77789 480.9364,
     16. 396.45389480.18939,395.70789 C479.4444,394.96191 478.1214,395.07489
     17. 477.23441,395.96191"Fill="White"Stretch="Fill"UseLayoutRounding="False"
     18. d:LayoutOverrides="GridBox" RenderTransformOrigin="0.5,0.5"
     19. Margin="1,0,1,2">
     20. <Path.RenderTransform>
     21. <CompositeTransform Rotation="45" TranslateY="1.8068817553285044E-09"/>
     22. </Path.RenderTransform>
     23. </Path>
     24. </Button>
     25. <Button x:Name="MinimizeButton" Width="16" Height="16" Margin="0,0,1,0"
     26. Style="{StaticResource TopButtonStyle}" ToolTipService.ToolTip="Minimize" Padding="0">
     27. <Path Data="F1M229.417,216.229L208.354,237.292C207.544,238.103,207.544,239.416,208.3
     28. 54,240.227C209.165,241.037,210.479,241.037,211.289,240.227L232.352,
     29. 219.165C233.162,218.354,233.162,217.04,232.352,216.229C231.541,215.419,
     30. 230.227,215.419,229.417,216.229"Fill="White" Stretch="Fill"
     31. UseLayoutRounding="False" RenderTransformOrigin="0.5,0.5" Height="6"
     32. Width="6" Margin="0,6,0,0">
     33. <Path.RenderTransform>
     34. <CompositeTransform Rotation="45"/>
     35. </Path.RenderTransform>
     36. </Path>
     37. </Button>
     38. <Button x:Name="MaximizeButton" Width="16" Height="16" Margin="0,0,1,0"
     39. Style="{StaticResource TopButtonStyle}"ToolTipService.ToolTip="Maximize">
     40. <Path Data="F1M74.313,328.7861C74.313,329.5261,73.713,330.1281,72.973,330.1281L53.03
     41. 3,330.1281C52.292,330.1281,51.692,329.5261,51.692,328.7861L51.692,
     42. 308.4751C51.692,307.7351,52.292,307.1361,53.033,307.1361L72.973,

307.1361C73.713,307.1361, 74.313,307.7351,74.313,308.4751zM74.806,

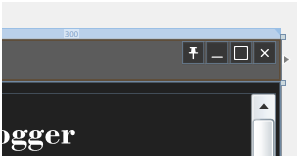
305.0221L51.199,305.0221C50.323,305.0221,49.612,305.7311,49.612,

