

Silverlight 4

فهرست مطالب

فصل ۱۹ - معرفی سایر امکانات و ویژگی‌های اختصاصی Silverlight 4	۴۱۷
مقدمه	۴۱۷
کنترل RichTextBox	۴۱۷
استفاده از Webcam و Microphone	۴۲۰
امکانات FluidUI	۴۲۶
کنترل کلیک راست Mouse	۴۲۸
استفاده از امکانات چاپگر	۴۳۰
دسترسی به Clipboard	۴۳۳
کنترل ViewBox	۴۳۶

چاپ عمومی غیر رایگان این مطالب بدون مجوز کتبی از طرف نویسنده به هر نحوی غیرمجاز است.
انتشار این مطالب بر روی اینترنت و یا استفاده از آن به صورت مستقیم و یا غیر مستقیم در نشریات الکترونیکی با ذکر مأخذ بلا مانع است.

فصل ۱۹ – معرفی سایر امکانات و ویژگی‌های اختصاصی Silverlight 4

مقدمه

تاکنون در لابلای فصول قبلی کتاب جاری بسیاری از ویژگی‌های جدید Silverlight 4 معرفی شدند؛ مانند: پشتیبانی از زبان‌های راست به چپ مانند زبان فارسی، امکانات جدید اعتبار سنجی، امکانات مهیا شده برای پیاده سازی بهتر الگوی MVVM، پشتیبانی از کشیدن و رها کردن (drag & drop)، بهبودهای حاصل در عملیات Binding، قالب‌های جدید، Implicit Style Manager و غیره. در این فصل قصد داریم سایر ویژگی‌های جدید این نگارش از Silverlight را بررسی نمائیم. البته لازم به ذکر است که معرفی قابلیت‌های اختصاصی Silverlight 4 در همین فصل پایان نیافته و یک سری از قابلیت‌های جدید دیگر آن مختص برنامه‌هایی از Silverlight است که جهت اجرا در خارج از مرورگر طراحی و تهیه شده‌اند؛ که در طی فصل‌های آتی به آن‌ها خواهیم پرداخت.

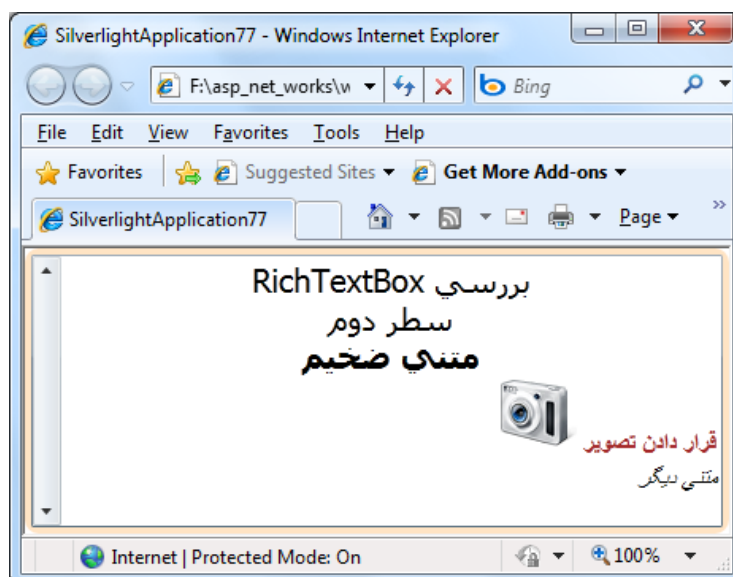
کنترل RichTextBox

جهت نمایش متونی با امکان قالب بندی ویژه آن‌ها و یا فراهم آوردن یک جعبه‌ی متنی مانند ویرایشگرهای حرفه‌ای متن، کنترل جدیدی به Silverlight 4 اضافه شده است به نام RichTextBox. این کنترل در دو حالت پیش فرض و قابل ویرایش (همانند یک TextBox معمولی) و حالت فقط خواندنی (همانند مثال بعد با تنظیم خاصیت IsReadOnly به True)، قابل استفاده است. در نگارش‌های قبلی Silverlight تنها اشیاء Run و LineBreak جهت تهیه‌ی متنی با قالب بندی مهیا بودند، اما کنترل RichTextBox این امکانات را توسعه داده است. اشیاء Span برای اعمال قالبی ویژه به متن، اشیاء Bold، Italic، Underline و Hyperlink نیز توسط این کنترل ارائه شده‌اند که مثالی از کاربرد آن‌ها را در کدهای XAML مثال بعد مشاهده خواهید نمود. علاوه بر آن امکان استفاده‌ی تو در تو از این اشیاء نیز مهیا است. برای مثال نمایش ضخیم یک سطر که قسمتی از آن Italic شده است به کمک بکارگیری تو در توی این امکانات میسر می‌باشد.

برای استفاده از کنترل‌های متداول UI در یک RichTextBox، از شیء InlineUIContainer باید کمک گرفته شود. مثالی از آن در ادامه جهت قرار دادن تصویری در متن ذکر گردیده است.

کنترل RichTextBox از شیء‌ایی به نام Paragraph برای معرفی متون قسمت‌های مختلف خود استفاده می‌کند. علاوه بر آن خاصیت Xaml این کنترل امکان انتساب یک متن با فرمت Xaml را نیز فراهم می‌کند. کاربرد

این خاصیت (برای مثال MyRichTxt1.Xaml) می‌تواند دریافت و سپس ذخیره سازی متن قالب بندی شده در بانک اطلاعاتی جهت بازیابی و نمایش‌های بعدی باشد.



شکل ۱- مثالی از توانایی‌های کنترل RichTextBox.

