Silverlight 4

پژوهش و نگارش: وحید نصیری
Vahid_Nasiri@Yahoo.Com
بهار ۱۳۸۹

فهرست مطالب

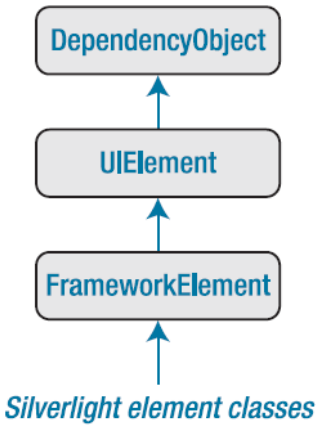
فصل ۵ – آشنایی با نحوه‌ی مدیریت رویدادها در Silverlight

مقدمه

پشتیبانی از رویدادها از نگارش‌های اولیه‌ی .NET Framework وجود داشته است. توسط رویدادها پیغامی از یک نوع، به نوعی دیگر ارسال می‌شود؛ برای مثال ارسال پیغامی از یک دکمه به فرم در برگیرنده‌ی آن. سیستم مدیریت رویدادها در WPF و Silverlight با رویه‌ی متداول موجود در WinForms اندکی متفاوت است. در Silverlight هر شیء می‌تواند از چندین زیر شیء نیز تشکیل شود و ساخت کنترل‌های ترکیبی در این سیستم به سادگی میسر است. برای مثال یک دکمه را در Silverlight در نظر بگیرید که داخل آن یک Border ، داخل Border یک Grid و در داخل این Grid یک تصویر قرار گرفته است. اگر با سیستم متداول رویدادگردانی در WinForms بخواهیم این دکمه را مدیریت کنیم باید برای تک تک عناصر قرار گرفته در دکمه، روال رویداد گردان بنویسیم. همچنین باید در نظر داشت که در Silverlight ظاهر پیش فرض یک شیء را کاملا می‌توان توسط مفهومی به نام Templates تعویض نمود. به همین جهت در Silverlight برای مدیریت رویدادها در این حالت پیچیده، سیستم Routed events طراحی شده است. با کمک Routed events ، رویدادهای زیر شیءها به والدین خود نیز به صورت خودکار انتقال داده خواهند شد (bubbled events) و به این صورت دیگر نیازی نخواهد بود تا برای کلیه عناصر قرار گرفته در یک شیء، روال رویدادگردان تهیه کرد.

رویدادهای عناصر پایه‌ای Silverlight

المان‌های Silverlight مجموعه‌ی رخدادهای اصلی خود را از دو کلاس پایه‌ای UIElement و FrameworkElement به ارث می‌برند (شکل ۱). تنها المان‌هایی که از سیستم Event bubbling ذکر شده استفاده می‌کنند، مربوط به کلاس UIElement می‌باشند.



شکل ۱– سلسله مراتب المان‌های Silverlight

در کلاس UIElement ، فقط رخدادهای KeyDown ، KeyUp ، GotFocus ، LostFocus ، MouseLeftButtonDown ، MouseLeftButtonUp ، MouseMove و MouseWheel از نوع Routed evenets هستند و رخدادهای MouseEnter ، MouseLeave و LostMouseCapture رخدادهای معمولی به شمار می‌روند.

کلاس FrameworkElement تنها رویدادهای محدودی را به این مجموعه اضافه می‌نماید که هیچکدام از نوع Routed events نیستند : Loaded ، SizeChanged ، LayoutUpdated و BindingValidationError .

تعریف رویدادها در Silverlight

در Silverlight از دو طریق می‌توان نسبت به تعریف رویدادها اقدام نمود : با کمک کدهای XAML و یا کد نویسی در فایل‌های Code behind برنامه. در مثال بعد متدهای رویدادگردان برنامه در XAML معرفی شده‌اند :

