Elements

Introduction

- The Silverlight platform provides us with a set of elements. It is a subset derived from WPF.
- The Silverlight Toolkit offers many more specialized elements.

Text Elements

- The `TextBlock` is one of the simplest available elements.

  We use it to for displaying a simple text.

```
<TextBlock>Bla Bla Bla</TextBlock>
<TextBlock Text="Bla Bla Bla" ></TextBlock>
<TextBlock Text="Bla Bla Bla" Foreground="Yellow"></TextBlock>
```

Font Properties

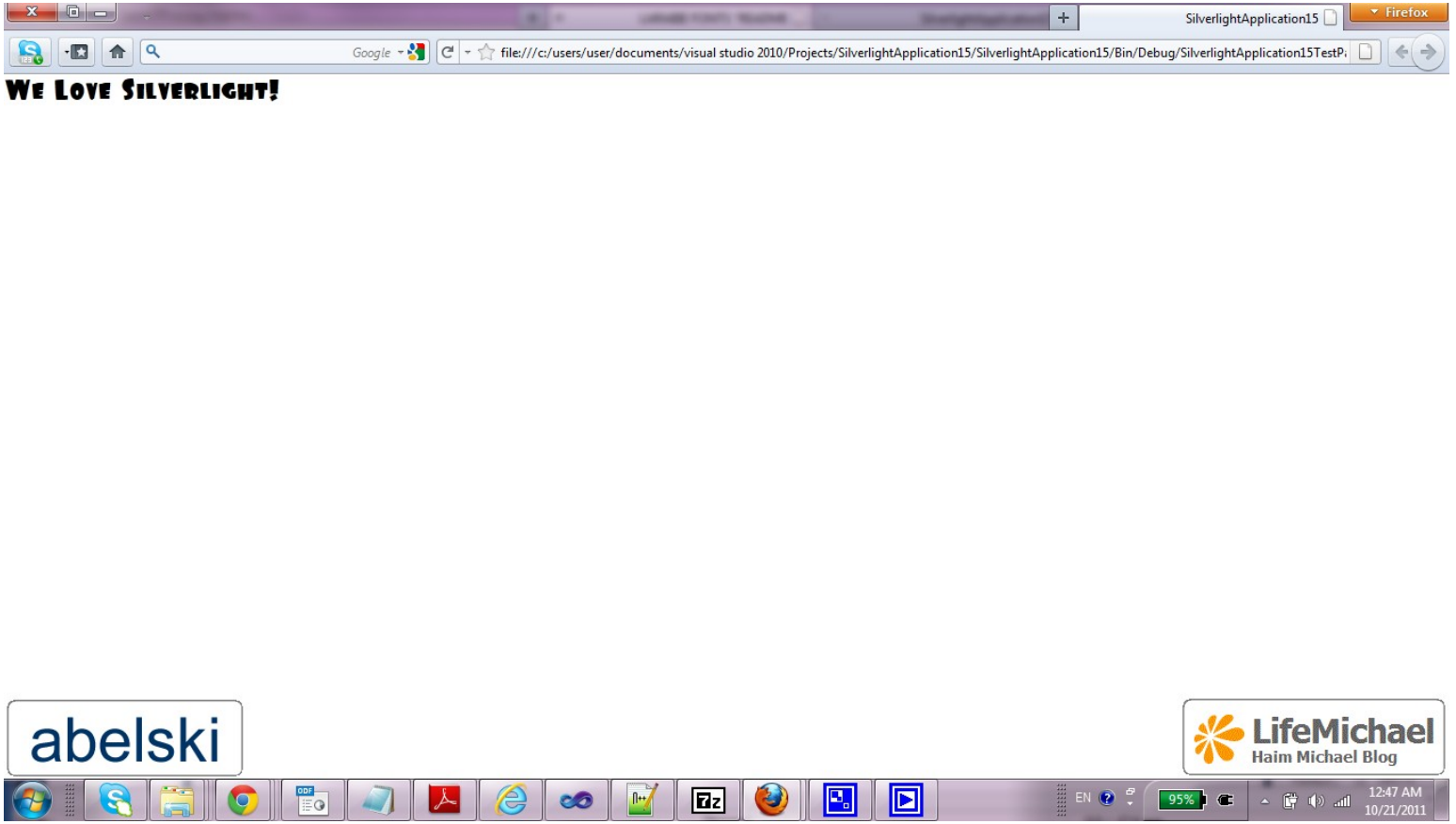
- These properties include `FontFamily`, `FontSize`, `FontStyle`, `FontWeight` and `FontStretch`.

Font Families

- Apart of the font families that Silverlight supports we can add new ones.

```
<UserControl x:Class="SilverlightApplication15.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock FontSize="24" FontFamily="Foo.ttf#Foo">
            We Love Silverlight!
        </TextBlock>
    </Grid>

</UserControl>
```

Font Families



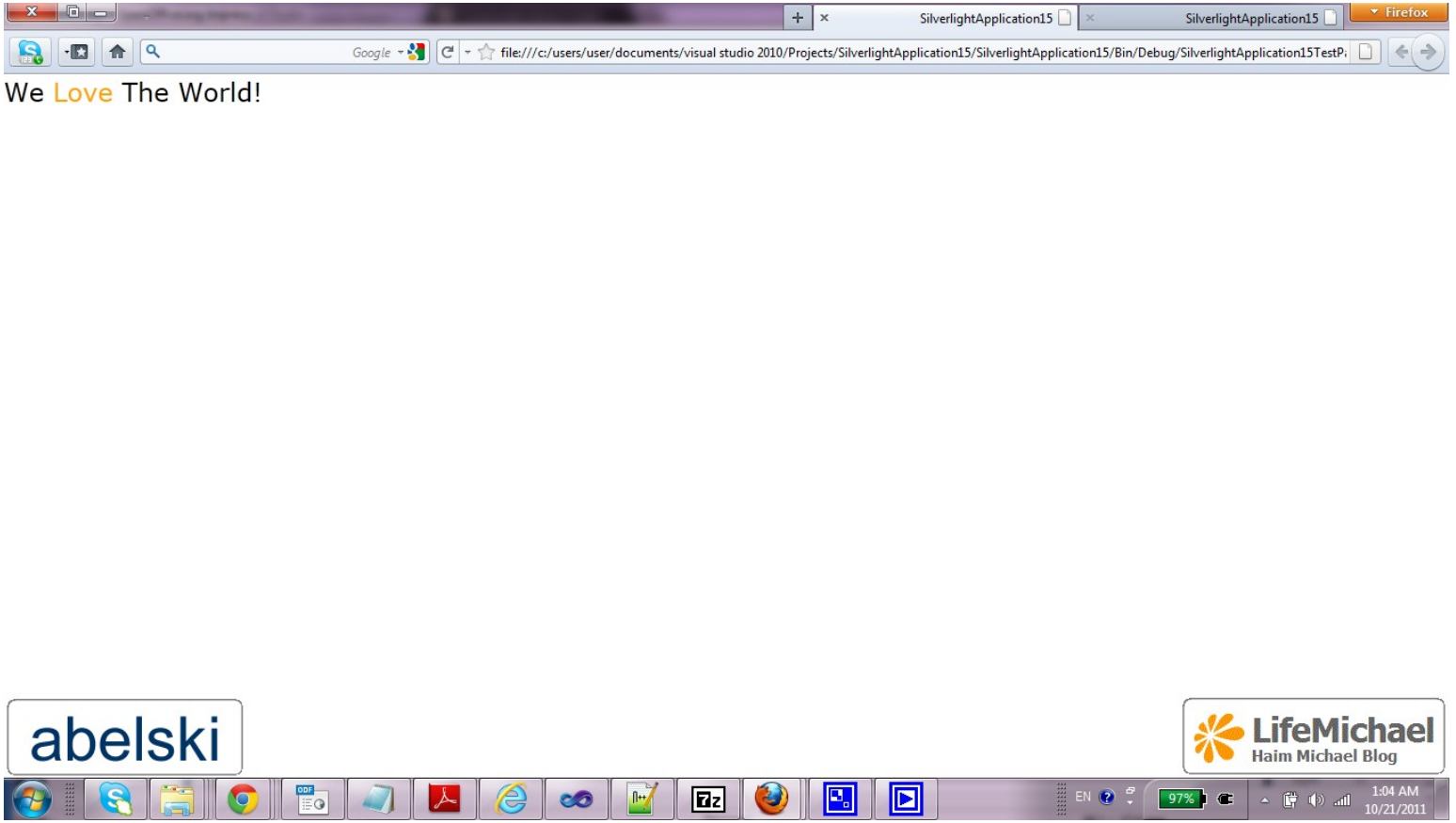
The Run Object

- This object allows us to embed a single text paragraph within another while having each one of them with a different format.