در ادامه مثالی از کاربرد مفاهیم عنوان شده را در کدهای XAML بعد ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication77.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">
        <Border BorderThickness="4" BorderBrush="Bisque"
            CornerRadius="7">
            <RichTextBox
                Name="rchTxt"
                IsReadOnly="True" TextWrapping="Wrap"
                VerticalScrollBarVisibility="Visible"
                FlowDirection="RightToLeft">
                <Paragraph TextAlignment="Center"
                    FontSize="20"
                    FontFamily="Tahoma">بررسی RichTextBox
                <LineBreak/>
                سطر دوم
            <LineBreak/>
        </Border>
    </Grid>
</UserControl>
```

```

        <Bold>متنی ضخیم</Bold>
    </Paragraph>
    <Paragraph TextAlignment="Left">
        <Span FontSize="15" Foreground="Brown"
            FontWeight="Bold">
            قرار دادن تصویر
        </Span>
        <InlineUIContainer>
            <Image Stretch="None"
                Source="Images/Camera.png" />
        </InlineUIContainer>
        <LineBreak />
        <Run FontSize="15" FontStyle="Italic">
            متنی دیگر
        </Run>
    </Paragraph>
</RichTextBox>
</Border>
</Grid>
</UserControl>

```

در این مثال با تنظیم FlowDirection به RightToLeft که جزو ویژگی‌های جدید Silverlight 4 می‌باشد، RichTextBox را برای نمایش متون فارسی آماده نمودیم. خاصیت TextWrapping این کنترل نیز به Wrap تنظیم گردید تا متون طولانی را بتوان در سطرهاى بعدی ملاحظه نمود. اگر تعداد سطرهاى نمایشی این کنترل زیاد است بهتر است تا خاصیت VerticalScrollBarVisibility آن به Visible تنظیم گردد. این کنترل به صورت پیش فرض به همراه یک نوار ابزار، همانند ویرایشگرهای متنی حرفه‌ای ارائه نمی‌شود؛ اما بدیهی است کلیه اشیاء نامبرده شده را از طریق برنامه نویسی نیز می‌توان ایجاد نمود و هر کدام از اشیاء XAML، متناظر با یک شیء NET می‌باشند. برای مشاهده‌ی مثالی کامل در مورد پیاده سازی این نوار ابزار، لطفاً به آدرس ذیل مراجعه نمائید (برنامه‌ی Rich Notepad پیاده سازی شده با Silverlight 4):

<http://www.silverlight.net/community/samples/silverlight-4/rich-notepad/>

برای نمونه فرض کنید قصد داریم به کمک روال رخداد گردان یک دکمه، متن انتخابی یک RichTextBox را ضخیم کنیم. کدهای این روال به شرح بعد می‌باشند:

```

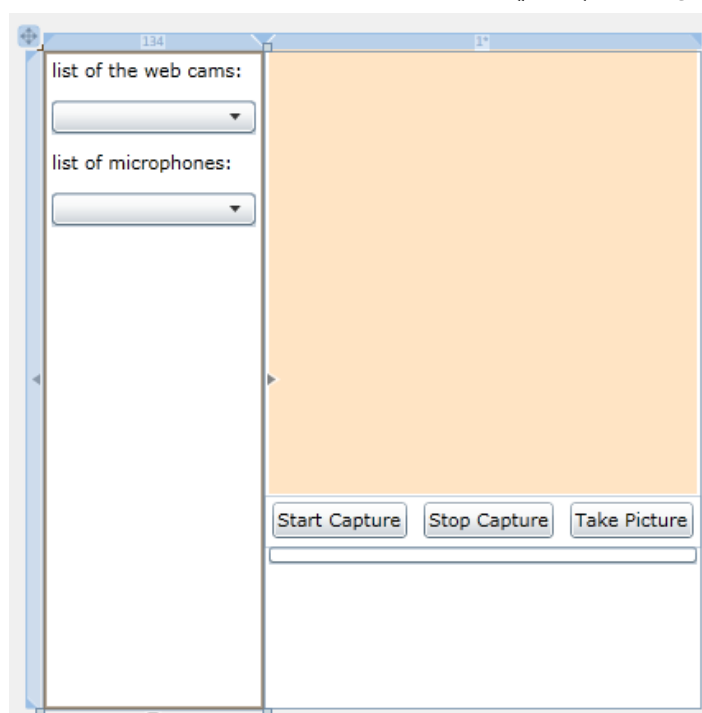
C#
private void Button_Click(object sender, RoutedEventArgs e)
{
    if (!String.IsNullOrEmpty(rchTxt.Selection.Text))
    {
        rchTxt.Selection.ApplyPropertyValue(
            TextElement.FontWeightProperty,
            FontWeights.Bold);
    }
}

```

در اینجا توسط متد ApplyPropertyValue، یک Dependency property متناظر با وزن قلم را باید مقدار دهی نمود.

استفاده از Webcam و Microphone

امکان استفاده از Webcam و همچنین microphone بر اساس تقاضاهای مکرر جامعه‌ی برنامه نویسان Silverlight به این مجموعه اضافه شده است. برای نمایش کاربردی استفاده از این امکانات (شکل ۲) لطفاً به کدهای بعد و توضیحات آن‌ها دقت بفرمائید.



شکل ۲- نمایشی از طراحی رابط کاربری کار با webcam و microphone.

کدهای XAML تشکیل دهنده‌ی رابط کاربری برنامه‌ی کار با webcam و microphone را در ادامه ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication78.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="400" d:DesignWidth="400">
```

```

<Grid x:Name="LayoutRoot" Background="White">
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="134" />
    <ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
<StackPanel Grid.Column="0">
    <TextBlock Text="list of the web cams:" Margin="5" />
    <ComboBox x:Name="WebCamList" Margin="5">
        <ComboBox.ItemTemplate>
            <DataTemplate>
                <TextBlock Text="{Binding FriendlyName}" />
            </DataTemplate>
        </ComboBox.ItemTemplate>
    </ComboBox>
    <TextBlock Text="list of microphones:" Margin="5" />
    <ComboBox x:Name="MicrophoneList" Margin="5">
        <ComboBox.ItemTemplate>
            <DataTemplate>
                <TextBlock Text="{Binding FriendlyName}" />
            </DataTemplate>
        </ComboBox.ItemTemplate>
    </ComboBox>
</StackPanel>

<StackPanel Grid.Column="1" HorizontalAlignment="Left">
    <Rectangle x:Name="ViewBox"
        Width="260" Height="270" Fill="Bisque"/>
    <StackPanel Orientation="Horizontal"
        HorizontalAlignment="Center">
        <Button Margin="5" x:Name="btnStartCapture"
            Content="Start Capture"
            Click="btnStartCapture_Click" />
        <Button Margin="5" x:Name="btnStopCapture"
            Content="Stop Capture"
            Click="btnStopCapture_Click" />
        <Button Margin="5" x:Name="btnTakePicture"
            Content="Take Picture"
            Click="btnTakePicture_Click" />
    </StackPanel>

    <ScrollViewer Width="260"
        VerticalScrollBarVisibility="Hidden"
        HorizontalScrollBarVisibility="Auto">
        <ItemsControl x:Name="Pictures">
            <ItemsControl.ItemTemplate>
                <DataTemplate>
                    <Image Source="{Binding}"
                        Margin="5"
                        Stretch="UniformToFill"
                        Height="240" />
                </DataTemplate>
            </ItemsControl.ItemTemplate>
        </ItemsControl>
    </ScrollViewer>
</StackPanel>