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication23.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">
    <StackPanel x:Name="LayoutRoot"
       Background="LightBlue" Margin='20'>
        <TextBlock Text='Specify Event Handler in XAML'
                            Margin='20' />
        <Button Content='Add Handler, no x:Name'
```

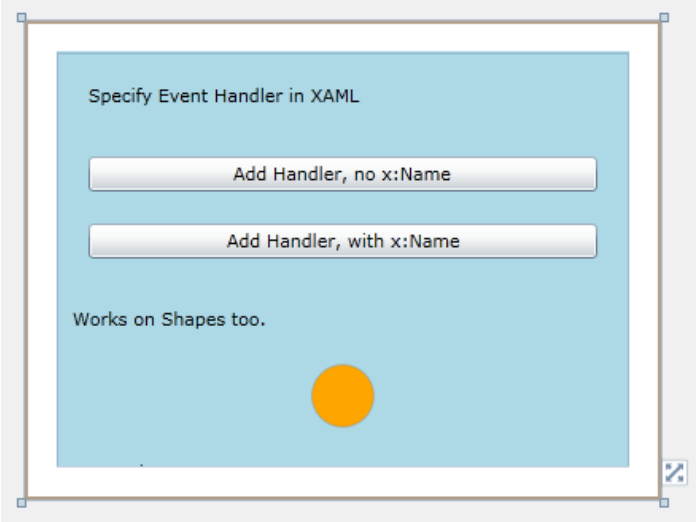
```
                Click="Button_Click"
              Margin='20,10' />
        <Button Content='Add Handler, with x:Name'
              Margin='20,10'
             Click="DemoButton1_Click"
              x:Name='DemoButton1' />
        <TextBlock Margin='10,20'
               Text='Works on Shapes too.' />
        <Ellipse Width='40'
               Height='40'
               Fill='Orange'
              MouseLeftButtonUp="DemoShape_MouseLeftButtonUp"
               x:Name='DemoShape'
               />
       <TextBlock Margin='10,20'  x:Name='resultTextBlock'
        Text='Output here...' />
    </StackPanel>
</UserControl>
```

و کدهای متناظر با این روال‌های رخدادگردان صفحه به شرح بعد می‌توانند باشند:

MainPage.xaml .cs

```
using System.Windows;
using System.Windows.Input;

namespace SilverlightApplication23
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            resultTextBlock.Text = "Button";
        }

        private void DemoButton1_Click(object sender, RoutedEventArgs e)
        {
            resultTextBlock.Text = "DemoButton";
        }

        private void DemoShape_MouseLeftButtonUp(object sender,
             MouseButtonEventArgs e)
        {
```

```
            resultTextBlock.Text = "DemoShape";
        }
    }
}
```



شکل ۲– نمایی از مثال اول برنامه‌ی فصل

در این مثال (شکل ۲) دکمه‌ی اول بدون نام بوده و نام دکمه‌ی دوم به صورت صریح ذکر شده است. بنابراین اگر بر روی هر کدام از دکمه‌ها دوبار کلیک نمائیم تا روال رویدادگردان Click آن‌ها به صورت خودکار ایجاد شود، نام روال رویداد گردان کلیک دکمه‌ای که دارای نام می‌باشد، در ابتدای نام این روال ذکر خواهد شد؛ در غیر اینصورت یک نام عمومی برای آن انتخاب می‌گردد. در ادامه همچنین برای شیء دایره‌ی ترسیم شده نیز روال رویدادگردان معرفی شده است.

در مثال بعد قصد داریم معرفی و تعریف روال‌های رویدادگردان را صرفا در کدهای برنامه انجام دهیم:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication24.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">
    <StackPanel x:Name="LayoutRoot"
       Background="LightGreen" Margin='20'>
        <TextBlock Text='Specify Event Handler in Code Behind'
```

```
                Margin='20' />
        <Button Content='Add Handler in code behind file.'
                                        Margin='20,10'
                                        x:Name='demoButton1' />
        <TextBlock Margin='10,20' Text='Works on Shapes too.' />
        <Ellipse Width='40'
                Height='40'
                Fill='Orange'
                x:Name='demoShape' />
        <TextBlock Margin='10,20' x:Name='resultTextBlock'
                Text='Output here...' />
    </StackPanel>
</UserControl>
```

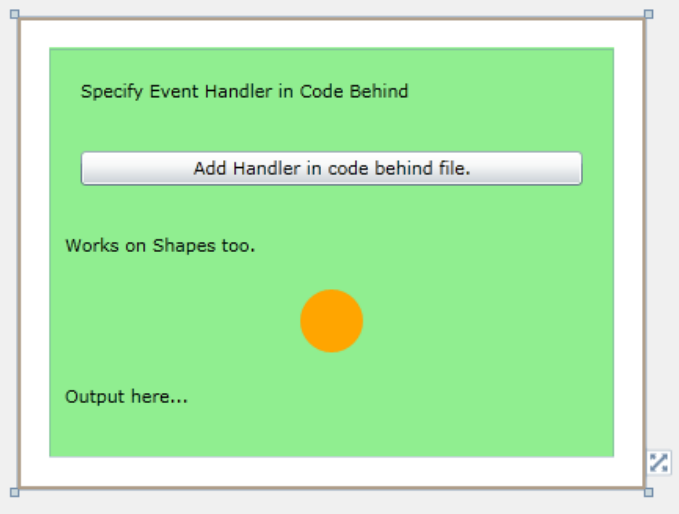
کدهای متناظر با این صفحه که در آن معرفی و مدیریت روال‌های رویدادگردان صورت گرفته است به شرح بعد می‌باشند:

MainPage.xaml .cs

```
using System.Windows;
using System.Windows.Input;

namespace SilverlightApplication24
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            this.Loaded += InCodeBehind_Loaded;
        }

        void InCodeBehind_Loaded(object sender, RoutedEventArgs e)
        {
            demoShape.MouseLeftButtonUp += shape_Handler;
            //or ...
            demoShape.AddHandler(MouseLeftButtonUpEvent,
                            new MouseButtonEventHandler(shape_Handler),
                            false);

            demoButton1.Click += demoButton1_Click;
        }

        void demoButton1_Click(object sender, RoutedEventArgs e)
        {
            resultTextBlock.Text = "Button.";
        }
```

```
        void shape_Handler(object sender, MouseButtonEventArgs e)
        {
            resultTextBlock.Text = "Shape.";
        }
    }
}
```

در این مثال (شکل ۳) ابتدا روال رویداد گردان Loaded مدیریت شده است و سپس توسط آن روال‌های رویدادگردان مرتبط با دکمه و دایره معرفی گشته‌اند. جهت معرفی روال رویدادگردان دایره، دو روش ارائه شده است که هر یک از آن‌ها به تنهایی کافی می‌باشد و روش دوم صرفا جهت آشنایی با نحوه‌ی دیگر معرفی Routed events است.



شکل ۳– نمایی از مثال دوم برنامه‌ی فصل

روش سومی نیز برای معرفی روال‌های رویدادگردان در WPF و Silverlight جهت بکارگیری در الگوی طراحی MVVM (Model-View-ViewModel) مرسوم است که از اشیاء Commands در آن استفاده می‌شود. با این روش در طی فصل‌های آتی بیشتر آشنا خواهیم شد.

درک مفهوم Event Bubbling

در ادامه سه مثال را در زمینه‌ی انتشار رخدادها از اشیاء فرزند در Silverlight به اشیاء والد، بررسی خواهیم نمود.

مثال اول

در مثال اول یک شیء پیچیده را به صورت زیر ایجاد کرده‌ایم (شکل ۴):

Grid → Button → Border → StackPanel → Image + TextBlock

برای استفاده از تصاویر، یک پوشه به نام images به پروژه جاری اضافه گردیده است و سپس تصویر information.png را به آن افزوده‌ایم (کلیک راست بر روی پوشه، انتخاب گزینه‌ی Add → Existing Item...) و انتخاب فایل تصویری کپی شده در این پوشه. اگر به خواص این تصویر مراجعه نمائیم، خاصیت Build Action آن به Resource تنظیم شده است).

سپس یک ListBox را نیز جهت نمایش انتشار رویدادها از اشیاء فرزند به والد در پایین صفحه قرار داده‌ایم. کدهای XAML این مثال را در ادامه مشاهده خواهید کرد:

MainPage.xaml

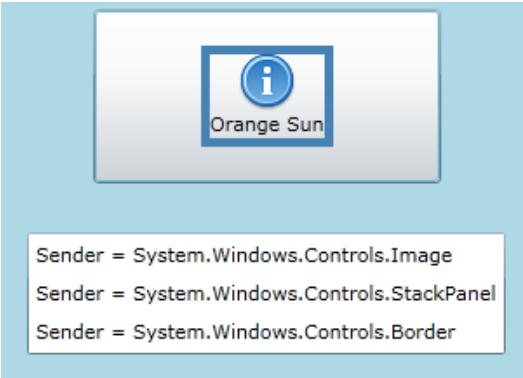
```
<UserControl x:Class="SilverlightApplication25.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
       "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot"
                           Background="LightBlue">
        <Grid.RowDefinitions>
            <RowDefinition Height="0.382*" />
            <RowDefinition Height="0.618*" />
        </Grid.RowDefinitions>
        <Button x:Name="DemoButton"
               Width='200' Height='100'  Margin="0">
            <Border x:Name="DemoBorder" BorderThickness='5'
                      BorderBrush='SteelBlue'>
                <StackPanel x:Name="DemoStackPanel">
                    <Image x:Name="DemoImage"
                              Source='images/information.png'
                              Width="32" Height="32"  />
                  <TextBlock x:Name="DemoTextBlock"
                              HorizontalAlignment='Center'>Orange Sun</TextBlock>
                </StackPanel>
            </Border>
        </Button>
        <ListBox HorizontalAlignment="Center"
                Margin="20"
                VerticalAlignment="Top"
                Grid.Row="1"
                x:Name='listResults' />
    </Grid>
</UserControl>
```