306.6081L49.612,330.6551C49.612,331.5321,50.323,332.2411,51.199,

332.2411L74.806,332.2411C75.683,332.2411,76.393,331.5321,

76.393,330.6551L76.393,306.6081C76.393,305.7311,75.683,305.0221,

74.806,305.0221" Fill="White" Stretch="Fill"UseLayoutRounding="False"/>

* + 1. </Button>
    2. <Button x:Name="CloseButton" Width="16" Height="16" Margin="0" Style="{StaticResource TopButtonStyle}" ToolTipService.ToolTip="Close">
    3. <Path Data="M0.54881889,0 C0.68929577,0 0.82977271,0.053546324
    4. 0.93699646,0.16063902 L3.3335769,2.5573325 L5.730268,0.1606418
    5. C5.9444532,0.053543516 6.2919092,-0.053543516 6.506361,0.1606418
    6. C6.7205462,0.37508935 6.7205462,0.72254932 6.506361,0.93699688
    7. L4.1097264,3.3335183 L6.5063634,5.730268 C6.7205486,5.9444532
    8. 6.7205486,6.291913 6.5063634,6.5063605 C6.2919116,6.7205458
    9. 5.9447179,6.7205458 5.7302704,6.5063605 L3.3335588,4.1096492
    10. L0.93673187,6.5063634 C0.72254658,6.7205486 0.37508655,6.7205486
    11. 0.16063894,6.5063634 C-0.053546317,6.2919121 -0.053546317,5.9447184
    12. 0.16063894,5.7302704 L2.5574095,3.3334999 L0.16064122,0.93673182 C
    13. 0.053544026,0.72254652 –0.053544026,0.3750906 0.16064122,0.16063902
    14. C0.26786506,0.053546324 0.40834194,0 0.54881889,0 z" Fill="White"
    15. Stretch="Fill" UseLayoutRounding="False" Margin="2" />
    16. </Button>
    17. </StackPanel>
    18. 
    19. Figure 4
    20. WindowControlsPanel
  1. We now need to add functionality to each button. The four buttons are pin to top, minimize, maximize, and close. We’ll walk through each event handler, but first we need to register the Click events for each button. Do this in the constructor of MainPage.
     1. C#
     2. public MainPage()
     3. {
     4. InitializeComponent();
     5. // Register Click events for the buttons in the WindowControlsPanel
     6. this.OnTopButton.Click += new RoutedEventHandler(OnTopButton\_Click);
     7. this.MinimizeButton.Click += new RoutedEventHandler(MinimizeButton\_Click);
     8. this.MaximizeButton.Click += new RoutedEventHandler(MaximizeButton\_Click);
     9. this.CloseButton.Click += new RoutedEventHandler(CloseButton\_Click);
     10. }
     11. //Pin to Top – Setting the value of Application.Current.MainWindow.TopMost enables
     12. //you to pin your application on top of every other app.
     13. void OnTopButton\_Click(object sender, RoutedEventArgs e)
     14. {
     15. // Toggle between being on top and not
     16. Application.Current.MainWindow.TopMost = !Application.Current.MainWindow.TopMost;
     17. }
     18. //Minimize – To minimize an out of browser application set the value of
     19. //Application.Current.MainWindow.WindowState to WindowState.Minimized.
     20. void MinimizeButton\_Click(object sender, RoutedEventArgs e)
     21. {
     22. // Minimize the application
     23. Application.Current.MainWindow.WindowState = WindowState.Minimized;
     24. }
     25. //Maximize– To maximize the window, set the value of
     26. //Application.Current.MainWindow.WindowState to WindowState.Maximized. In this event
     27. //handler we’re toggling between the Normal WindowState and Maximized.
     28. void MaximizeButton\_Click(object sender, RoutedEventArgs e)
     29. {
     30. // Toggle between the Normal and Maximized state
     31. if (Application.Current.MainWindow.WindowState != WindowState.Maximized)
     32. {
     33. Application.Current.MainWindow.WindowState = WindowState.Maximized;
     34. }
     35. else
     36. {
     37. Application.Current.MainWindow.WindowState = WindowState.Normal;
     38. }
     39. }
     40. //Close – To close, simply call the Close method of Application.Current.MainWindow.
     41. void CloseButton\_Click(object sender, RoutedEventArgs e)
     42. {
     43. // Close the application
     44. Application.Current.MainWindow.Close();
     45. }
     46. Visual Basic
     47. Partial Public Class MainPage
     48. Inherits UserControl
     49. Public Sub New()
     50. InitializeComponent()
     51. ' Register Click events for the buttons in the WindowControlsPanel
     52. AddHandler OnTopButton.Click, AddressOf OnTopButton\_Click
     53. AddHandler MinimizeButton.Click, AddressOf MinimizeButton\_Click
     54. AddHandler MaximizeButton.Click, AddressOf MaximizeButton\_Click
     55. AddHandler CloseButton.Click, AddressOf CloseButton\_Click