The Run Object

```
<UserControl x:Class="SilverlightApplication15.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock FontSize="24">
            We  <Run Foreground="Orange">Love</Run>  The World!
        </TextBlock>
    </Grid>
</UserControl>
```

The Run Object



Images

- The supported formats include PNG and JPG. GIF is not supported.

```
<Image Source="mypix.jpg"></Image>
```

- Manipulating the image is feasible through the code as well as through the XAML file.

```
img.Source = new BitmapImage(new Uri("mypix.jpg", UriKind.Relative));
```

Images

- We can set the image dimension either using the `Width` and `Height` properties or by placing it within a container that will set the size.
- Using the `Stretch` property we can determine how the image will be resized when the dimension of the image element doesn't meet the dimension of the image itself.

Images

- The `Stretch` possible values include `Fill`, `None`, `Uniform` which is the default value and `UniformToFill`. `Fill` will stretch the image in width and height to fit the Image element. `None` will keep the image native size. `Uniform` will set the largest possible size while keeping the aspect ratio unchanged. `UniformToFill` will size the width and the height proportionately until the image fills the available height and width.

Content Controls

- The content controls hold a piece of content. Usually it will be a single nested element, which is also the difference comparing with layout containers.
- Samples for content controls include `ListItem`, `TabItem`, `ToolTip`, `DataGridCell`, `ChildWindow`, `ScrollViewer`, `Frame` and `ButtonBase` which is the super type for various button types, such as `Button`, `HyperlinkButton`, `ToggleButton` and many others.

Content Controls

- The content controls share the `Content` property through which we can set their content.

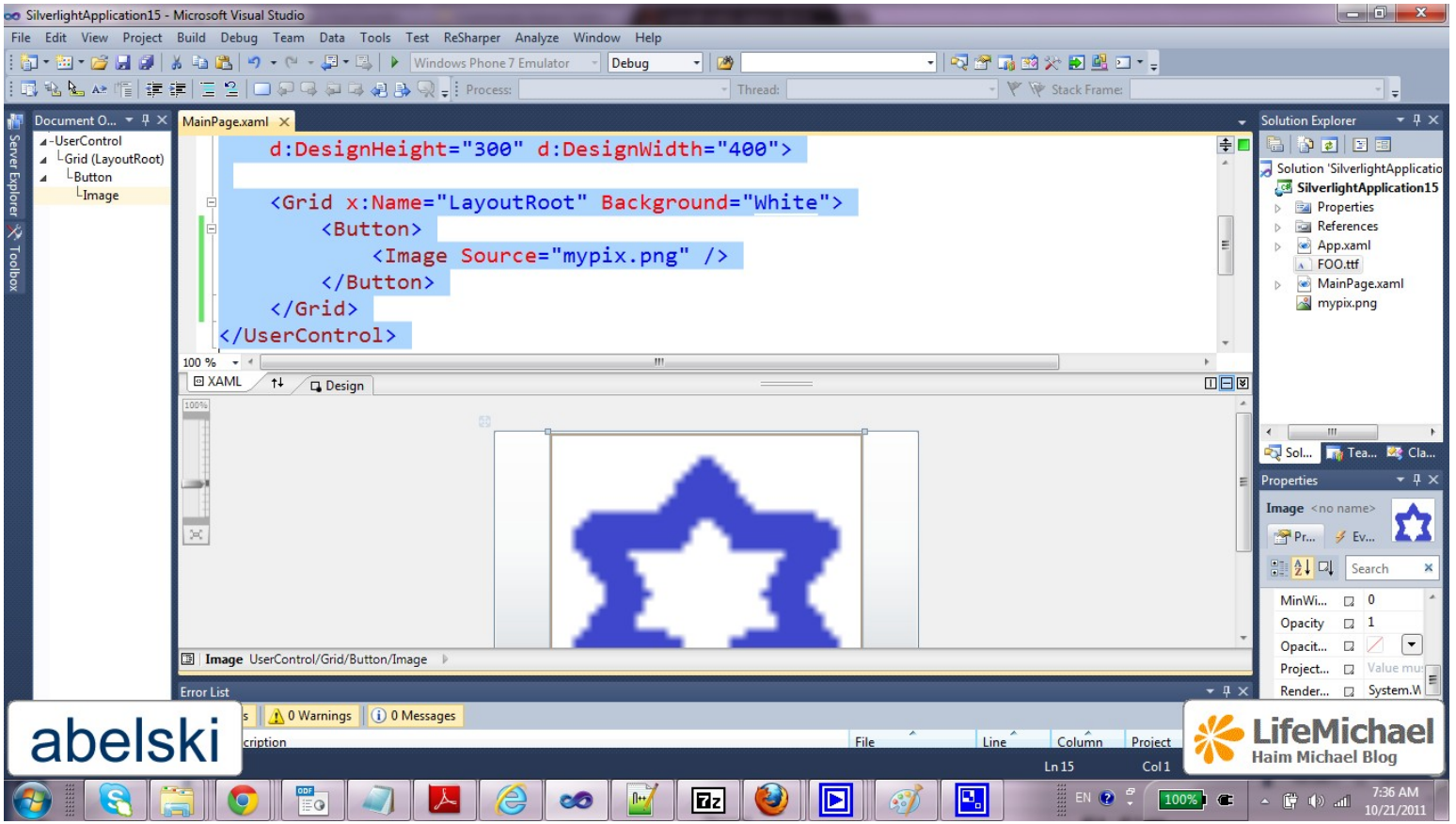
```
<Button Content="OK"/>
```

- We can place the content as a child element as. That allows more sophisticated possibilities.