در کدهای این صفحه، رویداد MouseLeftButtonDown کلیه عناصر تعریف شده، دریافت و سپس ارسال کننده‌ی هر رویداد (Sender) به ListBox اضافه خواهد شد. همچنین اگر بر روی ListBox کلیک نمائیم، محتویات آن حذف خواهد شد.

برای آزمایش و درک بهتر انتشار رخدادها از اشیاء فرزند به والد، ابتدا بر روی حاشیه‌ی تعریف شده کلیک نمائید، سپس بر روی تصویر و سایر موارد کلیک کنید تا بهتر بتوان مفهوم Event bubbling را درک کرد.

MainPage.xaml .cs

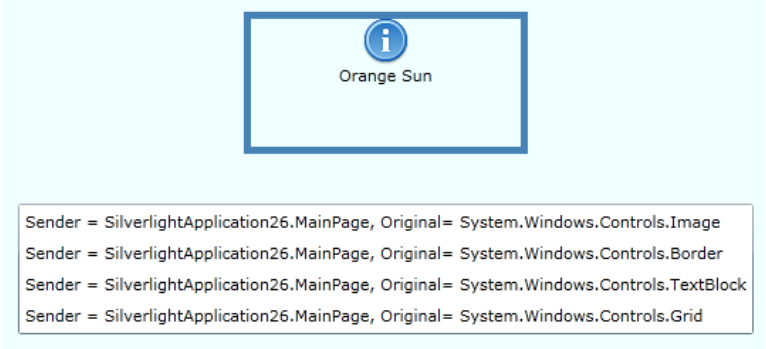
```
using System.Windows.Input;

namespace SilverlightApplication25
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();

            this.MouseLeftButtonDown += CommonMouseDownCode;
            DemoBorder.MouseLeftButtonDown += CommonMouseDownCode;
            DemoButton.MouseLeftButtonDown += CommonMouseDownCode;
            DemoImage.MouseLeftButtonDown += CommonMouseDownCode;
            DemoStackPanel.MouseLeftButtonDown += CommonMouseDownCode;
            DemoTextBlock.MouseLeftButtonDown += CommonMouseDownCode;

            listResults.MouseLeftButtonUp +=
                        listResults_MouseLeftButtonUp;
        }

        void listResults_MouseLeftButtonUp(object sender,
              MouseButtonEventArgs e)
        {
            listResults.Items.Clear();
        }

        void CommonMouseDownCode(object sender, MouseButtonEventArgs e)
        {
            string whatHappened = string.Format("Sender = {0}", sender);
            listResults.Items.Add(whatHappened);
        }
    }
}
```



شکل ۴– نمایی از مثال اول بررسی Event Bubbling پس از کلیک بر روی تصویر

مثال دوم

در مثال دوم نمایش توانایی‌های سیستم Event Bubbling (شکل ۵)، مدیریت کننده‌ی رویدادها را بالاترین المان موجود در صفحه یعنی همان User control تعریف کرده‌ایم.



شکل ۵– نمایی از مثال اول بررسی Event Bubbling

کدهای XAML این مثال بسیار شبیه به مثال قبلی است؛ با این تفاوت که شیء پیچیده‌ی تشکیل شده به صورت زیر است:

Grid → Grid → Border → StackPanel → Image + TextBlock

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication26.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="Azure">
        <Grid.RowDefinitions>
            <RowDefinition Height="0.382*" />
            <RowDefinition Height="0.618*" />
        </Grid.RowDefinitions>