```

```

        </DataTemplate>
    </ItemsControl.ItemTemplate>
    <ItemsControl.ItemsPanel>
        <ItemsPanelTemplate>
            <StackPanel Orientation="Horizontal"
                VerticalAlignment="Center"
                HorizontalAlignment="Center"/>
        </ItemsPanelTemplate>
    </ItemsControl.ItemsPanel>
</ItemsControl>
</ScrollViewer>
</StackPanel>
</Grid>
</UserControl>

```

در این کدها از دو ComboBox برای نمایش لیست webcams و microphones نصب شده در سیستم استفاده می‌گردد. همچنین سه دکمه برای شروع به کار با وسایل ذکر شده، توقف کار و تهیه عکس از تصویر جاری webcam در نظر گرفته شده است. در ذیل آن، تصاویر دریافتی نمایش داده خواهند شد. کدهای متناظر با این صفحه را در ادامه ملاحظه خواهید نمود:

MainPage.xaml.cs

```

using System.Collections.ObjectModel;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;

namespace SilverlightApplication78
{
    public partial class MainPage
    {
        #region Fields (2)

        CaptureSource _captureSource;
        readonly ObservableCollection<WriteableBitmap> _pictures =
            new ObservableCollection<WriteableBitmap>();

        #endregion Fields

        #region Constructors (1)

        public MainPage()
        {
            InitializeComponent();
            this.Loaded += MainPageLoaded;
        }

        #endregion Constructors
    }
}

```



```
#region Methods (6)

// Private Methods (6)

private void btnStartCapture_Click(object sender,
                                   RoutedEventArgs e)
{
    if (_captureSource == null) return;

    // stop whatever device may be capturing
    _captureSource.Stop();

    if (WebCamList.SelectedItem != null)
        _captureSource.VideoCaptureDevice =
            (VideoCaptureDevice)WebCamList.SelectedItem;

    if (MicrophoneList.SelectedItem != null)
        _captureSource.AudioCaptureDevice =
            (AudioCaptureDevice)MicrophoneList.SelectedItem;

    var vidBrush = new VideoBrush();
    vidBrush.SetSource(_captureSource);
    ViewBox.Fill = vidBrush;

    // request access to webcam and audio devices
    if (CaptureDeviceConfiguration.AllowedDeviceAccess
        || CaptureDeviceConfiguration.RequestDeviceAccess())
    {
        _captureSource.Start();
    }
}

private void btnStopCapture_Click(object sender,
                                   RoutedEventArgs e)
{
    if (_captureSource == null) return;
    _captureSource.Stop();
}

private void btnTakePicture_Click(object sender,
                                   RoutedEventArgs e)
{
    if (_captureSource == null) return;
    _captureSource.CaptureImageAsync();
}
```

```
private static void captureSourceCaptureFailed(object sender,
    ExceptionRoutedEventArgs e)
{
    MessageBox.Show(
        string.Format("Failed to capture image: {0}",
            e.ErrorException.Message));
}

private void captureSourceCaptureImageCompleted(object sender,
    CaptureImageCompletedEventArgs e)
{
    _pictures.Add(e.Result);
}

void MainPageLoaded(object sender, RoutedEventArgs e)
{
    // creating a new capture source
    _captureSource = new CaptureSource();

    // get list of the web cams
    WebCamList.ItemsSource =
        CaptureDeviceConfiguration.
            GetAvailableVideoCaptureDevices();
    if (WebCamList.Items.Count > 0)
        WebCamList.SelectedIndex = 0;

    // get list of microphones
    MicrophoneList.ItemsSource =
        CaptureDeviceConfiguration.
            GetAvailableAudioCaptureDevices();
    if (MicrophoneList.Items.Count > 0)
        MicrophoneList.SelectedIndex = 0;

    Pictures.ItemsSource = _pictures;

    // async capture failed event handler
    _captureSource.CaptureFailed +=
        captureSourceCaptureFailed;

    // async capture completed event handler
    _captureSource.CaptureImageCompleted +=
        captureSourceCaptureImageCompleted;
}

#endregion Methods
}
```

