Silverlight

abelski

Introduction

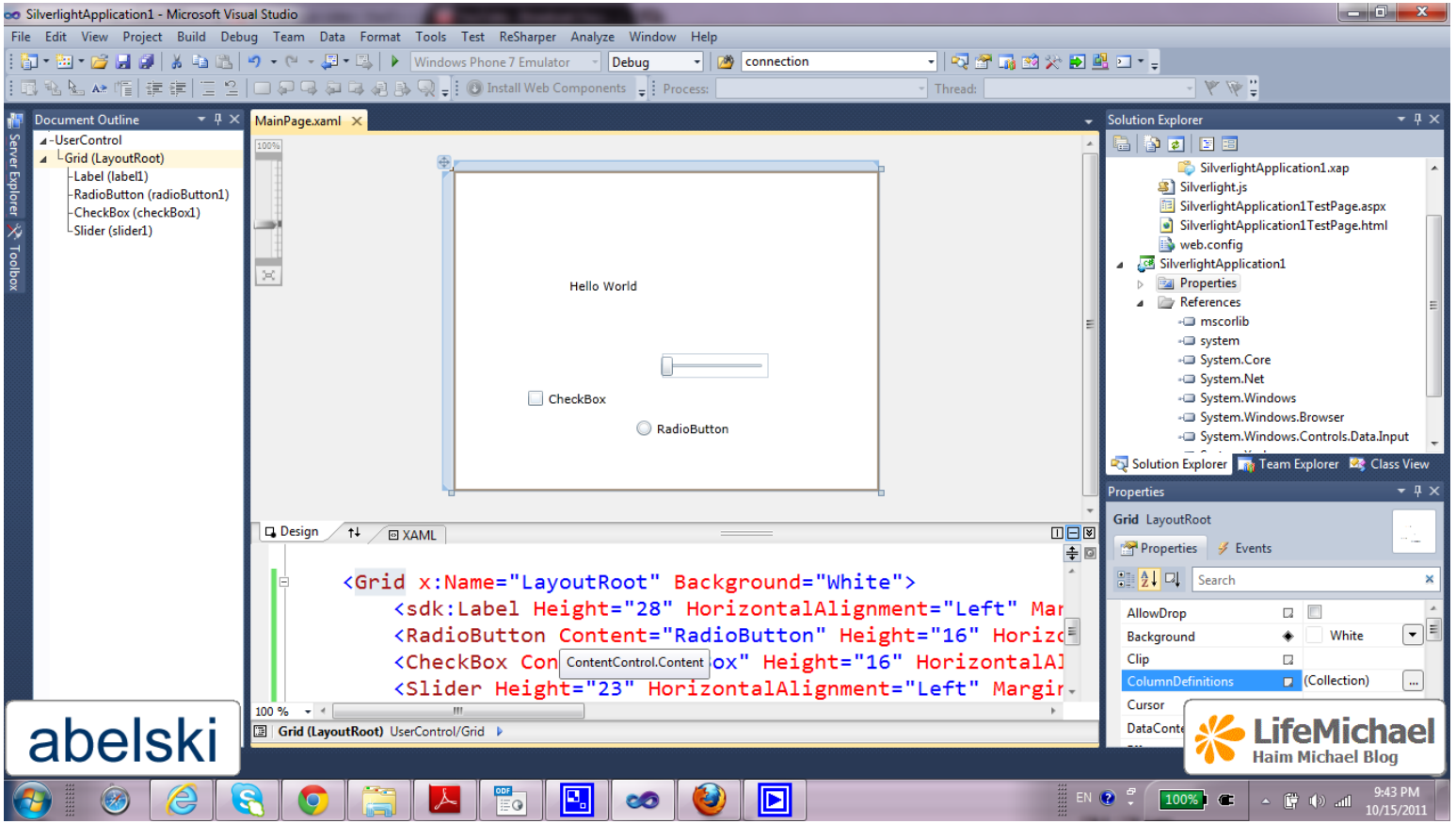
Introduction

- It is possible to develop a Silverlight application either using the Visual Studio or the Expression Blend.
- The Visual Studio is optimized for developers. The Expression Blend is optimized for designers.

Hello World

- We can easily create a new Silverlight project using the Visual Studio. We can select using either C# or VB.NET.

Hello World



The `App.xaml` File

- The `App.xaml` and `App.cs.xaml` include the configuration for our Silverlight application.
- Within these files we can define resources that will be available for all pages.
- Within these files we can interact with the application events, such as the startup and the errors handling.