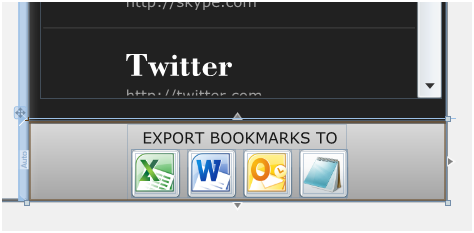
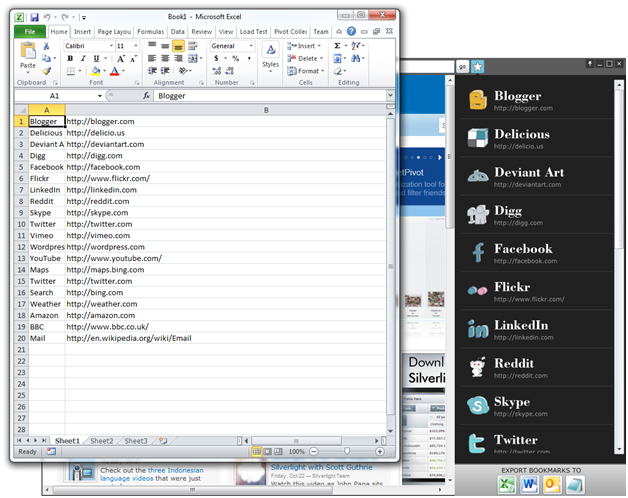
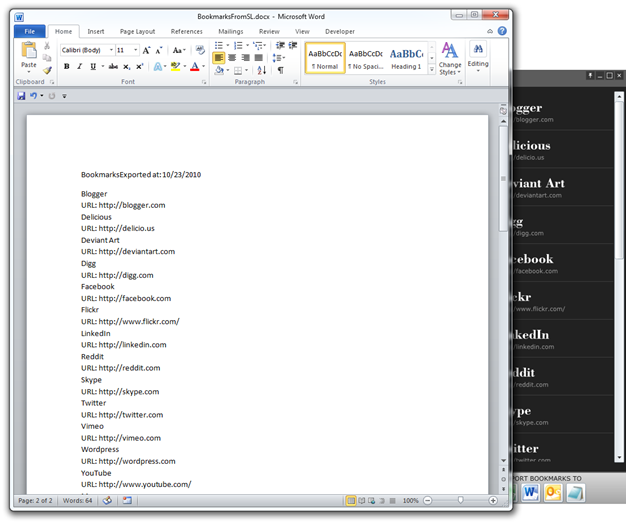
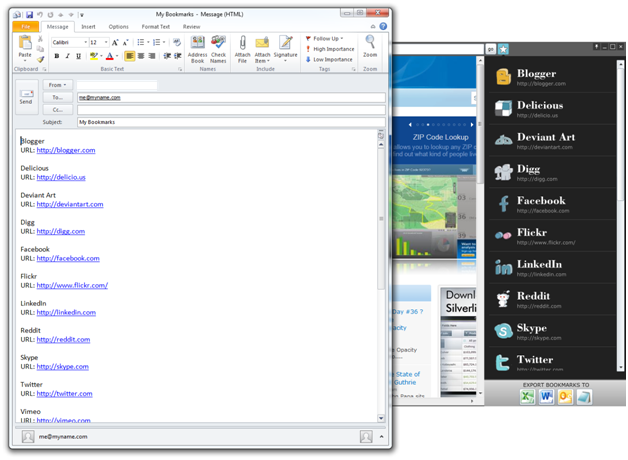
End Sub

* + 1. Private Sub OnTopButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    2. ' Toggle between being on top and not
    3. Application.Current.MainWindow.TopMost = Not Application.Current.MainWindow.TopMost
    4. End Sub
    5. Private Sub MinimizeButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    6. ' Minimize the application
    7. Application.Current.MainWindow.WindowState = WindowState.Minimized
    8. End Sub
    9. Private Sub MaximizeButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    10. ' Toggle between the Normal and Maximized state
    11. If Application.Current.MainWindow.WindowState <> WindowState.Maximized Then
    12. Application.Current.MainWindow.WindowState = WindowState.Maximized
    13. Else
    14. Application.Current.MainWindow.WindowState = WindowState.Normal
    15. End If
    16. End Sub

