Silverlight 4

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

فهرست مطالب

فصل ۱۷ – استفاده از تصاویر و فایل‌های چند رسانه‌ای در Silverlight

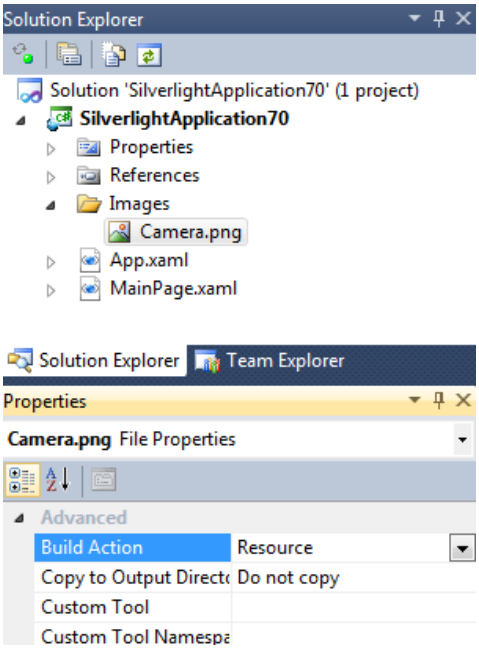
در Silverlight دو نوع منبع (resource) را می‌توان تعریف کرد:

- Binary Resources : همانند اضافه کردن تصاویر، قلم‌ها، فایل‌های ویدیویی و صوتی به برنامه. به این نوع منابع، منابع مدفون شده (Embedded Resources) نیز گفته می‌شود؛ زیرا در حین Compile در قسمت منابع اسمبلی برنامه قرار خواهند گرفت.
- Logical Resources : تعریف اشیاء .NET. به صورت resource dictionary در سطح برنامه یا User control و امثال آن؛ جهت تعریف Styles ، control templates و غیره.

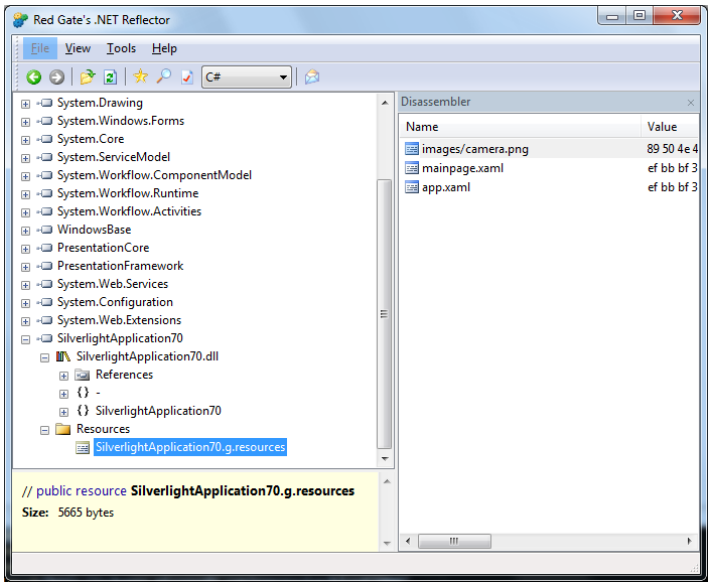
استفاده از کنترل Image به همراه تصاویر منبع

بسیاری از برنامه‌های Silverlight نیاز دارند تا تصاویر مدفون شده‌ی مختلفی را در حین اجرای برنامه نمایش دهند. نحوه‌ی استفاده از VS.NET جهت افزودن اینگونه منابع بسیار ساده است. با استفاده از منوی پروژه، گزینه‌ی Add Existing item ، می‌توان یک تصویر یا سایر منابع Binary را به پروژه اضافه کرد. خوشبختانه در این حالت Build action مناسبی (شکل ۱) جهت این فایل تصویری اضافه شده توسط VS.NET در نظر گرفته می‌شود. علاوه بر تنظیم Build action به حالت Resource ، باید مقدار خاصیت Copy to Output Directory نیز به Do not Copy تنظیم شده باشد. در این حالت اگر برنامه را Compile کرده و فایل اسمبلی آن‌را توسط برنامه‌ی Reflector مطالعه نمائید، می‌توان به سادگی منابع مدفون شده در آن را تشخیص داد (شکل ۲). برنامه‌ی Reflector از آدرس بعد قابل دریافت است:

http://www.red-gate.com/products/reflector/



شکل ۱– تنظیم Build action مناسب جهت منابع مدفون شده



شکل ۲– استفاده از برنامه‌ی Reflector جهت مشاهده‌ی منابع مدفون شده‌ی یک اسمبلی

در ادامه قصد داریم این تصویر اضافه شده به پروژه را توسط کنترل تصویر، نمایش دهیم. لازم به ذکر است که این کنترل تنها قادر به نمایش فایل‌هایی از نوع PNG و JPG می‌باشد. پیشتر نیز با نحوه‌ی استفاده از این نوع منابع آشنا شده بودیم:

XAML

```
<Image Source="images/Camera.png" Width="48" Height="48" />
```

مقدار Source در اینجا یک مسیر نسبی است. مثال ذیل یک مسیر مطلق را نمایش می‌دهد:

XAML

```
<Image Source="http://mysite.com/images/Camera.png"
                                    Width="48" Height="48" />
```

اکنون اگر تعریف کنترل Image به نحو زیر باشد :

XAML

```
<Image Name="img1" />
```

و قصد داشته باشیم تا تصویر مدفون شده‌ی images/Camera.png متعلق به پروژه‌ی SilverlightApplication70 را از طریق برنامه نویسی نمایش دهیم، روش انجام کار به صورت زیر خواهد بود:

C#

```
using System;
using System.Windows;
using System.Windows.Media.Imaging;
...
void showImg()
{
   var sr =
           Application.GetResourceStream(
           new Uri("SilverlightApplication70;component/Images/Camera.png",
                       UriKind.Relative));
  var bmp = new BitmapImage();
  bmp.SetSource(sr.Stream);
  img1.Source = bmp;
  img1.Width = 48;  img1.Height = 48;
}
```

همانطور که ملاحظه می‌نمائید برای انجام اینکار از متد Application.GetResourceStream کمک گرفته می‌شود. نحوه‌ی تشکیل پارامتر اصلی آن نیز به صورت زیر است:

```
new Uri("SilverlightApplication70;component/Images/Camera.png",
              UriKind.Relative)
```

در اینجا SilverlightApplication70 نام اسمبلی برنامه‌ی جاری بوده و سپس به کمک component/; تصویر مورد نظر قید می‌گردد. همچنین مسیردهی نسبی آن نیز حائز اهمیت است.

استفاده از کنترل Image به همراه تصاویر قرار گرفته در فایل XAP

فایل‌های تصویری برنامه را بجای مدفون سازی در DLL برنامه می‌توان در خود فایل XAP نهایی تولید شده نیز قرار داد. همانطور که پیشتر نیز ذکر شد این نوع فایل‌ها اساسا فایل ZIP می‌باشند و به سادگی می‌توان فایل‌های مورد نظر را به آن‌ها افزود. در این حالت نیز روش بارگذاری فایل‌های تصویری با استفاده از برنامه نویسی همانند مثال قبل خواهند بود؛ تنها با این تفاوت که آرگومان متد Application.GetResourceStream به صورت زیر تغییر خواهد کرد:

C#

```
new Uri("MyImage.png", UriKind.Relative)
```

در اینجا دیگری نیازی به ذکر نام اسمبلی و قسمت component/; نمی‌باشد.

بارگذاری تصاویر از خارج از Site

روش بارگذاری تصاویر از خارج از Site جاری به کمک برنامه نویسی به شرح بعد است:

C#

```
using System;
using System.Windows;
using System.Windows.Media.Imaging;
...
private void loadExternalImg()
{
  var bi = new BitmapImage
           {
             UriSource = new Uri("http://www.google.com/images/firefox/personas.png")
           };
  img1.Source = bi;
  img1.ImageOpened += img1_ImageOpened;
  img1.ImageFailed += img1_ImageFailed;
}

void img1_ImageFailed(object sender, ExceptionRoutedEventArgs e)
{
   MessageBox.Show(e.ErrorException.Message);
}

void img1_ImageOpened(object sender, RoutedEventArgs e)
{
}
```

دو نکته‌ی مهم به همراه این مثال ارائه شده است. رخدادهای به پایان رسیدن بارگذاری تصویر مورد نظر و همچنین رخداد شکست بارگذاری یک تصویر قرار گرفته در خارج از سایت جاری.

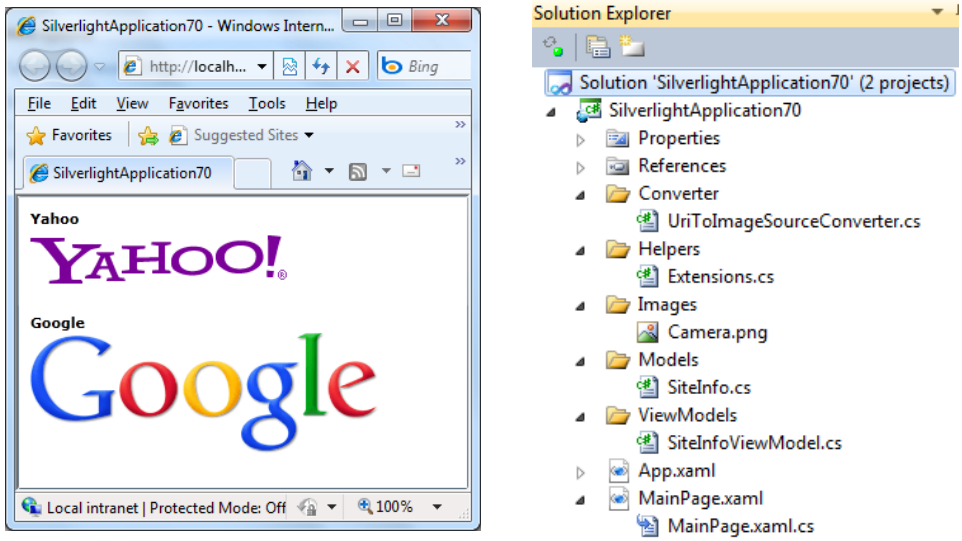
اگر توسط روال رویداد گردان img1_ImageFailed پیغام AG_E_NETWORK_ERROR را دریافت کردید، این خطا به معنای عدم دسترسی، به علت نبود فایل‌های clientaccesspolicy.xml و امثال آن نیست؛ زیرا در Silverlight مجاز به بارگذاری تصاویر خارجی می‌باشیم. این خطا بدین علت صادر می‌شود که در Silverlight درخواست بارگذاری فایل یا تصویر از یک پروتکل مانند file:// (زمانیکه ASP.NET Web site همراه را ایجاد نکرده باشید) به پروتکل دیگری مانند http:// مجاز نیست و به صورت خودکار سد خواهد شد. بنابراین در این حالات بهتر است حتما ASP.NET Web site همراه را نیز ایجاد نمائید تا برنامه تحت پروتکل http اجرا گردد.