Content Controls

```
<UserControl x:Class="SilverlightApplication15.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <Button>
            <Image Source="mypix.png" />
        </Button>
    </Grid>
</UserControl>
```

Content Controls

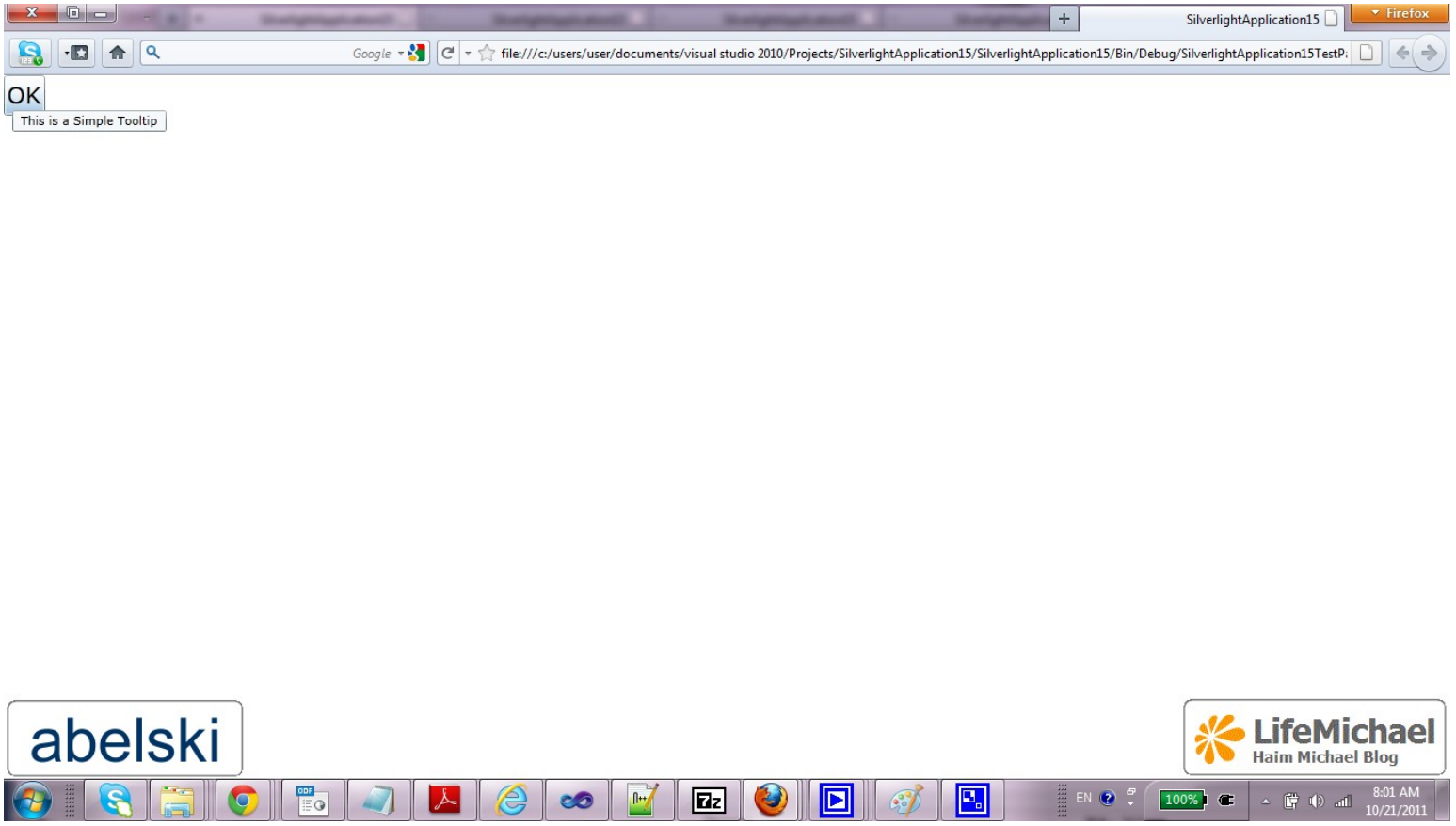


Tooltips

- The tooltips are represented by the `ToolTip` content control. We don't need to add a `ToolTip` element. We can set an attached property and the Silverlight platform will create the tool tip automatically.

```
<Canvas>
        <Button FontSize="22"
                ToolTipService.ToolTip="This is a Simple Tooltip"
                Content="OK"></Button>
</Canvas>
```

Tooltips

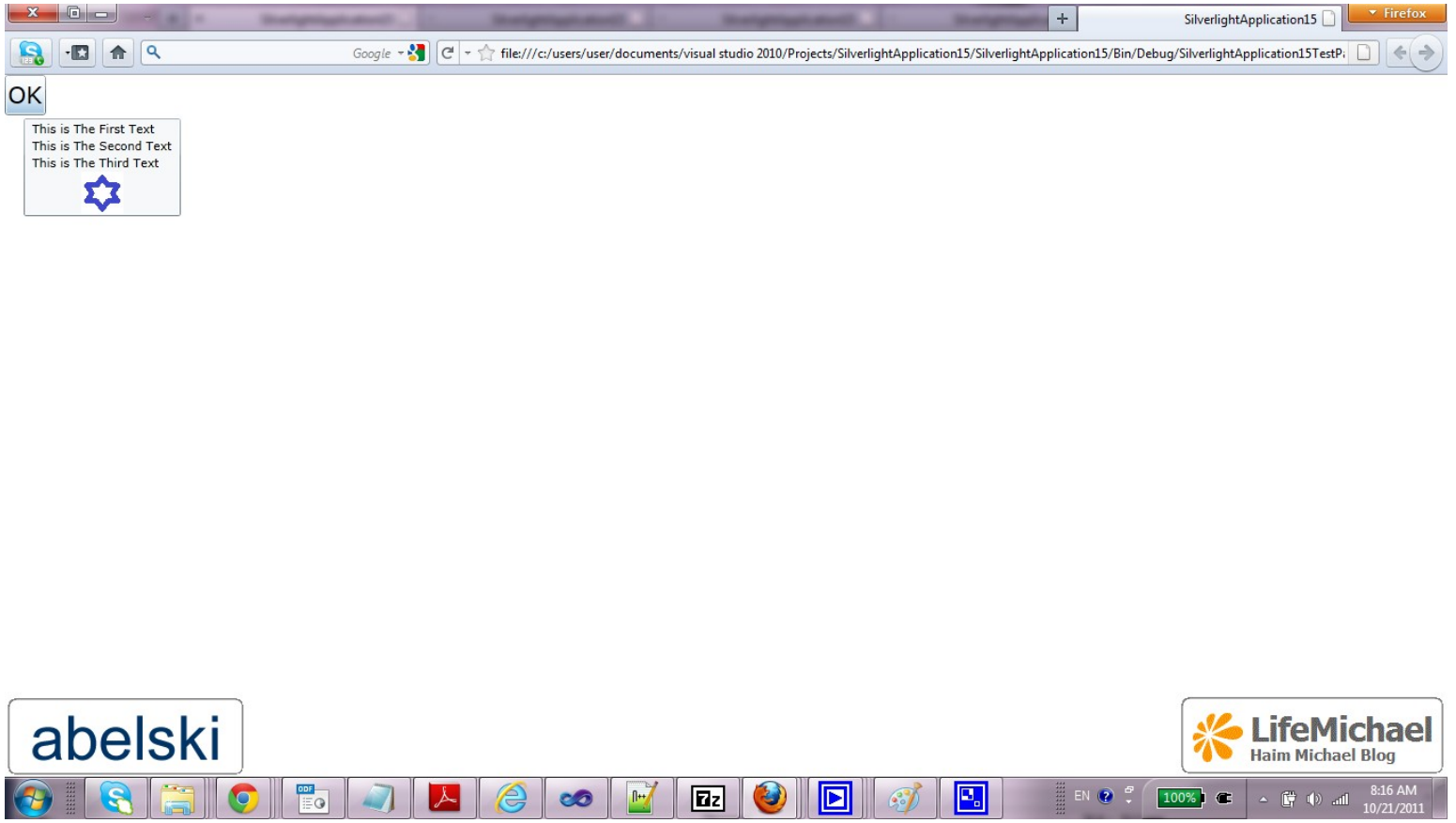


Tooltips

- We can easily customize the tooltip by using the `ToolTipService.ToolTip` associated property.

```
<Canvas>
    <Button FontSize="22" Content="OK">
        <ToolTipService.ToolTip>
            <StackPanel Orientation="Vertical">
                <TextBlock Text="This is The First Text" />
                <TextBlock Text="This is The Second Text" />
                <TextBlock Text="This is The Third Text" />
                <Image Source="mypix.png" />
            </StackPanel>
        </ToolTipService.ToolTip>
    </Button>
</Canvas>
```

Tooltips



Popup Control

- The `Popup` control can hold one single piece of content. It can be any element. That single piece of content is stored within the `Popup.Child` property.
- The `Popup.IsOpen` property controls whether the popup is visible or not.
- The functionality the popup supports is the main difference comparing with a simple tooltip.