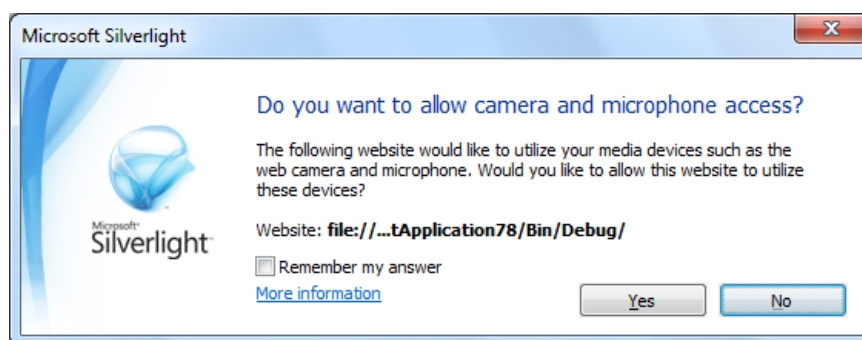
توضیحات:

کار اصلی این کلاس از متد `mainPageLoaded` شروع می‌شود. در اینجا ابتدا یک وهله از شیء `CaptureSource` جهت دسترسی به امکانات سخت افزارهای مورد نظر ایجاد می‌شود. در ادامه با کمک متدهای استاتیک کلاس `CaptureDeviceConfiguration`، لیست `webcams` و همچنین `microphones` نصب شده بر روی سیستم دریافت و از طریق عملیات `binding` در اختیار `Combox` های برنامه قرار خواهند گرفت. یک شیء `ObservableCollection` از نوع `WriteableBitmap` نیز برای دریافت تصاویر `webcam` و افزودن آن‌ها به `ItemsControl` ایی به نام `Pictures`، تعریف شده است. از آنجائیکه تهیه‌ی عملیات دریافت تصاویر از `webcam` به صورت غیرهمزمان است نیاز بود تا روال رخدادگردان پایان عملیات نیز تعریف گردد. تا اینجا آماده سازی ابتدایی رابط کاربر به پایان می‌رسد.

عملیات شروع به کار `webcam` و `microphone` توسط متد `btnStartCapture_Click` مدیریت می‌گردد. در اینجا اگر وسیله‌ای در حال استفاده است، ابتدا به کمک متد `Stop`، متوقف شده و سپس `VideoCaptureDevice` و `AudioCaptureDevice` های انتخاب شده توسط کاربر به شیء `_captureSource` برای تعاریف `webcam` و `microphone` مورد نظر، معرفی می‌گردند.

نکته‌ی جالب این مثال استفاده از یک `VideoBrush` برای نمایش تصاویر دریافتی از `webcam` است. توسط یک `VideoBrush` می‌توان هر نوع `UI Element` ایی را با محتوای ویدیویی و چند رسانه‌ای پوشش داد. در اینجا یک شیء `Rectangle` را با این قلم ویدیویی پوشش می‌دهیم.

پس تعاریف مقدماتی فوق، ابتدا باید از کاربر کسب مجوز کرد (شکل ۳). پس از آن مجوز فراخوانی متد `Start` را خواهیم داشت. متد `RequestDeviceAccess` این درخواست را ارائه داده و از خاصیت `AllowedDeviceAccess` جهت بررسی دسترسی احتمالی قبلی، برای عدم نمایش مکرر این صفحه کمک خواهیم جست.



شکل ۳- کسب مجوز از کاربر برای استفاده از `microphone` یا `webcam`.

برای دریافت تصاویر ابتدا متد `btnTakePicture_Click` فراخوانی شده و یک عملیات غیرهمزمان آغاز می‌گردد. سپس، پایان عملیات توسط متد `captureSourceCaptureImageCompleted` مشخص خواهد شد. `e.Result` دریافتی همان تصویر اخذ شده است (از نوع `Bitmap`).

در این مثال بجای دریافت webcam مورد نظر از کاربر، از webcam پیش فرض تعریف شده در سیستم به کمک متد ذیل نیز می‌شد جهت مقدار دهی VideoCaptureDevice استفاده کرد:

CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice()

علاوه بر این امکانات، یک پروژه‌ی سورس باز دیگر نیز در این زمینه در آدرس ذیل قابل دسترسی است:

Free Silverlight Voice/Video Conferencing Modules

<http://silverlightvideochat.codeplex.com/>

همانطور که پیشتر نیز ذکر شد، تصویر دریافتی در این حالت از نوع Bitmap می‌باشد. برای کاهش حجم تصاویر (برای مثال جهت ذخیره سازی آن‌ها) یک PNG Encoder سورس باز مخصوص Silverlight را از آدرس ذیل می‌توانید دریافت کنید:

<http://bit.ly/77mDsv>

امکانات FluidUI

FluidUI در Silverlight 4 امکان افزودن پویانمایی (Animation) را به رخدادهای load و unload هر یک از آیتم‌های یک ListBox، فراهم می‌سازد. مثالی در این مورد را در ادامه مشاهده خواهید نمود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication79.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">
    <UserControl.Resources>
        <DataTemplate x:Key="ListBoxDataTemplate">
            <Grid>
                <TextBlock Name="txt1" Text="{Binding}" />
            </Grid>
        </DataTemplate>

        <Style x:Key="ListBoxItemStyle1" TargetType="ListBoxItem">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="ListBoxItem">
                        <Grid Background="{TemplateBinding Background}">
                            <VisualStateManager.VisualStateGroups>
                                <VisualStateGroup x:Name="CommonStates">
```