        <Grid x:Name="DemoGrid"
                    Width='200'
                    Height='100'
                    Margin="0">
            <Border x:Name="DemoBorder" BorderThickness='5'
                    BorderBrush='SteelBlue'>
                <StackPanel x:Name="DemoStackPanel">
                    <Image x:Name="DemoImage"
                                Source='images/information.png'
                                 Width='32' />
                <TextBlock x:Name="DemoTextBlock"
                                HorizontalAlignment='Center'>Orange Sun</TextBlock>
                </StackPanel>
            </Border>
        </Grid>
        <ListBox HorizontalAlignment="Center"
               Margin="20"
               VerticalAlignment="Top"
               Grid.Row="1"
               x:Name='listResults' />
    </Grid>
</UserControl>
```

در ادامه کدهای رویداد گردان متناظر با این صفحه را ملاحظه خواهید کرد. در این کدها this به شیء جاری و یا همان User control مورد استفاده اشاره می‌کند.

MainPage.xaml.cs

```
using System.Windows.Input;

namespace SilverlightApplication26
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            this.MouseLeftButtonDown +=
                     BubbleToRoot_MouseLeftButtonDown;
```

```
        }

        void BubbleToRoot_MouseLeftButtonDown(object sender,
                  MouseButtonEventArgs e)
        {
            string whatHappened = string.Format(
                "Sender = {0}, Original= {1}", sender,
                 e.OriginalSource);
            listResults.Items.Add(whatHappened);
        }
    }
}
```

پس از اجرای برنامه، بر روی اشیاء مختلف قرار گرفته در صفحه کلیک نمائید تا بتوان انتشار رویدادها از فرزندان به والد را مشاهده نمود. Sender همواره شیءایی است که رویداد مورد نظر را مدیریت می‌کند. در این مثال چون User control مدیریت کننده‌ی اصلی است، Sender نمایش داده شده همواره ثابت خواهد بود.

مثال سوم

در این مثال نحوه‌ی متوقف سازی انتشار رخدادها از فرزندان به والدین را بررسی خواهیم کرد.

کدهای XAML این مثال دقیقا همانند مثال دوم بررسی سیستم Event Bubbling است و از ذکر مجدد آن‌ها خودداری می‌شود. اما کدهای متناظر با این فایل XAML در مثال سوم به شرح زیر می‌باشند:

MainPage.xaml.cs

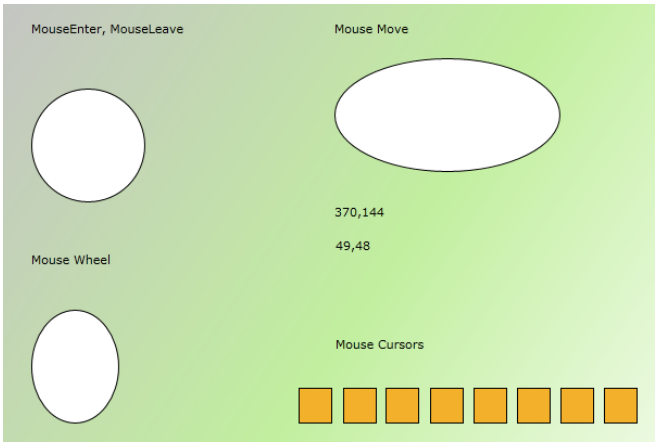
```
using System.Windows.Input;

namespace SilverlightApplication27
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();

            this.MouseLeftButtonDown +=
                      BubbleToRoot_MouseLeftButtonDown;
            DemoImage.MouseLeftButtonDown +=
                      DemoImage_MouseLeftButtonDown;
        }

        void DemoImage_MouseLeftButtonDown(object sender,
                  MouseButtonEventArgs e)
        {
            // stop bubbling
            e.Handled = true;
```

```
            listResults.Items.Add("Stopped");
        }

        void BubbleToRoot_MouseLeftButtonDown(object sender,
               MouseButtonEventArgs e)
        {