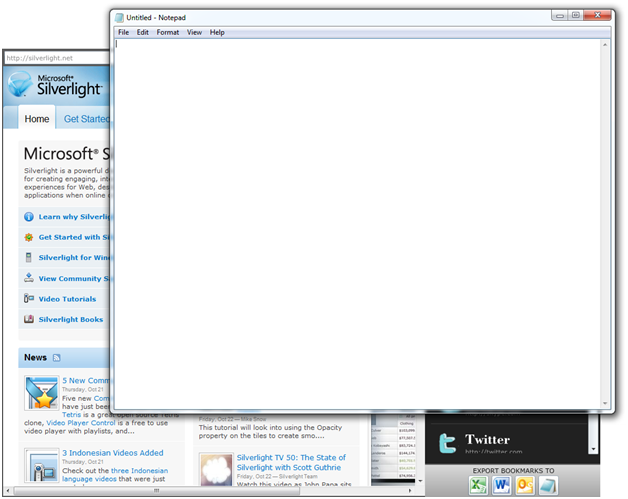
Private Sub CloseButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)' Close the application

* + 1. Application.Current.MainWindow.Close()
    2. End Sub
    3. End Class
  1. Now run you application. Make sure to run it out of the browser (either set the Silverlight project to the Startup project or by reinstalling it). You have now regained control of the Window.
     1. 
     2. Figure 5
     3. Running application