The `MainPage.xaml` File

- The `MainPage.xaml` and `MainPage.cs.xaml` files define the initial user interface that will be shown when the application starts.

The `AppManifest.xaml` File

- This file lists the assemblies that our application uses. We can find it within the `Properties` folder.

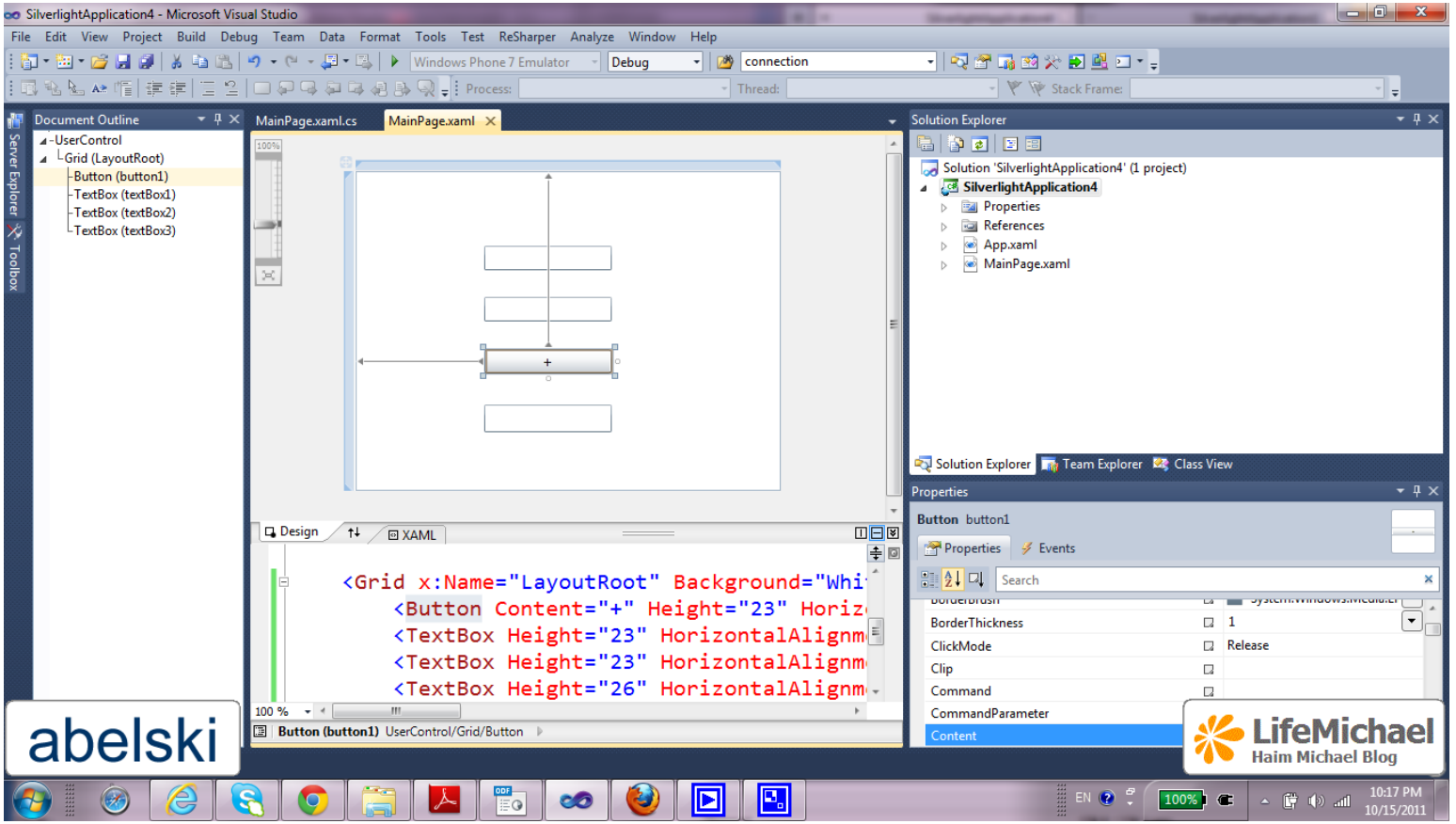
The `AssemblyInfo.cs` File

- This file contains information about our project (name, version, publisher etc.).
- This information will be embedded into our project assembly.