            string whatHappened = string.Format(
                "Sender = {0}, Original= {1}", sender,
                   e.OriginalSource);
            listResults.Items.Add(whatHappened);
        }
    }
}
```

در این مثال زمانیکه بر روی تصویر کلیک می‌شود، با مقدار دهی e.Handled به true کار مدیریت رخدادها به پایان می‌رسد و دیگر متد BubbleToRoot_MouseLeftButtonDown فراخوانی نخواهد شد. به این صورت از انتشار رخدادها به والد جلوگیری خواهد شد.

مدیریت رویدادهای Mouse در Silverlight

تعدادی رویداد ویژه مخصوص کار با Mouse در Silverlight وجود دارند که در مثال ذیل (شکل ۶) قصد بررسی آن‌ها را داریم:



شکل ۶– نمایی از برنامه‌ی مدیریت رویدادهای Mouse در Silverlight

کدهای XAML این مثال به شرح زیر هستند. در این مثال از سیستم طرح بندی Canvas استفاده گردیده است و توسط آن یک سری بیضی و مستطیل در مکان‌های ثابتی در صفحه قرار گرفته‌اند. هدف از مستطیل‌های تعریف شده، نمایش انواع و اقسام اشکال مکان‌نمای Mouse است.

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication28.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="480" d:DesignWidth="640">
    <Canvas x:Name="LayoutRoot">
        <Canvas.Background>
            <LinearGradientBrush EndPoint="1.012,1"
                       StartPoint="0.019,0.041">
                <GradientStop Color="#FFC5C5C5"
                      Offset="0" />
                <GradientStop Color="White"
                      Offset="1" />
                <GradientStop Color="#FFC3F0A0"
                      Offset="0.513" />
            </LinearGradientBrush>
        </Canvas.Background>
        <Ellipse Fill="White"
                Stroke="Black"
                Height="105" Width="105"
                Canvas.Left="42" Canvas.Top="124"
                 MouseEnter="Ellipse_MouseEnter"
                 MouseLeave='Ellipse_MouseLeave' />
        <TextBlock Height="17" Width="198"
                Canvas.Left="42" Canvas.Top="61"
                Text="MouseEnter, MouseLeave"
                TextWrapping="Wrap" />
        <Ellipse Fill="White"
                x:Name='moveEllipse'
                Stroke="Black"
                Height="105" Width="208"
                Canvas.Left="321" Canvas.Top="96"
                 MouseMove='Ellipse_MouseMove' />
        <TextBlock Height="31"
                Width="198"
                Canvas.Left="321" Canvas.Top="61"
                Text="Mouse Move"
                TextWrapping="Wrap" />
        <TextBlock x:Name="textBlockResults"
```

```
        Height="31" Width="198"
        Canvas.Left="321" Canvas.Top="230"
        Text="Move results here..."
        TextWrapping="Wrap" />
    <TextBlock x:Name="textBlockResults2"
        Height="31" Width="198"
        Canvas.Left="322" Canvas.Top="261"
        Text="Move results here..."
        TextWrapping="Wrap" />
    <Ellipse Fill="White"
        x:Name='wheelEllipse'
        Stroke="Black"
        Height="105" Width="105"
        Canvas.Left="42" Canvas.Top="328"
         MouseWheel='wheelEllipse_MouseWheel' />
    <TextBlock Height="31"
        Width="198"
        Canvas.Left="42" Canvas.Top="274"
        Text="Mouse Wheel"
        TextWrapping="Wrap" />
    <TextBlock Height="31"
        Width="198"
        Canvas.Left="322" Canvas.Top="352"
        Text="Mouse Cursors"
        TextWrapping="Wrap" />
    <Rectangle Fill="#FFF3B12B"
        Stroke="Black"
        Height="33" Width="31"
        Canvas.Left="329" Canvas.Top="400"
        Cursor="Arrow" />
    <Rectangle Fill="#FFF3B12B"
        Stroke="Black"
        Height="33" Width="31"
        Canvas.Left="368" Canvas.Top="400"
        Cursor="Eraser" />
    <Rectangle Fill="#FFF3B12B"
        Stroke="Black"
        Height="33" Width="31"
        Canvas.Left="409" Canvas.Top="400"
        Cursor="Hand" />
    <Rectangle Fill="#FFF3B12B"
        Stroke="Black"
        Height="33" Width="31"
        Canvas.Left="449" Canvas.Top="400"
        Cursor="IBeam" />
    <Rectangle Fill="#FFF3B12B"
        Stroke="Black"
        Height="33" Width="31"
```

```
                Canvas.Left="489" Canvas.Top="400"
            Cursor="SizeNS" />
        <Rectangle Fill="#FFF3B12B"
            Stroke="Black"
            Height="33" Width="31"
            Canvas.Left="529" Canvas.Top="400"
            Cursor="SizeWE" />
        <Rectangle Fill="#FFF3B12B"
            Stroke="Black"
            Height="33" Width="31"
            Canvas.Left="569" Canvas.Top="400"
            Cursor="Stylus" />
        <Rectangle Fill="#FFF3B12B"
            Stroke="Black"
            Height="33" Width="31"
            Canvas.Left="288" Canvas.Top="400"
            Cursor="Wait" />
    </Canvas>
</UserControl>
```