```

        <VisualState x:Name="Loaded"/>
        <VisualState x:Name="BeforeLoaded">
            <Storyboard
                x:Name="FadeInOperate1">
                <DoubleAnimationUsingKeyFrames
                    Storyboard.TargetProperty="(UIElement.Opacity)"
                    Storyboard.TargetName="contentPresenter">
                    <SplineDoubleKeyFrame KeyTime="00:00:00" Value="0"/>
                    <SplineDoubleKeyFrame KeyTime="00:00:01" Value="1"/>
                </DoubleAnimationUsingKeyFrames>
            </Storyboard>
        </VisualState>
        <VisualState x:Name="Unloaded" />
    </VisualStateManager>
</VisualStateManager>
<ContentPresenter
    x:Name="contentPresenter"
    ContentTemplate="{TemplateBinding ContentTemplate}"
    Content="{TemplateBinding Content}"
    HorizontalAlignment=
        "{TemplateBinding HorizontalContentAlignment}"
    Margin="{TemplateBinding Padding}"/>
</ContentPresenter>
</Setter.Value>
</Setter>
</Style>
</UserControl.Resources>
<StackPanel Margin="5">
    <ListBox Width="120"
        Margin="5"
        Height="200"
        Name="lstData"
        HorizontalAlignment="Left"
        ItemContainerStyle="{StaticResource ListBoxItemStyle1}"
        ItemTemplate="{StaticResource ListBoxDataTemplate}"
    >
</ListBox>
<Button Content="Add" Width="50"
    Margin="5"
    HorizontalAlignment="Left"
    Click="Button_Click" />
</StackPanel>
</UserControl>

```

هدف از این مثال، اعمال پویا نمایی FadeInOperate1 به هر یک از آیتم‌های اضافه شده در ListBox برنامه است. کدهای متناظر با افزودن آیتم‌های جدید به ListBox هم به شرح بعد می‌باشند:

MainPage.xaml.cs

```
using System;
using System.Windows;

namespace SilverlightApplication79
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();

            private void Button_Click(object sender, RoutedEventArgs e)
            {
                lstData.Items.Add(DateTime.Now.ToLongTimeString());
            }
        }
    }
}
```

توضیحات:

Fluid UI animation صرفاً قابل اعمال به اشیاء `ListBoxItem` یک کنترل `ListBox` هستند. برای این منظور باید خاصیت `ItemContainerStyle` یک `ListBox` را توسط `Style` جدیدی که تعریف خواهیم نمود مقدار دهی کرد. در مورد `VisualStateManager` و گروه‌بندی حالات مختلف در آن در طی فصول قبل بحث شد. در اینجا حالات `Loaded`، `BeforeLoaded` و `Unloaded` برای اعمال پویانمایی‌های لازم به هر یک از اشیاء اضافه یا حذف شده از `ListBox`، کاربرد دارند. برای نمونه در این مثال در حالت `BeforeLoaded` یک آیتم جدید، پویانمایی مطابق تعاریف ارائه شده، به شیء `ContentPresenter` اعمال می‌گردد. از این مثال به عنوان نمونه‌ای جهت تولید `ListBox` ایی که آیتم‌های آن قابل انتخاب نیستند نیز می‌توان استفاده کرد.

کنترل کلیک راست Mouse

در نگارش‌های قبلی Silverlight با کلیک راست بر روی هر قسمتی از برنامه تنها منوی نمایش تنظیمات Silverlight نمایش داده می‌شد (اصطلاحاً به آن `Context Menu` نیز گفته می‌شود). اکنون با تغییرات صورت گرفته در Silverlight 4، می‌توان عملیات متناسب به کلیک راست `Mouse` را نیز مدیریت نمود. برای این منظور باید دو رخداد `MouseRightButtonDown` و `MouseRightButtonUp` را کنترل کرد.

در ادامه در طی یک مثال، از این قابلیت به همراه کنترل Context Menu موجود در Silverlight 4 Toolkit استفاده خواهیم کرد. برای این منظور ابتدا کنترل Context Menu را از جعبه ابزار VS.NET کشیده و بر روی فرم قرار دهید. به این ترتیب ارجاعات لازم به این کنترل، به پروژهای جاری افزوده خواهند شد. سپس کدهای XAML برنامه به شرح بعد می‌باشند:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication80.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:toolkit="
        http://schemas.microsoft.com/winfx/2006/xaml/presentation/toolkit">
    <Grid x:Name="LayoutRoot" Background="White">
        <toolkit:ContextMenuService.ContextMenu>
            <toolkit:ContextMenu Name="contextMenu1" >
                <toolkit:MenuItem Header="Grow font"
                    x:Name="Grow"
                    Click="Grow_Click" />
                <toolkit:MenuItem Header="Shrink font"
                    x:Name="Shrink"
                    Click="Shrink_Click" />
            </toolkit:ContextMenu>
        </toolkit:ContextMenuService.ContextMenu>
        <TextBox Name="txtData"
            AcceptsReturn="True"
            MouseRightButtonDown="TextBox_MouseRightButtonDown"
            MouseRightButtonUp="TextBox_MouseRightButtonUp"/>
    </Grid>
</UserControl>
```

در اینجا رخدادهای MouseRightButtonDown و MouseRightButtonUp یک TextBox را می‌خواهیم کنترل نمائیم. این رخدادهای جهت تمامی UI Elements موجود نیز می‌توان تعریف کرد. اگر دقت کرده باشید کنترل Context menu را داخل ContextMenuService.ContextMenu قرار داده‌ایم. در WPF کلیه عناصر UI دارای خاصیت ContextMenu نیز می‌باشند. اما با توجه به عدم پشتیبانی این مورد در Silverlight، مجبور هستیم شیء دربرگیرنده‌ی آن‌را به این صورت تعریف نمائیم. در ادامه کدهای متناظر با این View را ملاحظه می‌نمائید:

MainPage.xaml.cs

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

namespace SilverlightApplication80
```

```

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

            private void Grow_Click(object sender, RoutedEventArgs e)
            {
                txtData.FontSize++;
            }

            private void Shrink_Click(object sender, RoutedEventArgs e)
            {
                txtData.FontSize--;
            }

            private void TextBox_MouseRightButtonDown(object sender,
                MouseButtonEventArgs e)
            {
                // handle the event so the default context menu is hidden
                e.Handled = true;
            }

            private void TextBox_MouseRightButtonUp(object sender,
                MouseButtonEventArgs e)
            {
                contextMenu1.IsOpen = true;
                contextMenu1.HorizontalOffset = e.GetPosition(LayoutRoot).X;
                contextMenu1.VerticalOffset = e.GetPosition(LayoutRoot).Y;
            }
        }
    }
}

```

نکته‌ی حائز اهمیت در این مثال آن است که اگر رویداد `MouseRightButtonDown` توسط برنامه مدیریت نشود، رویداد `MouseRightButtonUp` فراخوانی نخواهد شد.

استفاده از امکانات چاپگر

یکی از ضروریات برنامه‌های تجاری، امکان تهیه‌ی خروجی چاپی از گزارشات برنامه است. به همین منظور، امکانات استفاده از چاپگر نیز به Silverlight 4 اضافه شده است. کلاس‌های کار با چاپگر در فضای نام

PrintDocument ، مهم‌ترین کلاس این فضای نام ، System.Windows.Printing تعریف شده‌اند. می‌باشد و روشی غیرهمزمان را جهت ارسال فضایی مشخص، به چاپگر ارائه می‌دهد. لطفا جهت توضیحات بیشتر به مثال بعد دقت بفمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication81.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">
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="38" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <ScrollViewer Margin="4,4.135,4,4"
            HorizontalScrollBarVisibility="Auto"
            VerticalScrollBarVisibility="Auto">
            <Image x:Name="imgOne"
                Source="Images/Curls.png"
                Stretch="None" />
        </ScrollViewer>
        <Button x:Name="Print" Content="Print" Margin="4,5,4,0"
            Click="Print_Click" Grid.Row="1" />
    </Grid>
</UserControl>
```

در این View قصد داریم تصویر نمایش داده شده توسط کنترلی به نام imgOne را به چاپگر ارسال نمائیم. کدهای متناظر با این View در ادامه ذکر شده‌اند:

MainPage.xaml.cs

```
using System.Windows;
using System.Windows.Printing;
namespace SilverlightApplication81
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void Print_Click(object sender, RoutedEventArgs e)
        {
```

```

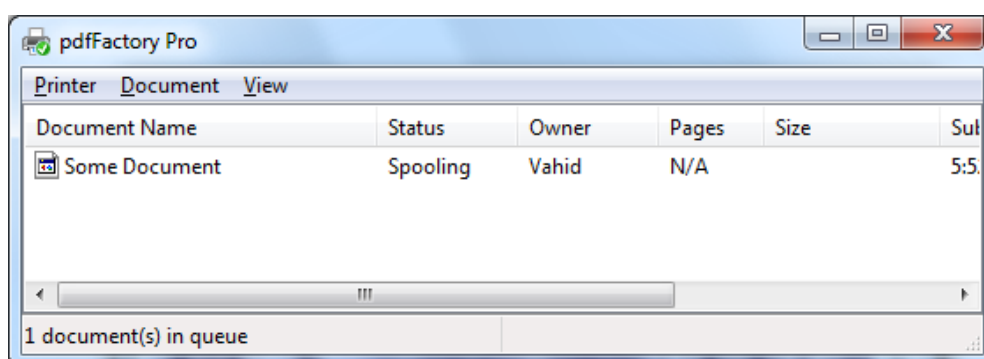
var pdoc = new PrintDocument();
pdoc.PrintPage += pdoc_PrintPage;
pdoc.EndPrint += pdoc_EndPrint;
pdoc.Print("Some Document");
}

void pdoc_EndPrint(object sender, EndPrintEventArgs e)
{
    MessageBox.Show("Printing operation completed");
}

void pdoc_PrintPage(object sender, PrintPageEventArgs args)
{
    args.PageVisual = imgOne;
    args.HasMorePages = false;
}
}
}

```

در این مثال ابتدا یک وهله از شیء `PrintDocument` ایجاد شده و سپس دو روال رخدادگردان `PrintPage` و `EndPrint` به آن جهت دریافت مشخصات ناحیه‌ی مورد نظر چاپی و تشخیص زمان پایان کار چاپ، تعریف شده‌اند. در پایان، توسط متد `Print`، عملیات چاپ با نمایش چاپگرهای موجود در سیستم آغاز خواهد شد (به دلایل امنیتی امکان ارسال محتوایی به چاپگر بدون تأیید کاربر وجود ندارد). آرگومان این متد در `printer spooler` ظاهر می‌گردد (شکل ۴).



شکل ۴- نمایشی از برنامه‌ی مدیریت صف موارد ارسالی به چاپگر در ویندوز.