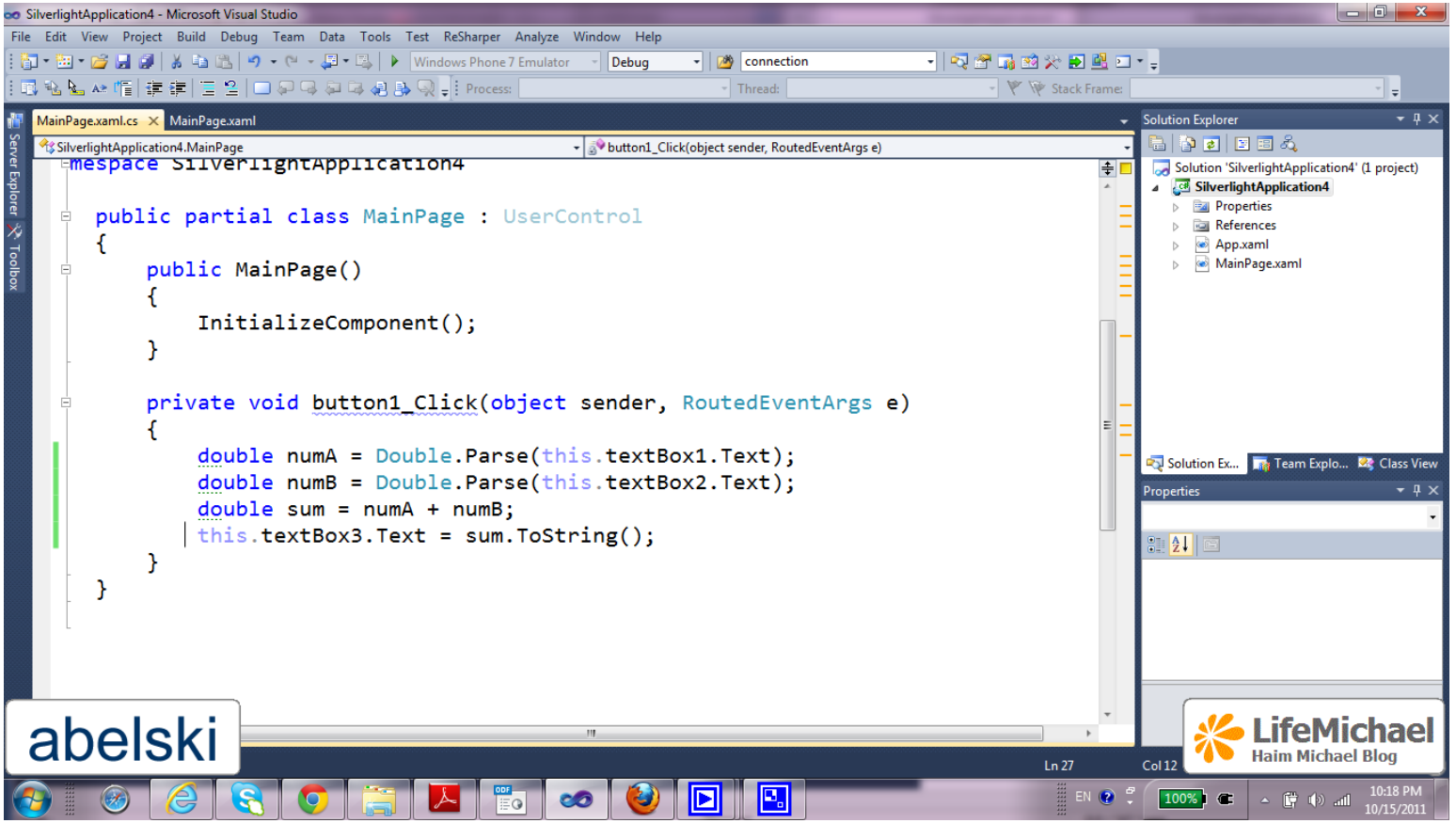
Events Handling

- Handling the user events is done similarly to handling user events in Java Script.