Popup Control

- The top-left corner is aligned with the top-left corner of the content region.

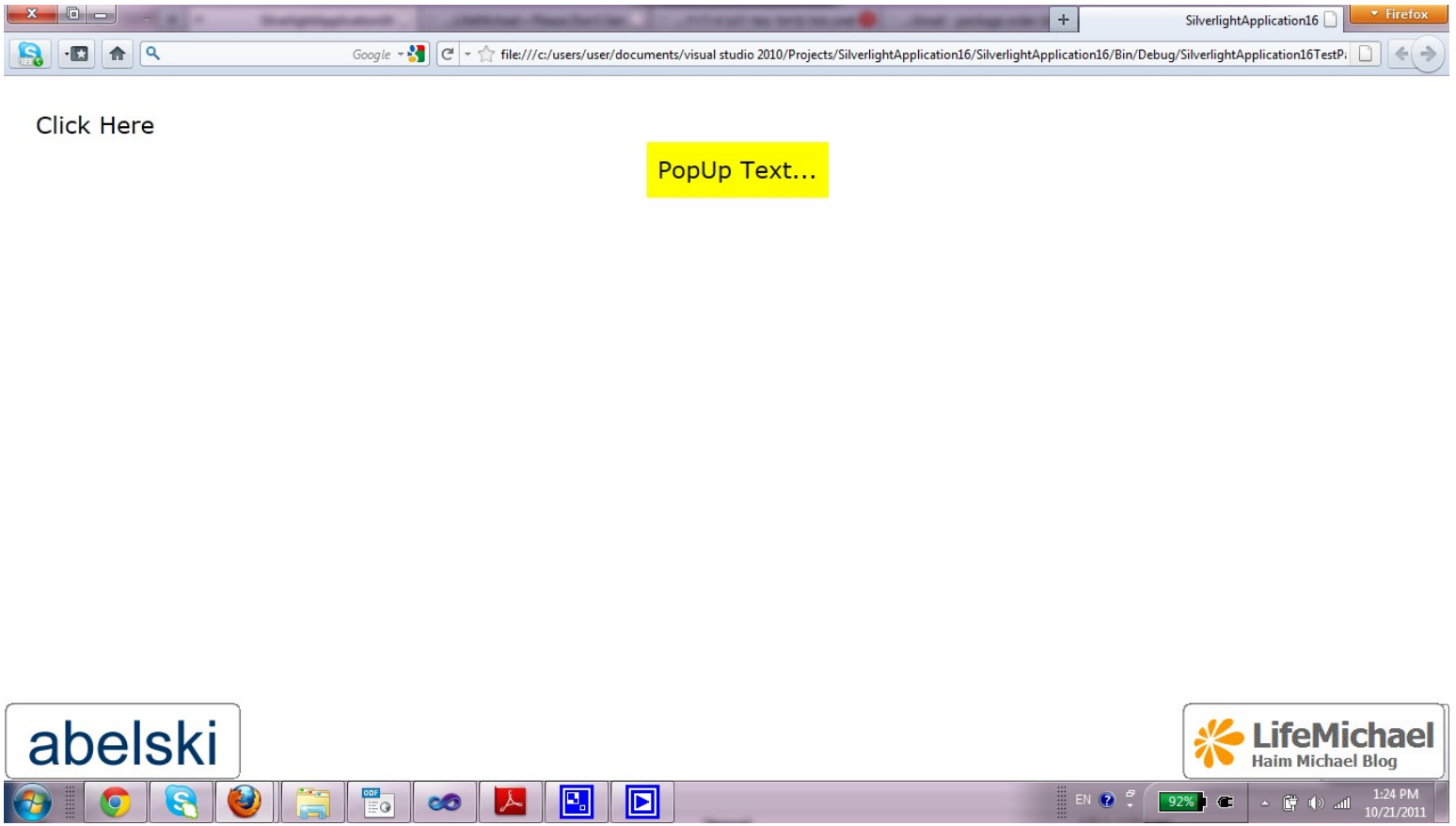
Popup Control

```
namespace SilverlightApplication16
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
                InitializeComponent();
        }
        private void TxtMouseLeftButtonDown(object sender,
                MouseButtonEventArgs e)
        {
                PopUpMenu.IsOpen = true;
        }
        private void PopUpMenuMouseLeftButtonDown(object sender,
                MouseButtonEventArgs e)
        {
                PopUpMenu.IsOpen = false;
        }
    }
}
```

Popup Control

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel Margin="30">
            <TextBlock  TextWrapping="Wrap"
                        MouseLeftButtonDown="TxtMouseLeftButtonDown"
                        Text="Click Here"
                        FontSize="22"/>
            <Popup x:Name="PopUpMenu" MaxWidth="150">
                <Border MouseLeftButtonDown="PopUpMenuMouseLeftButtonDown">
                    <Border.Background>Yellow</Border.Background>
                    <TextBlock FontSize="22" Margin="10"
                      Text="PopUp Text..."></TextBlock>
                </Border>
            </Popup>
        </StackPanel>
    </Grid>
</UserControl>
```

Popup Control



Items Controls

- These are those controls that wrap collections of items. Each item should be an `ItemControl` object.

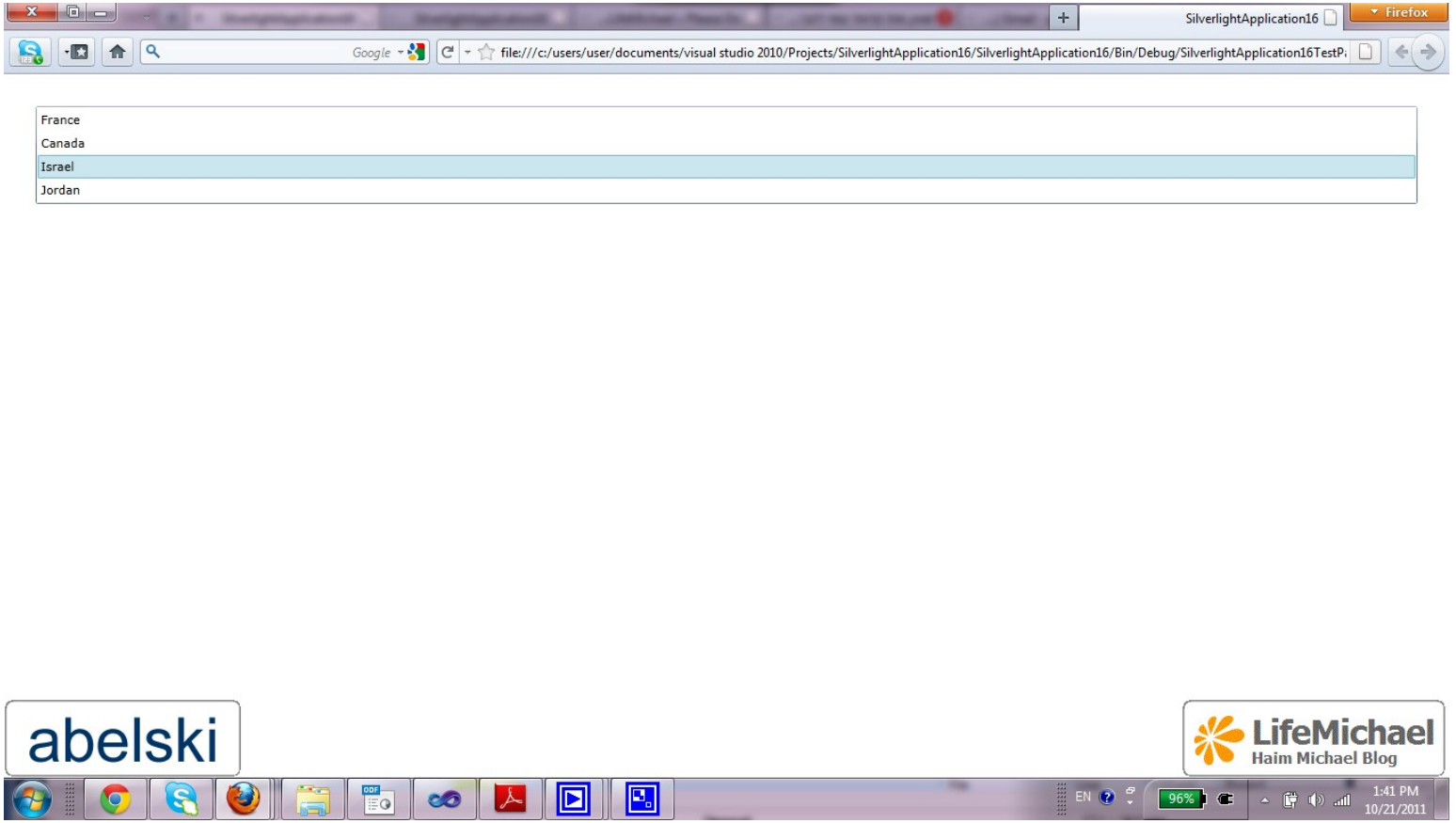
List Box

- This control is one of the simplest items controls to handle. It holds items of the ListBoxItem type.

List Box

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel Margin="30">
            <ListBox>
                <ListBoxItem Content="France"></ListBoxItem>
                <ListBoxItem Content="Canada"></ListBoxItem>
                <ListBoxItem Content="Israel"></ListBoxItem>
                <ListBoxItem Content="Jordan"></ListBoxItem>
            </ListBox>
        </StackPanel>
    </Grid>
</UserControl>
```