در متد `pdoc_PrintPage`، خاصیت `PageVisual` بیانگر ناحیه‌ای است که باید به چاپگر ارسال شود (می‌تواند هر نوع `UI Elements` ایی باشد). توسط شیء `args` در اینجا می‌توان اندازه‌ی مورد نظر از ناحیه‌ی معرفی شده را نیز به کمک خاصیت `PrintableArea` آن مشخص نمود. این روال رخدادگردان به ازای چاپ هر صفحه یکبار فراخوانی خواهد شد. به همین جهت توسط خاصیت `HasMorePages` می‌توان مشخص ساخت که آیا تعداد

صفحات بیشتری قرار است به چاپگر ارسال شوند یا خیر. برای مثال فرض کنید قصد دارید دو صفحه را به چاپگر ارسال کنید؛ اکنون روال رخداد گردان `pdoc_PrintPage` برنامه به شکل زیر درخواهد آمد:

C#

```
if (_pageNumber == 0)
{
    args.HasMorePages = true;
    //...
    _pageNumber++;
}
else
{
    args.HasMorePages = false;
    //...
}
```

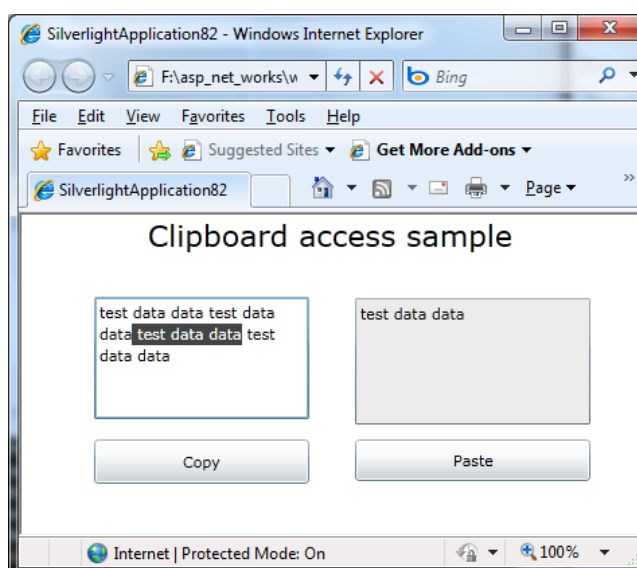
یک متغیر عددی به نام `_pageNumber` در سطح کلاس تعریف شده است و در ابتدای کار مقدار آن صفر خواهد بود. در اینجا نیاز است تا مقدار خاصیت `HasMorePages` به `true` تنظیم شود تا پس از پایان کار صفحه‌ی اول، این روال مجدداً فراخوانی گردد. اکنون برای بار دوم فراخوانی این متد، مقدار متغیر `_pageNumber` دیگر صفر نبوده و بیانگر صفحه‌ی دوم یا صفحه‌ی آخر مورد نظر ما است. بنابراین خاصیت `HasMorePages` را به `false` تنظیم کرده‌ایم.

در متد `pdoc_EndPrint` با بررسی خاصیت `e.ErrorMessage` می‌توان دریافت که آیا در حین چاپ خطایی رخ داده است یا خیر. این روال رخدادگردان در پایان کار چاپ و یا در زمان انصراف از عملیات فراخوانی می‌گردد.

توسط شیء `pdoc` فوق امکان تعریف روال رخداد گردان `StartPrint` نیز وجود دارد. این مورد می‌تواند جهت نمایش کنترل `BusyIndicator` و سپس محو آن در روال رخدادگردان `EndPrint`، مفید باشد.

دسترسی به Clipboard

یکی دیگر از امکاناتی که برنامه‌های کاربردی به آن نیاز دارند کار با `Clipboard` سیستم است که در Silverlight 4 به شکلی قابل اجرا در سکوهاى مختلف کارى مهیا شده است. دسترسی به `Clipboard` نیز همانند بسیاری از ویژگی‌های دیگر Silverlight نیاز به تأیید کاربر داشته و بدون آن به دلایل امنیتی میسر نخواهد بود. کلاس `Clipboard` در Silverlight 4 تنها محدود است به کار با اطلاعات متنی و شامل سه متد `GetText`، `SetText` و `ContainsText` می‌باشد. لطفاً به مثال بعد در این زمینه دقت بفرمائید (شکل ۵).



شکل ۵- نمایی از برنامه‌ی دسترسی به Clipboard در Silverlight 4.

کدهای XAML پروژه‌ی دسترسی به Clipboard را در ادامه ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication82.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"
        HorizontalAlignment="Center">
        <TextBlock Height="43" Margin="0" Name="textBlock1"
            Text="Clipboard access sample"
            VerticalAlignment="Top"
            Width="317" TextAlignment="Center" FontSize="22" />
        <Button Content="Copy" Height="33" HorizontalAlignment="Left"
            Margin="19,167,0,0" Name="btnCopy"
            VerticalAlignment="Top" Width="160"
            Click="btnCopy_Click" />
        <Button Content="Paste" Height="31" HorizontalAlignment="Left"
            Margin="213,167,0,0" Name="btnPaste"
            VerticalAlignment="Top" Width="175"
            Click="btnPaste_Click" />
        <TextBox Height="94" HorizontalAlignment="Left"
            Margin="213,62,0,0" Name="txtPaste" Text=""
            VerticalAlignment="Top" Width="175"
            IsReadOnly="True" TextWrapping="Wrap" />
    </Grid>
</UserControl>
```

```

        <TextBox Height="91" HorizontalAlignment="Left"
            Margin="19,61,0,0" Name="txtCopy"
            VerticalAlignment="Top" Width="160"
            TextWrapping="Wrap" />
        <TextBlock Height="24" HorizontalAlignment="Left"
            Margin="22,213,0,0" Name="tbMode" Text=""
            VerticalAlignment="Top" Width="366"
            TextAlignment="Center" FontWeight="Bold"
            FontSize="18" />
    </Grid>
</UserControl>

```

قصد داریم متن انتخابی در TextBox سمت چپ را در Clipboard سیستم کپی کرده و سپس آن را در TextBox سمت راست Paste نمائیم. کدهای متناظر با این View در ادامه ذکر شده‌اند:

MainPage.xaml.cs