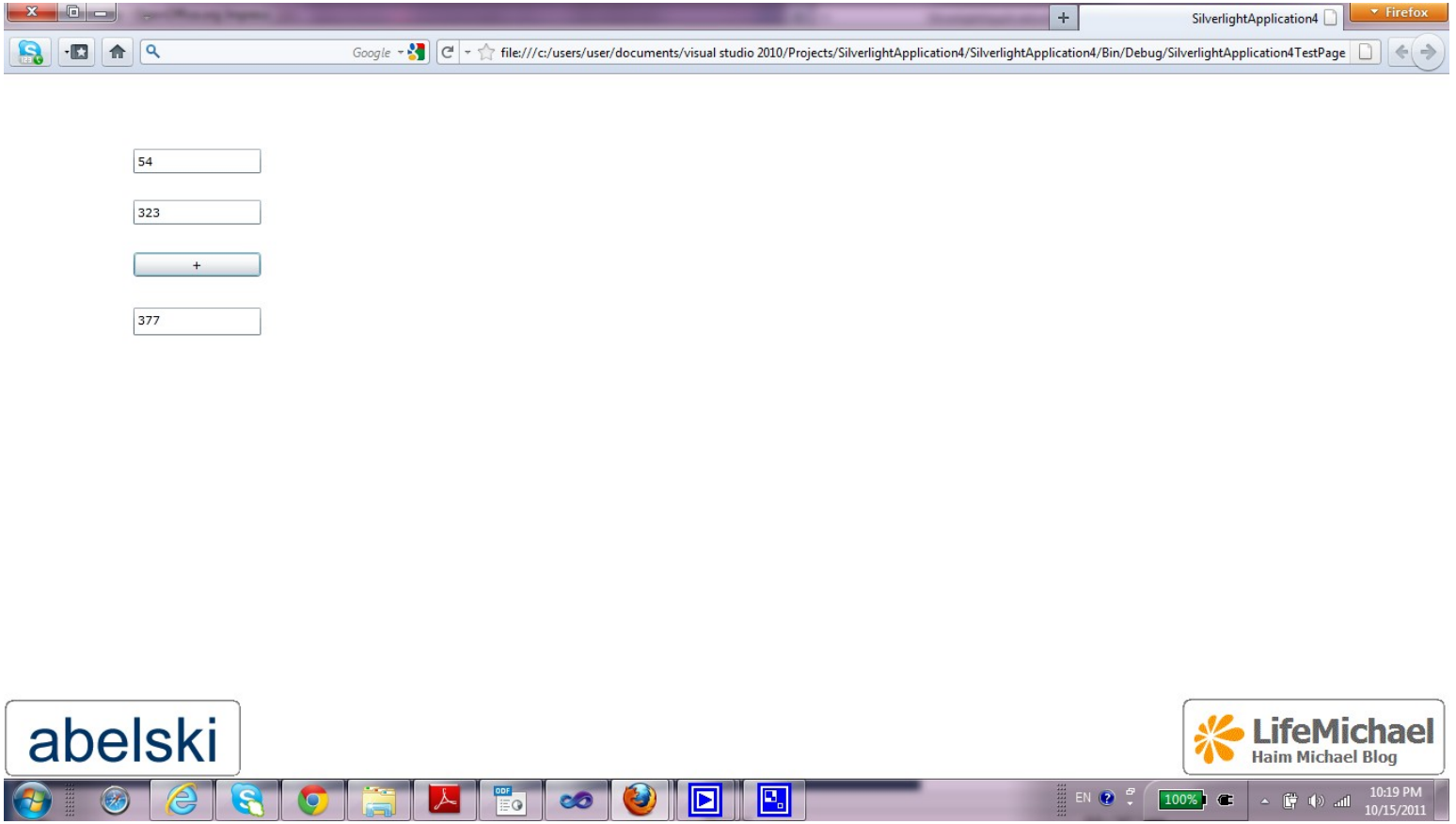
Events Handling



Events Handling



Events Handling



Silverlight Class Libraries

- Silverlight doesn't include all classes we can find in WPF and those it includes don't offer the same capabilities as in WPF.

Deployment

- The XAP file wraps the Silverlight application units, including the application manifest and the assemblies.
- The XAP file is a compressed ZIP archive. When the client receives it the XAP file is decompressed. The time required to download the application is reduced.
- When our Silverlight application uses add-on assemblies we will find them within the XAP file.

Deployment

- In order to deploy our Silverlight application we just need to copy the XAP file to a web server together with the test page or a similar other HTML file that includes the Silverlight content region.

Silverlight Add-on Assemblies

- In order to achieve lean execution environment some of the functionality was removed from the core runtime and was placed in separated add-on assemblies.
- These assemblies are considered to be part of the Silverlight platform but if we want to use them we will need to pack them with our application.