و کدهای متناظر با این صفحه را در ادامه ملاحظه می‌فرمائید:

MainPage.xaml.cs

```
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;

namespace SilverlightApplication28
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void Ellipse_MouseEnter(object sender, MouseEventArgs e)
        {
            var ellipse = sender as Ellipse;
            if (ellipse != null) ellipse.Fill =
                        new SolidColorBrush(Colors.Orange);
            // MouseEventArgs contains mouse details.
        }

        private void Ellipse_MouseLeave(object sender, MouseEventArgs e)
        {
            var ellipse = sender as Ellipse;
```

```
            if (ellipse != null) ellipse.Fill =
                 new SolidColorBrush(Colors.White);
        }

        private void Ellipse_MouseMove(object sender, MouseEventArgs e)
        {
            // get the mouse location relative to UserControl (this)
            var point = e.GetPosition(this);
            textBlockResults.Text =
                 string.Format("{0},{1}", point.X, point.Y);

            // get the mouse location relative to Ellipse
            var point2 = e.GetPosition(moveEllipse);
            textBlockResults2.Text =
               string.Format("{0},{1}", point2.X, point2.Y);
        }

        private void wheelEllipse_MouseWheel(object sender,
                   MouseWheelEventArgs e)
        {
            // wheel event only works on Internet Explorer on Windows
            wheelEllipse.Width += e.Delta / 10;
        }
    }
}
```

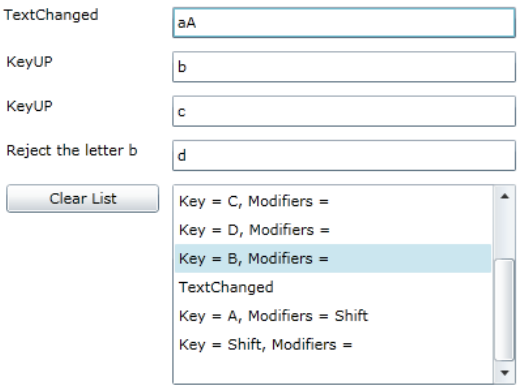
توضیحات:

هدف از این مثال بررسی رخدادهای MouseEnter ، MouseLeave ، MouseMove و MouseWheel می‌باشد.

در روال‌های رخدادگردان MouseEnter و MouseLeave، با پردازش Sender ، می‌توان به شیء بیضی مانندی که Mouse بر روی آن قرار گرفته است یا از محدوده‌ی آن خارج شده است، دست یافت. به این ترتیب برای مثال با دسترسی به آن شیء، رنگ آن را می‌توان تغییر داد. در روال رخدادگردان MouseMove ، با کمک آرگومان MouseEventArgs دریافتی و استفاده از متد GetPosition آن، امکان نمایش مختصات مکان نمای Mouse وجود خواهد داشت. رخداد مربوط به حرکت دکمه‌ی میانی Mouse و یا همان MouseWheel ، تنها در ویندوز و در Internet Explorer پشتیبانی می‌شود. بنابراین باید به این موضوع دقت داشت. در متد رخدادگردان آن با استفاده از آرگومان MouseWheelEventArgs ، جهت حرکت دکمه‌ی میانی مشخص می‌شود که در آن e.Delta همواره مضربی مثبت یا منفی از ۱۲۰ است.

مدیریت رویدادهای صفحه کلید در Silverlight

در طی یک مثال قصد داریم نوع کلیدهای فشرده شده را در یک ListBox نمایش دهیم و همچنین امکان فیلتر کردن یک سری از حروف را نیز به برنامه اضافه کنیم (شکل ۷).