Exercise 4: Export Data to Excel, Word, and more

* + 1. As mentioned above, enabling Elevated Privileges opens up a world of possibilities. One feature is the AutomationFactory. This enables you to call COM components from Silverlight. In our application we are going to enable the user to export their Bookmarks into Excel, Word, or e-mail them with Outlook.
  1. In MainPage.xaml uncomment lines 128-152. This is the ExportPanel that contains four buttons as seen below.
     1. XAML
     2. <Border x:Name="ExportPanel" BorderThickness="0,2,0,0" BorderBrush="#FF666666" Margin="0,2,0,0" Grid.Row="1" Height="57" >
     3. <Border.Background>
     4. <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
     5. <GradientStop Color="#FFDADADA"/>
     6. <GradientStop Color="#FFA5A5A5" Offset="1"/>
     7. </LinearGradientBrush>
     8. </Border.Background>
     9. <Grid HorizontalAlignment="Center">
     10. <TextBlock TextWrapping="Wrap" Text="EXPORT BOOKMARKS TO" d:LayoutOverrides="Height" Foreground="#FF181818" HorizontalAlignment="Center" Margin="2,2,0,0" TextOptions.TextHintingMode="Animated" VerticalAlignment="Top" />
     11. <StackPanel HorizontalAlignment="Left" Orientation="Horizontal" VerticalAlignment="Bottom" Height="35" Margin="2,0,0,2">
     12. <Button x:Name="ExcelButton" Margin="0,0,5,0">
     13. <Image Source="Images/Excel.png" Stretch="Fill"/>
     14. </Button>
     15. <Button x:Name="WordButton" Margin="0,0,5,0">
     16. <Image Source="Images/Word.png" Stretch="Fill"/>
     17. </Button>
     18. <Button x:Name="OutlookButton" Margin="0,0,5,0">
     19. <Image Source="Images/Outlook.png" Stretch="Fill"/>
     20. </Button>
     21. <Button x:Name="PowerPointButton" Margin="0">
     22. <Image Source="Images/notepad.png" Stretch="Fill"/>
     23. </Button>
     24. </StackPanel>
     25. </Grid>
     26. </Border>
     27. 
     28. Figure 1
     29. Export Panel
  2. The ExportPanel contains four buttons: one for exporting to Excel, Word, creating an e-mail with Outlook, and a button to open Notepad. In MainPage.xaml.cs we’ll add Click event handlers for all four buttons. First, in the constructor, register the Click events for each button:
     1. C#
     2. // Register Click events for the Export buttons
     3. this.ExcelButton.Click += new RoutedEventHandler(ExcelButton\_Click);
     4. this.WordButton.Click += new RoutedEventHandler(WordButton\_Click);
     5. this.OutlookButton.Click += new RoutedEventHandler(OutlookButton\_Click);
     6. this.PowerPointButton.Click += new RoutedEventHandler(PowerPointButton\_Click);
     7. Visual Basic
     8. 'Register Click events for the Export buttons
     9. AddHandler ExcelButton.Click, AddressOf ExcelButton\_Click
     10. AddHandler WordButton.Click, AddressOf WordButton\_Click
     11. AddHandler OutlookButton.Click, AddressOf OutlookButton\_Click
     12. AddHandler PowerPointButton.Click, AddressOf PowerPointButton\_Click
  3. For exporting to Excel copy and paste the below event handler into MainPage.xaml.cs. Here we’re using the AutomationFactory to create an instance of Excel. Then we construct a worksheet.
     1. C#
     2. void ExcelButton\_Click(object sender, RoutedEventArgs e)
     3. {
     4. dynamic excel = AutomationFactory.CreateObject("Excel.Application");
     5. excel.Visible = true;
     6. dynamic workbook = excel.workbooks;
     7. workbook.Add();
     8. dynamic sheet = excel.ActiveSheet;
     9. dynamic cell = null;
     10. int i = 1;
     11. // Populate the excel sheet
     12. foreach (var item in (LayoutRoot.DataContext as Bookmarks).Sites)
     13. {
     14. cell = sheet.Cells[i, 1];
     15. cell.Value = item.Title;
     16. cell = sheet.Cells[i, 2];
     17. cell.Value = item.Uri;
     18. cell.ColumnWidth = 100;
     19. i++;
     20. }
     21. }
     22. Visual Basic
     23. Private Sub ExcelButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
     24. Dim excel As Object = AutomationFactory.CreateObject("Excel.Application")
     25. excel.Visible = True
     26. Dim workbook As Object = excel.workbooks
     27. workbook.Add()
     28. Dim sheet As Object = excel.ActiveSheet
     29. Dim cell As Object = Nothing
     30. Dim i As Integer = 1
     31. ' Populate the excel sheet
     32. For Each item In (TryCast(LayoutRoot.DataContext, Bookmarks)).Sites cell = sheet.Cells(i, 1)
     33. cell.Value = item.Title
     34. cell = sheet.Cells(i, 2)
     35. cell.Value = item.Uri
     36. cell.ColumnWidth = 100
     37. i += 1
     38. Next item
     39. End Sub
     40. 
     41. Figure 2
     42. Instance of Excel Page
  4. For exporting to Word copy and paste the below event handler into MainPage.xaml.cs. Here we’re using the AutomationFactory to create an instance of Word. Then we construct a write data to the newly created page in Word. The interesting piece about this call to word is the last line where the SaveAs method is called. Calling this method saves the word doc to your Documents folder.
     1. C#
     2. void WordButton\_Click(object sender, RoutedEventArgs e)
     3. {
     4. if (AutomationFactory.IsAvailable)
     5. {
     6. dynamic word = AutomationFactory.CreateObject("Word.Application");
     7. word.Visible = true;
     8. word.Documents.Add();
     9. word.Selection.TypeText("Bookmarks");
     10. word.Selection.TypeText(string.Format("Exported at: {0}", DateTime.Today.ToShortDateString()));
     11. word.Selection.TypeParagraph();
     12. foreach (var item in (LayoutRoot.DataContext as Bookmarks).Sites)
     13. {
     14. word.Selection.TypeText(string.Format("{0} \vURL: {1}\v", item.Title, item.Uri));
     15. }
     16. word.ActiveDocument.SaveAs("BookmarksFromSL");
     17. }
     18. }
     19. Visual Basic
     20. Private Sub WordButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
     21. If AutomationFactory.IsAvailable Then
     22. Dim word As Object = AutomationFactory.CreateObject("Word.Application")
     23. word.Visible = True
     24. word.Documents.Add()
     25. word.Selection.TypeText("Bookmarks")
     26. word.Selection.TypeText(String.Format("Exported at: {0}",Date.Today.ToShortDateString()))
     27. word.Selection.TypeParagraph()
     28. For Each item In (TryCast(LayoutRoot.DataContext, Bookmarks)).Sites
     29. word.Selection.TypeText(String.Format("{0} " & vbVerticalTab & "URL: {1}" & vbVerticalTab, item.Title, item.Uri))
     30. Next item
     31. word.ActiveDocument.SaveAs("BookmarksFromSL")
     32. End If
     33. End Sub
     34. 
     35. Figure 3
     36. Instance of Word Page
  5. For calling to Outlook copy and paste the below event handler into MainPage.xaml.cs. Here we’re using the AutomationFactory to create an instance of Outlook and then construct an e-mail.
     1. C#
     2. void OutlookButton\_Click(object sender, RoutedEventArgs e)
     3. {
     4. if (AutomationFactory.IsAvailable)
     5. {
     6. dynamic outlook = AutomationFactory.CreateObject("Outlook.Application");
     7. dynamic mail = outlook.CreateItem(0);
     8. mail.To = "me@myname.com";
     9. mail.Subject = "My Bookmarks";
     10. StringBuilder sb = new StringBuilder();
     11. foreach (var item in (LayoutRoot.DataContext as Bookmarks).Sites)
     12. {
     13. sb.Append(string.Format("{0} \vURL: {1}\v\v", item.Title, item.Uri));
     14. }
     15. mail.Body = sb.ToString();
     16. mail.Display();
     17. }
     18. }
     19. Visual Basic
     20. Private Sub OutlookButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
     21. If AutomationFactory.IsAvailable Then
     22. Dim outlook As Object = AutomationFactory.CreateObject("Outlook.Application")
     23. Dim mail As Object = outlook.CreateItem(0)
     24. mail.To = "me@myname.com"
     25. mail.Subject = "My Bookmarks"
     26. Dim sb As New StringBuilder()
     27. For Each item In (TryCast(LayoutRoot.DataContext, Bookmarks)).Sites
     28. sb.Append(String.Format("{0} " & vbVerticalTab & "URL: {1}" & vbVerticalTab & vbVerticalTab, item.Title, item.Uri))
     29. Next item
     30. mail.Body = sb.ToString()
     31. mail.Display()
     32. End If
     33. End Sub
     34. 
     35. Figure 4