List Box



List Box

- `ListBox` is capable of creating the `ListBoxItem` objects implicitly. We can just place our objects within the `ListBox` element and the `ListBoxItem` objects will be automatically created.

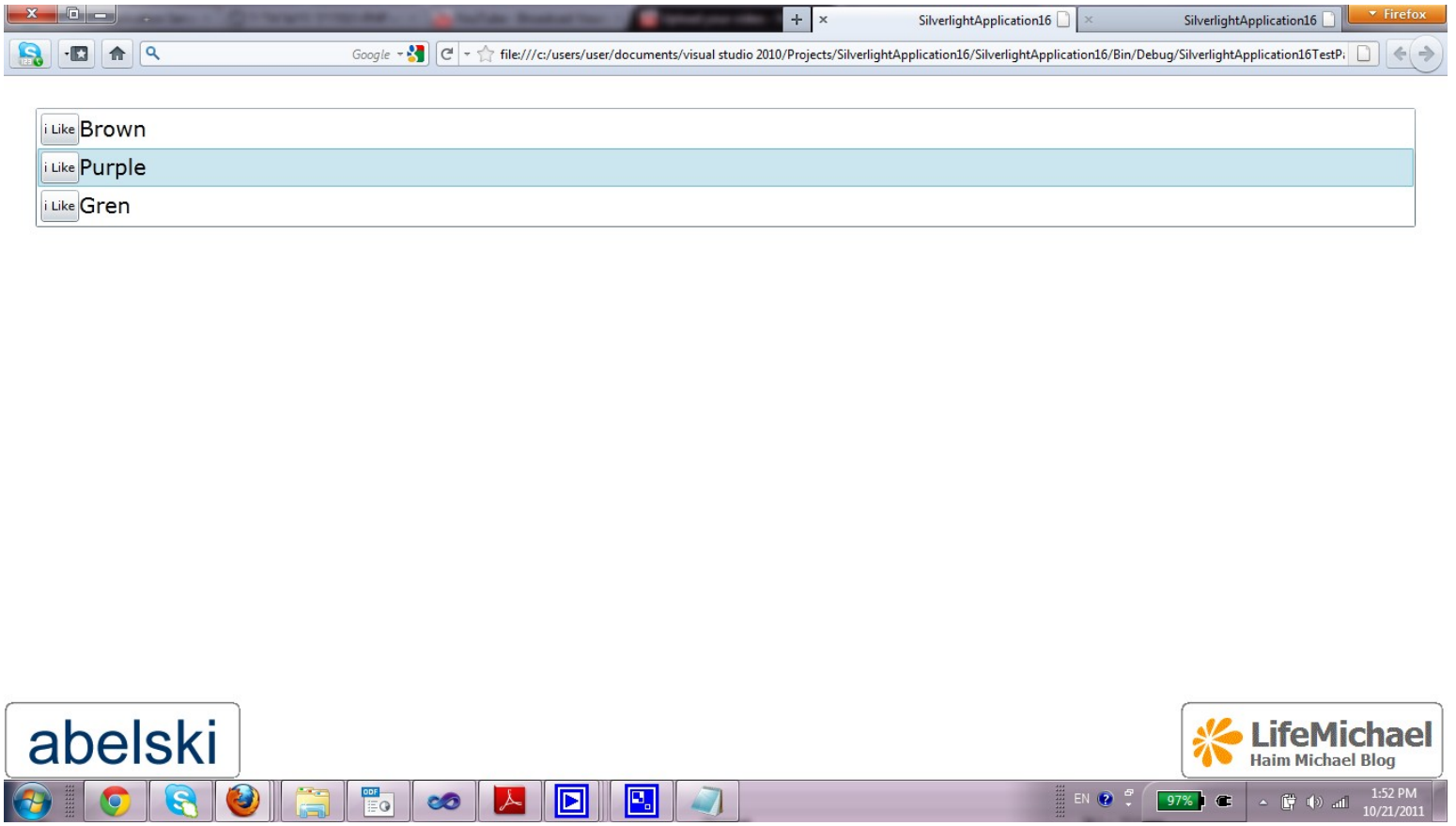
List Box

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel Margin="30">
            <ListBox >

                <StackPanel Orientation="Horizontal">
                    <Button Content="i Like" />
                    <TextBlock FontSize="20" Text="Brown"/>
                </StackPanel>
```

List Box

```
                <StackPanel Orientation="Horizontal">
                    <Button Content="i Like" />
                <TextBlock FontSize="20" Text="Purple"/>
                </StackPanel>

            <StackPanel Orientation="Horizontal">
                <Button Content="i Like" />
                <TextBlock FontSize="20" Text="Gren"/>
                </StackPanel>