شکل ۷– نمایی از مثال مدیریت رویدادهای صفحه کلید

کدهای XAML تشکیل دهنده‌ی رابط کاربر برنامه به شرح زیر هستند:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication29.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"
          MaxWidth="400" MaxHeight="300"
                KeyUp='LayoutRoot_KeyUp'>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.317*" />
            <ColumnDefinition Width="0.683*" />
```

```
        </Grid.ColumnDefinitions>
        <TextBlock Margin="3"
                Text="TextChanged"
                TextWrapping="Wrap" />
        <TextBlock Margin="5"
                Text="KeyUP"
                TextWrapping="Wrap"
                Grid.Row="1" />
        <TextBlock Margin="5"
                Text="KeyUP"
                TextWrapping="Wrap"
                Grid.Row="2" />
        <TextBlock Margin="5"
                Text="Reject the letter b"
                TextWrapping="Wrap"
                Grid.Row="3" />
        <Button Margin="5" VerticalAlignment="Top"
               Grid.Row="4"
               Content="Clear List"
               Click='Button_Click' />
        <ListBox Margin="5"
                Grid.Column="1"
                Grid.Row="4"
                x:Name='listResults' />
        <TextBox Margin="5"
                Grid.Column="1"
                Text=""
                TextWrapping="Wrap"
               TextChanged='TextBox_TextChanged' />
        <TextBox Margin="5"
                Grid.Column="1"
                Text=""
                TextWrapping="Wrap"
                Grid.Row="1" />
        <TextBox Margin="5"
                Grid.Column="1"
                Text=""
                TextWrapping="Wrap"
                Grid.Row="2" />
        <TextBox Margin="5"
                Grid.Column="1"
                Text=""
                TextWrapping="Wrap"
                Grid.Row="3"
                KeyDown='TextBox_KeyDown' />
    </Grid>
</UserControl>
```

و در ادامه کدهای متناظر با صفحه‌ی فوق به شرح بعد می‌باشند:

MainPage.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace SilverlightApplication29
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            listResults.Items.Clear();
        }

        private void LayoutRoot_KeyUp(object sender, KeyEventArgs e)
        {
            //  ALT key is used by IE
            string modifiers = null;

            if ((Keyboard.Modifiers & ModifierKeys.Control) ==
                  ModifierKeys.Control)
            {
                modifiers += "Ctrl |";
            }

            if ((Keyboard.Modifiers & ModifierKeys.Shift) ==
                  ModifierKeys.Shift)
            {
                modifiers += "Shift";
            }

            listResults.Items.Add(string.Format(
                "Key = {0}, Modifiers = {1}", e.Key, modifiers));
        }

        private void TextBox_TextChanged(object sender,
               TextChangedEventArgs e)
        {
            listResults.Items.Add("TextChanged");
        }
```

```
        private void TextBox_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.B)
            {
                // reject this key and do not show in textbox
                e.Handled = true;
            }
        }
    }
}
```

توضیحات:

در این مثال تنها در TextBox اول، رخداد TextChanged معرفی شده است و دو TextBox بعدی هیچگونه روال رخدادگردانی ندارند. اما اگر برنامه را اجرا کنیم، نوع کلیدهای فشرده شده در این سه TextBox نیز در ListBox پایین صفحه نمایش داده می‌شود. علت این امر به تعریف و مدیریت رخداد KeyUp در سطح Grid بر می‌گردد:

```
    <Grid x:Name="LayoutRoot"
          MaxWidth="400" MaxHeight="300"
                KeyUp='LayoutRoot_KeyUp'>
```

به این صورت رویدادهای KeyUp کلیه عناصر داخل گرید، به والد خود (یعنی Grid در اینجا)، سرایت کرده و مدیریت خواهند شد (یکی دیگر از کاربردهای Event Bubbling).

در آخرین TextBox تعریف شده به صورت ویژه‌ای رخداد KeyDown تعریف و مدیریت خواهد شد. در روال رخداد گردان آن اگر کلید b توسط آرگومان KeyEventArgs دریافت شود، با تنظیم e.Handled به مقدار True، از حرف b صرفنظر گردیده و نمایش داده نخواهد شد. این مورد برای ساخت TextBox های ویژه مانند TextBox عددی (که کاربران تنها امکان ورود اعداد را در آن دارند) بسیار مفید است.

نکته:

TabIndex تمام کنترل‌ها در صفحه به عدد ۱ تنظیم شده‌اند. بنابراین با تنظیم TabIndex یکی از آن‌ها به صفر، این کنترل در زمان آغاز فرم، Focus را در اختیار خواهد داشت. در غیر اینصورت مدیریت TabIndex ها و استفاده از Tab جهت مراجعه به کنترل بعدی توسط Silverlight مدیریت خواهد شد و همواره کاربر به نزدیکترین کنترل بعدی هدایت می‌گردد. در صورت نیاز می‌توان TabIndex کنترل‌ها را مقدار دهی نمود.