```

using System.Windows;

namespace SilverlightApplication82
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void btnCopy_Click(object sender, RoutedEventArgs e)
        {
            if (txtCopy.SelectedText.Length > 0)
                Clipboard.SetText(txtCopy.SelectedText);
            else
                MessageBox.Show("Nothing to copy!");
        }

        private void btnPaste_Click(object sender, RoutedEventArgs e)
        {
            if (Clipboard.ContainsText())
                txtPaste.Text = Clipboard.GetText();
        }
    }
}

```

با استفاده از متد `Clipboard.SetText`، متن انتخابی در Clipboard کپی خواهد شد (البته پس از اخذ مجوز از کاربر). توسط متد `Clipboard.ContainsText` بررسی خواهیم کرد که آیا متنی در Clipboard موجود است؟ اگر پاسخ مثبت بود، این متن توسط متد `Clipboard.GetText` دریافت می‌گردد.

کنترل ViewBox

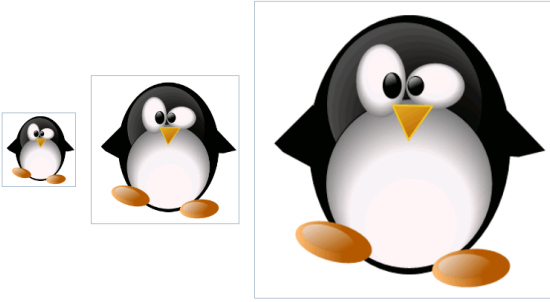
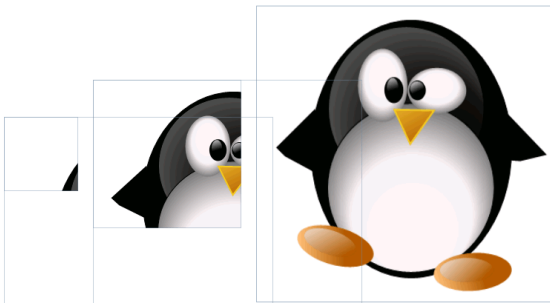
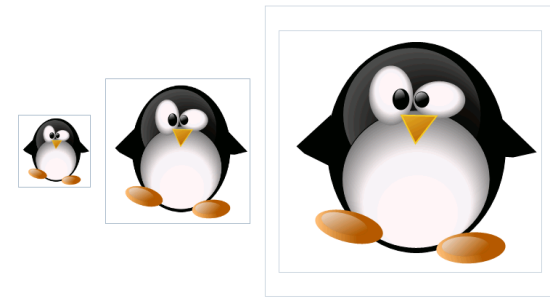
کنترل ViewBox یکی از کنترل‌هایی است که پس از تکمیل نهایی آن در Silverlight toolkit، با کتابخانه‌ی اصلی Silverlight 4 یکی شده است؛ همچنین برای طراحان WPF نیز از نگارش یک آن در دسترس بوده است. عناصر فرزند قرار گرفته در این کنترل جهت پر کردن فضای آن به صورت خودکار تغییر ابعاد خواهند داد. این کنترل نیز همانند کنترل Image دارای خاصیت Stretch بوده (با همان مقادیر، اما قابل اعمال به کلیه عناصر UI) و امکان تنظیم خاصیت StretchDirection در آن نیز میسر است. خاصیت StretchDirection سه مقدار Both (پیش فرض)، UpOnly و DownOnly را می‌پذیرد. در حالت UpOnly، کنترل‌های قرار گرفته داخل ViewBox تنها امکان بزرگ شدن را خواهند داشت و با تنظیم مقدار آن به DownOnly، این کنترل‌ها فقط می‌توانند کوچک شوند. در حالت Both، هر دو حالت بزرگ شدن یا کوچک شدن پشتیبانی می‌گردد. لطفا جهت توضیحات بیشتر به مثال بعد دقت بفرمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication83.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="450" d:DesignWidth="800">
    <StackPanel Orientation="Horizontal">
        <Viewbox Height="100" Width="100" Margin="10"
            Stretch="Fill" StretchDirection="DownOnly" >
            <Image Source="Images/Penguin.png" Stretch="Fill" />
        </Viewbox>
        <Viewbox Height="200" Width="200" Margin="10"
            Stretch="Fill" StretchDirection="DownOnly">
            <Image Source="Images/Penguin.png" />
        </Viewbox>
        <Viewbox Height="400" Width="400" Margin="10"
            Stretch="Fill" StretchDirection="DownOnly">
            <Image Source="Images/Penguin.png" />
        </Viewbox>
    </StackPanel>
</UserControl>
```

در این مثال سه ViewBox با اندازه‌های متفاوت جهت نمایش تصاویری بر روی صفحه قرار گرفته‌اند. تصاویری از این مثال را در حالت طراحی با مقادیر مختلف خواص ViewBox در جدول بعد ملاحظه خواهید نمود.

جدول ۱ - تاثیر مقادیر مختلف خواص ViewBox بر نحوه‌ی نمایش نهایی اشیاء فرزند آن

مقادیر متفاوت خواص ViewBox	نتیجه نهایی
Stretch="Fill" StretchDirection="Both"	
Stretch="Fill" StretchDirection="UpOnly"	
Stretch="Fill" StretchDirection="DownOnly"	

یکی از کاربردهای ViewBox، ایجاد یک صفحه‌ی مجازی از طریق کد نویسی برای قرار دادن محتوای درنظر گرفته شده بر روی آن جهت ارسال به چاپگر می‌باشد.