            </ListBox>
        </StackPanel>
    </Grid>
</UserControl>
```

List Box



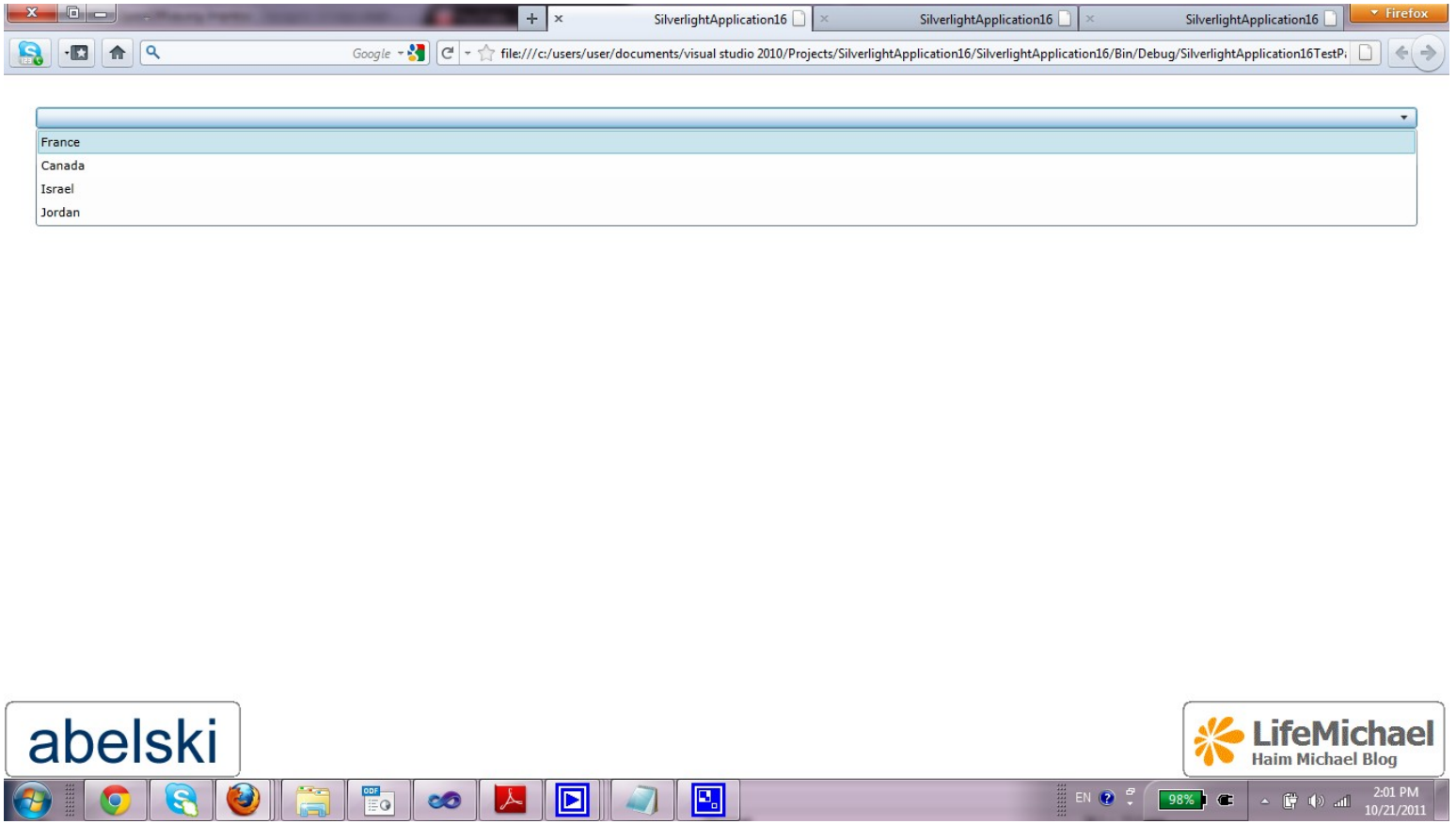
Combo Box

- This control is very similar to list box. Each item is represented using a `ComboBoxItem` object.
- Unlike the list box, the combo box uses a drop down list and allows only one item to be selected.

Combo Box

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel Margin="30">
            <ComboBox>
                <ComboBoxItem Content="France"></ComboBoxItem>
                <ComboBoxItem Content="Canada"></ComboBoxItem>
                <ComboBoxItem Content="Israel"></ComboBoxItem>
                <ComboBoxItem Content="Jordan"></ComboBoxItem>
            </ComboBox>
        </StackPanel>
    </Grid>
</UserControl>
```

Combo Box



Tab Control

- The `TabControl` allows us to create a simple set of tabs. Each tab is represented using a `TabItem` element.
- This control is defined in a separated assembly. When we add it to a page the Visual Studio IDE will automatically add a reference to the `System.Windows.Controls.dll` assembly and map a new XML namespace accordingly.

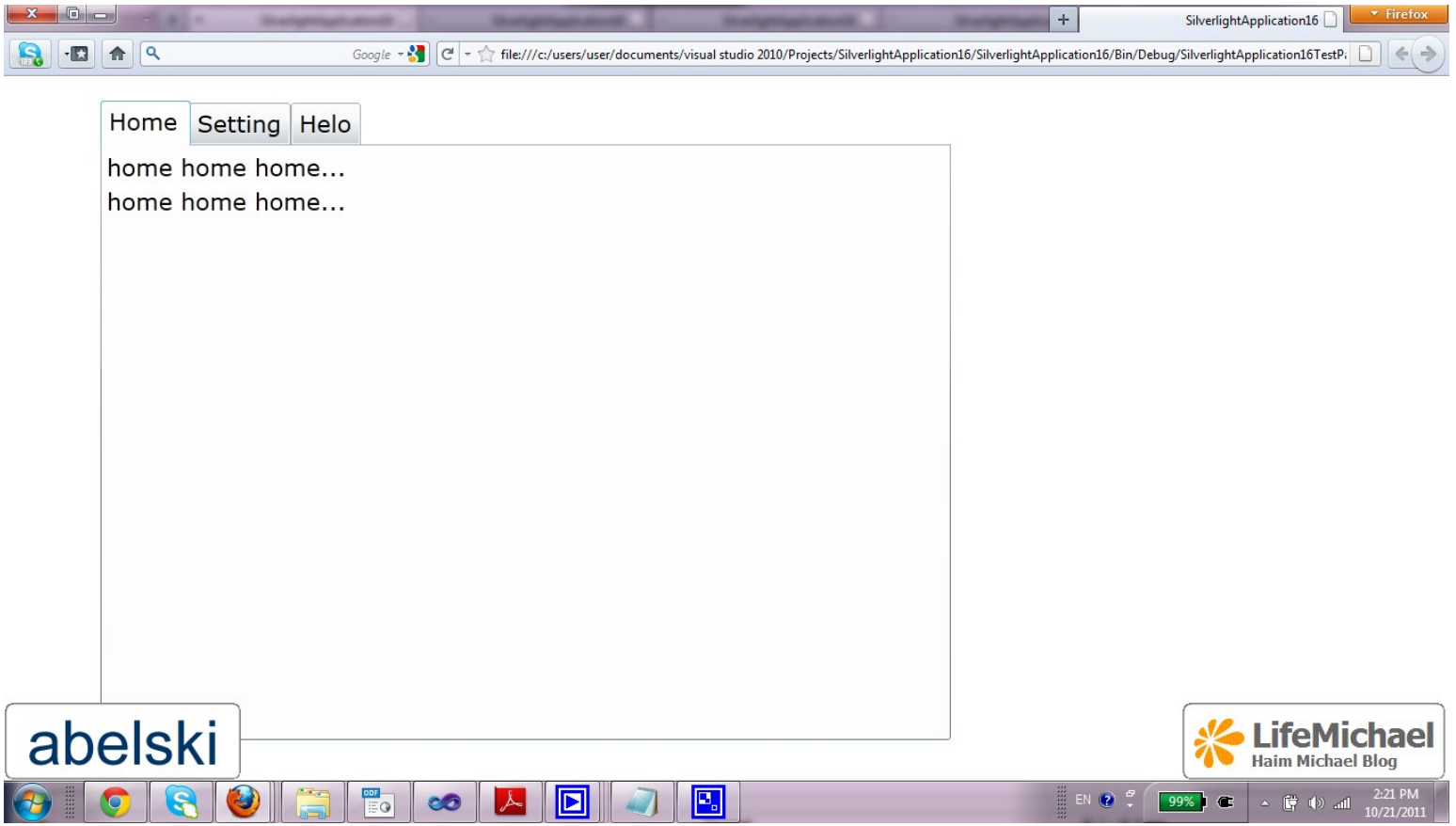
Tab Control

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400"
    xmlns:sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">