Assembly Caching

- It is a deployment technique that allows us to leave dependent assemblies out of our XAP file in separated ZIP files in the same folder.
- The application start time is reduced. Clients keep cached copies of frequently used assemblies.
- By default, the Silverlight applications we develop are not configured to use this technique.

The HTML Entry Page

- The HTML test page doesn't contain Silverlight markup or code. It just sets up the content region for the Silverlight plug-in.

The Mark of The Web

- The mark of the web is an auto generated additional comment the HTML test page includes.
- The web browser uses this mark in order to provide a less restricted execution environment.

XAML

- The XAML (short for Extensible Application Markup Language and pronounced zammel) is a markup language.
- It was initially designed as part of Windows Presentation Foundation (WPF).
- We use XAML for when creating the Silverlight user interface.

The Code Behind

- The code behind is where we write the events handling code.

Elements Names

- The names the elements have allow us to interact with them pro-grammatically.

```
<Grid x:Name="TheRootLayout">
</Grid>
```

- The Name attribute tells the XAML parser to add a field with the same name to the auto generated portion of the class, also known as the code behind.

```
private System.Windows.Controls.Grid TheRootLayout;
```

Properties and Events

- The XAML file includes the usage of attributes translated into properties and events in the code behind.
- Every XAML code can be replaced with a set of code statements that perform the same.

Complex Properties

- In some cases we can use the property-element syntax. The child element is added with a name in the form of Parent.PropertyName.

```
<Grid x:Name="grid1">
    <Grid.Background>
    ...
    </Grid.Background>
...
</Grid>
```

Attached Properties

- May apply to several elements but defined in a different class. The attached properties are frequently used to control the layout of our user interface. They are translated into methods calls.

```
<TextBox ... Grid.Row="0">
</TextBox>
...
<Button ... Grid.Row="1">
</Button>
```

Nesting Elements

- XAML allows each element to deal with its nested elements in a different way.
- One possible way is having the parent implementing `IList<T>` or `IDictionary<T>`.

Events

- These are attributes mapped to events. We assign these attributes with names of functions.

```
<Button ... Click="bt_Click">
```

XAML Resources

- The Silverlight platform includes a resources system closely integrated with the XAML code.
- Each element includes the Resources property that stores a collection of resources as key value pairs.

XAML Resources

```
<UserControl x:Class="EightBall.MainPage" ... >
    <UserControl.Resources>
        <LinearGradientBrush x:Key="BackgroundNice">
            <LinearGradientBrush.GradientStops>
                <GradientStop Offset="0.00" Color="Green" />
                <GradientStop Offset="0.70" Color="Red" />
                <GradientStop Offset="1.00" Color="Yellow" />
            </LinearGradientBrush.GradientStops>
        </LinearGradientBrush>
    </UserControl.Resources>
    ...
</UserControl>
```

XAML Resources

- We can now use this resource in our XAML. We should use the following special syntax.

```
<Grid x:Name="grid1" Background="{StaticResource BackgroundNice}">
```

Unlike WPF, Silverlight supports Static Resources Only!

Sample

```
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             x:Class="SilverlightApplication6.App"
             >
    <Application.Resources>
        <LinearGradientBrush x:Key="mybackground">
            <LinearGradientBrush.GradientStops>
                <GradientStop Offset="0.00" Color="Black" />
                <GradientStop Offset="0.50" Color="Purple" />
                <GradientStop Offset="1.00" Color="Red" />
            </LinearGradientBrush.GradientStops>
        </LinearGradientBrush>
    </Application.Resources>
</Application>
```