Instance of Outlook Page

* 1. Finally, the last button doesn’t necessarily export the Bookmarks, rather it open Notepad. This is more or less to demonstrate a call to a non Office product. Using the WScript.Shell you could call a number of other external programs. So here’s your freebie. Enjoy, but make sure to play nice when working with Elevated Privileges and Out-of-Browser.
     1. C#
     2. void PowerPointButton\_Click(object sender, RoutedEventArgs e)
     3. {
     4. if (AutomationFactory.IsAvailable)
     5. {
     6. dynamic cmd = AutomationFactory.CreateObject("WScript.Shell");
     7. cmd.Run(@"c:\windows\notepad.exe", 1, true);
     8. }
     9. }
     10. Visual Basic
     11. Private Sub PowerPointButton\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
     12. If AutomationFactory.IsAvailable Then
     13. Dim cmd As Object = AutomationFactory.CreateObject("WScript.Shell")
     14. cmd.Run("c:\windows\notepad.exe", 1, True)
     15. End If
     16. End Sub
     17. 
     18. Figure 5
     19. Open Notepad

Summary

* + 1. In this exercise you examined how to create a Silverlight project for the desktop. Additionally you explored many aspects of the out-of-browser feature and satisfied the following requirements:
  + Enabling Out of Browser
  + Display the correct interface based on the context the application is running under
  + Prompt the user to install the application from the web to their desktop
  + Enable elevated privileges
  + Customize the look of your application
  + Enable the exporting of data to Office products
    1. This application demonstrated some of the more basic features of out-of-browser. In other labs you will take a closer look at the more advanced features.