    <Grid x:Name="LayoutRoot" Background="White">
        <sdk:TabControl Height="100" HorizontalAlignment="Left"
            Margin="91,23,0,0" Name="tabControl1" VerticalAlignment="Top"
            Width="200">
            <sdk:TabItem Header="Home" Name="homeTab">
                <StackPanel>
                    <TextBlock>home home home...</TextBlock>
                    <TextBlock>home home home...</TextBlock>
                </StackPanel>
            </sdk:TabItem>
```

Tab Control

```
            <sdk:TabItem Header="Setting" Name="settingTab">
                <StackPanel>
                    <TextBlock>setting setting setting...</TextBlock>
                    <TextBlock>setting setting setting...</TextBlock>
                </StackPanel>
            </sdk:TabItem>
            <sdk:TabItem Header="Helo" Name="helpTab">
                <StackPanel>
                    <TextBlock>help help help...</TextBlock>
                    <TextBlock>help help help...</TextBlock>
                </StackPanel>
            </sdk:TabItem>
        </sdk:TabControl>
    </Grid>
</UserControl>
```

Tab Control



Text Box

- The `TextBox` control holds a string that is accessible through the `Text` property.

Password Box

- The `PasswordBox` control looks like `TextBox`. Unlike the `TextBox` it displays symbols instead of the text.

Auto Complete Box

- The `AutoCompleteBox` control looks like `TextBox`. Unlike the `TextBox` it displays a drop down list of suggestions to choose from.

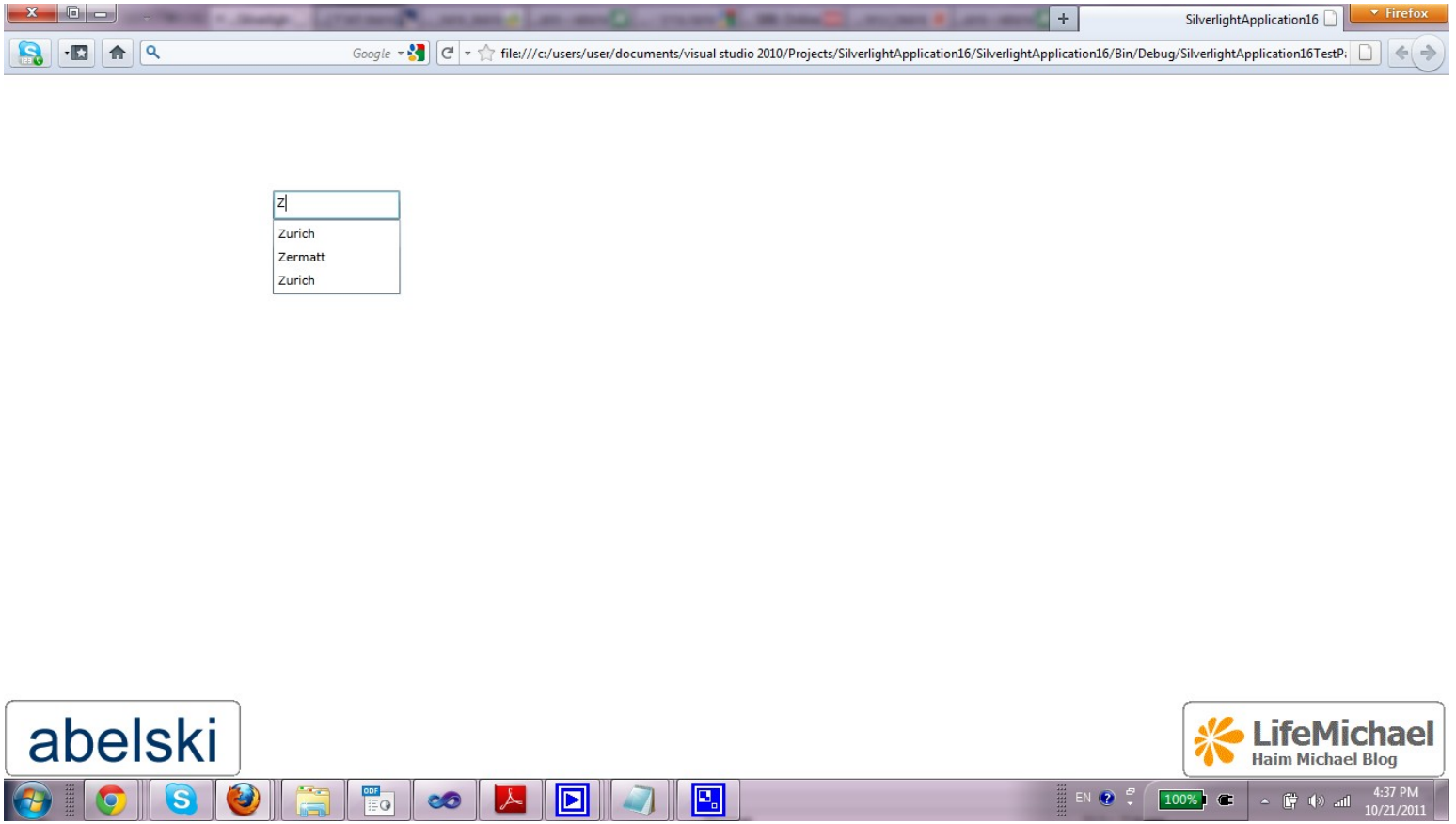
Auto Complete Box

```
namespace SilverlightApplication16
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            string[] cities = {"Tel-Aviv","Zurich","Zermatt","Milano",
               "Haifa","Jerusalem","Zurich","Eilat"};
            CityBox.ItemsSource = cities;
        }
    }
}
```

Auto Complete Box

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="600" d:DesignWidth="800"
    xmlns:sdk=
        "http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">

    <Grid x:Name="LayoutRoot" Background="White">
        <sdk:AutoCompleteBox Height="28" HorizontalAlignment="Left"
            Margin="253,108,0,0" Name="CityBox" VerticalAlignment="Top"
            Width="120" />
    </Grid>

</UserControl>
```

Auto Complete Box



Rich Text Box

- The `RichTextBox` control allows us to display text in a rich format.