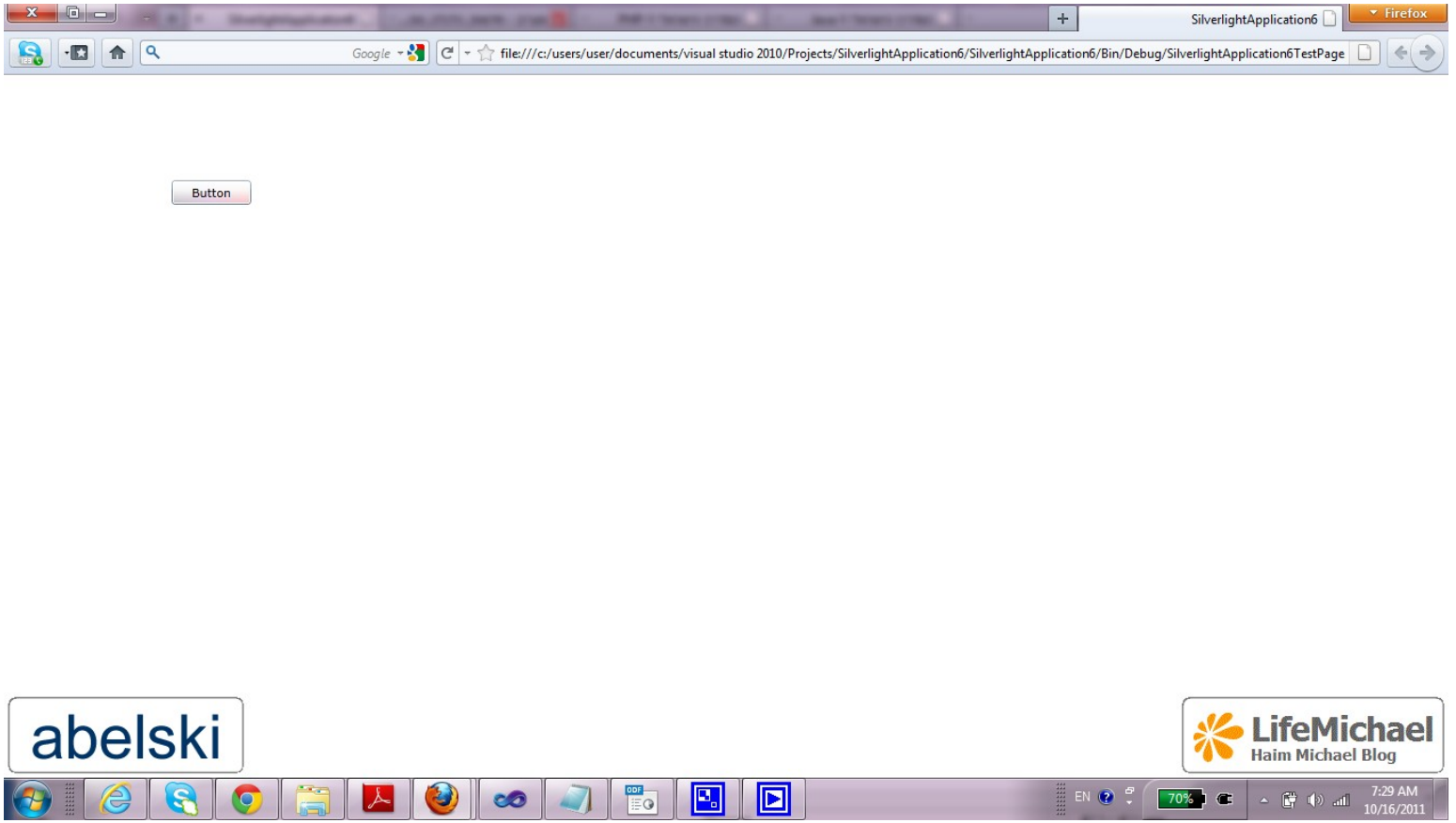
Sample

```
<UserControl x:Class="SilverlightApplication6.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 Content="Button" Height="23"
                HorizontalAlignment="Left"
                Margin="158,99,0,0" Name="button1"
                VerticalAlignment="Top" Width="75"
                Background="{StaticResource mybackground}">

        </Button>
    </Grid>
</UserControl>
```

Sample



Accessing Resources

- It is possible to access the resources from within the code itself.

```
LinearGradientBrush brush = (LinearGradientBrush)this.Resources["btfc"];
```

XAML Resources

- Silverlight starts the search for the required resource from the most inner element and ends checking the `<Application.Resources>` section of the `App.xaml` file.

Resource Dictionaries

- We can organize the resources into resource dictionaries. Resource dictionary is a simple XAML document.
- We create a resource dictionary by right clicking the project we develop and selecting `Add` → `New Item` → `Silverlight Resource Dictionary Template`.

Binding

```
<UserControl x:Class="SilverlightApplication7.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 Height="23" HorizontalAlignment="Left" Margin="178,94,0,0"
            Name="textBlock1" Text="GoGonGa Bonga" VerticalAlignment="Top"
            FontSize="{Binding ElementName=slider, Path=Value}" />
        <Slider Minimum="10" Maximum="100" Height="23"
            HorizontalAlignment="Left" Margin="156,185,0,0" Name="slider"
            VerticalAlignment="Top" Width="200" />
    </Grid>
</UserControl>
```