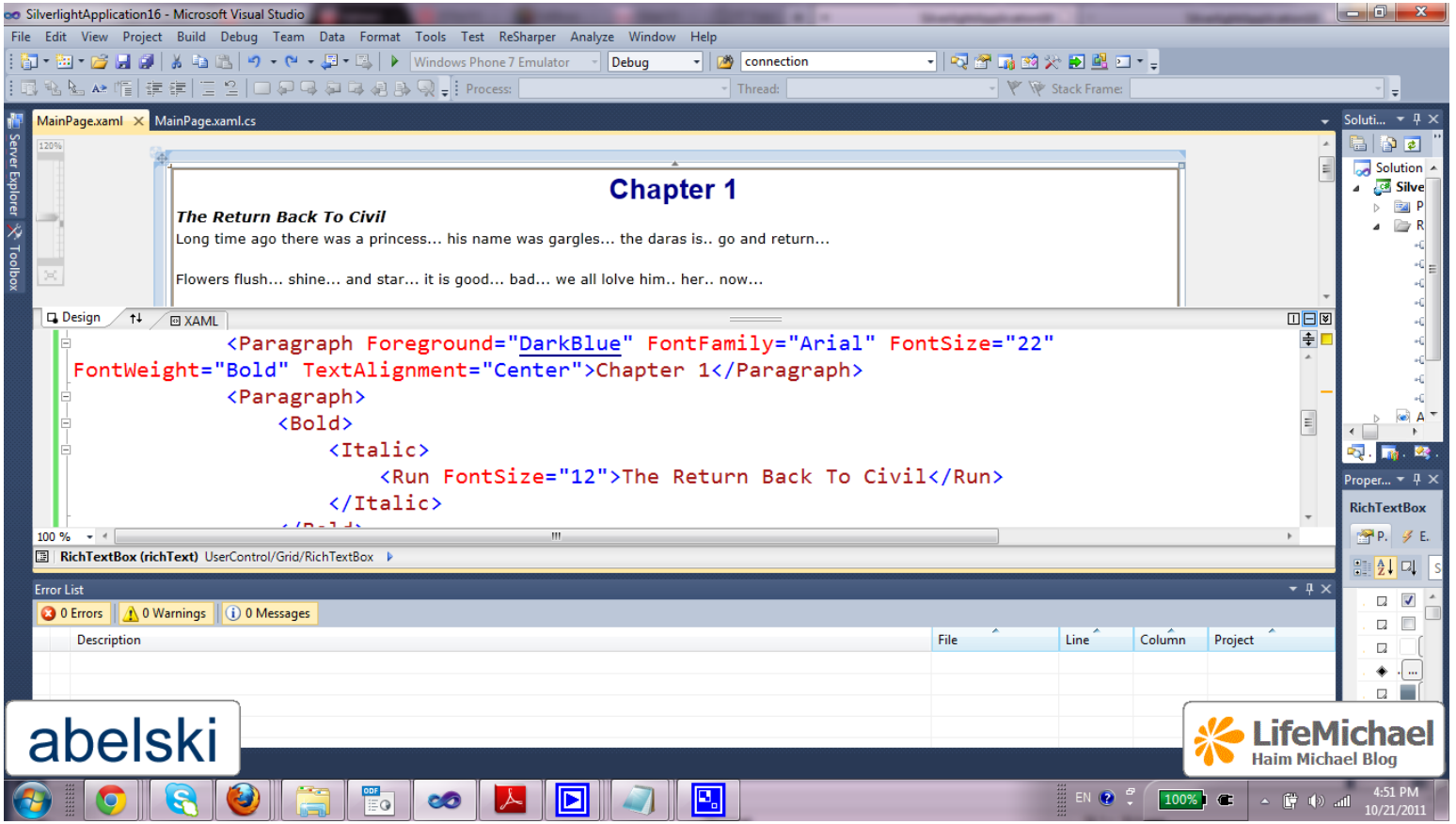
Rich Text Box

```
<UserControl x:Class="SilverlightApplication16.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="600" d:DesignWidth="800"
    xmlns:sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">
    <Grid x:Name="LayoutRoot" Background="White">


        <RichTextBox Margin="5" x:Name="richText">
            <Paragraph Foreground="DarkBlue" FontFamily="Arial" FontSize="22"
            FontWeight="Bold" TextAlignment="Center">Chapter 1</Paragraph>
```

Rich Text Box

```
            <Paragraph>
                <Bold>
                    <Italic>
                        <Run FontSize="12">The Return Back To Civil</Run>
                    </Italic>
                </Bold>
            </Paragraph>
            <Paragraph>
                 Long time ago there was a princess... his name was gargles...
                 the daras is.. go and return...
                <LineBreak></LineBreak>
            </Paragraph>
            <Paragraph>
                 Flowers flush... shine... and star... it is good... bad...
                 we all lolve him.. her.. now...
            </Paragraph>
        </RichTextBox>

    </Grid>
</UserControl>
```

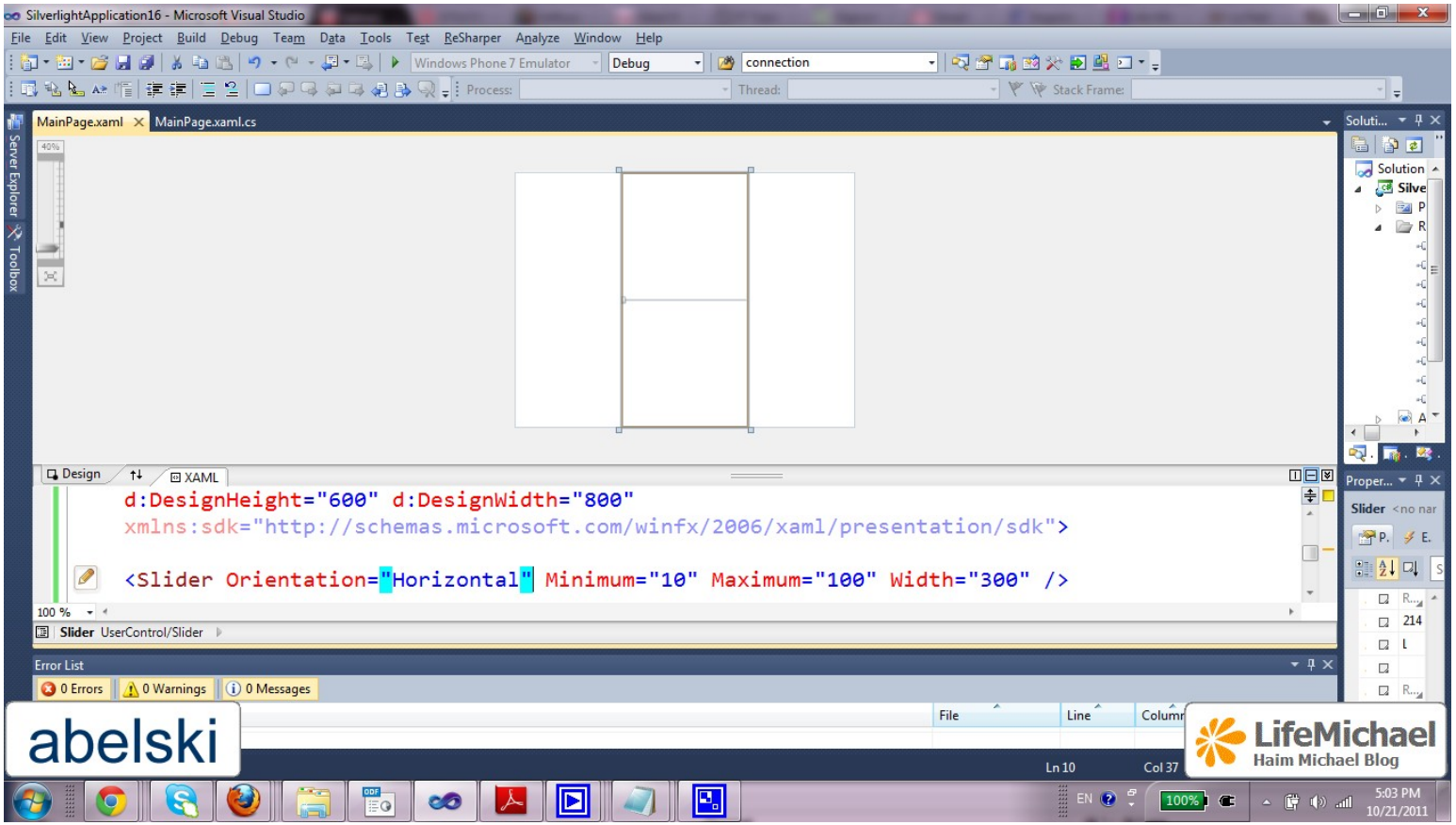
Rich Text Box



Slider

- The sliders allows the user to set numeric values in a convenient way.

Slider



ProgressBar

- The progress bar is very similar to the slider. Unlike the slider it isn't interactive.