استفاده از کنترل Image به همراه تصاویر حاصل از عملیات Binding

اگر در حین عملیات Binding ، Uri یک فایل تصویری را به خاصیت Source یک کنترل Image نسبت دهیم، تصویری نمایش داده نخواهد شد؛ زیرا این خاصیت از نوع ImageSource است و نه Uri . برای رفع این مشکل باید یک کلاس تبدیل کننده، تهیه شود که نحوه‌ی انجام آن‌را در مثال بعد ملاحظه خواهید نمود (شکل ۳).

ابتدا یک برنامه‌ی Silverlight جدید را به همراه پروژه‌ی ASP.NET Web Site آن آغاز نمائید (همانطور که پیشتر نیز ذکر گردید، وجود این Web Site برای دریافت تصاویر از یک سایت خارجی الزامی است). در این پروژه از کلاس Extensions توسعه یافته در فصول قبل استفاده می‌گردد و از ذکر مجدد کدهای آن صرفنظر خواهد شد.

کدهای کلاس تبدیل کننده‌ی Url به نوع قابل استفاده در ImageSource را در ادامه ملاحظه می‌نمائید. در این کلاس نیز رخداد ImageFailed جهت بررسی خطاهای حاصل از بارگذاری تصویر، اضافه و مدیریت گردیده است. از آنجائیکه در این مثال قصد داریم آدرس‌های اینترنتی را نمایش دهیم، نوع Uri تعریف شده به صورت UriKind.Absolute مشخص گردیده است.



شکل ۳– نمایی از پروژه و برنامه‌ی بارگذاری تصاویر به کمک Binding

UriToImageSourceConverter.cs

```
using System;
using System.Diagnostics;
using System.Globalization;
using System.Windows.Data;
using System.Windows.Media.Imaging;
namespace SilverlightApplication70.Converter
{
    public class UriToImageSourceConverter : IValueConverter
    {
        public object Convert(object value,
            Type targetType, object parameter,
            CultureInfo culture)
        {
          if (value == null) return null;
          string imageUrl = value.ToString();
          var uriSource = new Uri(imageUrl, UriKind.Absolute);
          var result = new BitmapImage(uriSource);
          result.ImageFailed +=
          (sender, e) => Debug.WriteLine(e.ErrorException.ToString());
          return result;
        }

        public object ConvertBack(object value, Type targetType,
            object parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

در ادامه کدهای Model برنامه را ملاحظه خواهید نمود. این مدل از خواص عنوان و مسیر فایل Logo یک سایت تشکیل شده است:

SiteInfo.cs

```
using System.ComponentModel;
using SilverlightApplication70.Helpers;

namespace SilverlightApplication70.Models
{
    public class SiteInfo : INotifyPropertyChanged
    {
        private string _title = string.Empty;
        public string Title
        {
            get { return _title; }
            set
            {
                if (_title != value)
                {
                    _title = value;
                    PropertyChanged.Raise(() => Title);
                }
            }
        }

        private string _logoUrl = string.Empty;
        public string LogoUrl
        {
            get { return _logoUrl; }
            set
            {
                if (_logoUrl != value)
                {
                    _logoUrl = value;
                    PropertyChanged.Raise(() => LogoUrl);
                }
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
    }
}
```

کدهای ViewModel برنامه به شرح بعد هستند. در این ViewModel ، لیستی از اشیاء Model ساخته شده و در اختیار View برنامه یا همان صفحه‌ی اصلی برنامه قرار خواهند گرفت.

SiteInfoViewModel.cs

```
using System.Collections.ObjectModel;
using SilverlightApplication70.Models;

namespace SilverlightApplication70.ViewModels
{
    public class SiteInfoViewModel
    {
        public ObservableCollection<SiteInfo> SitesInfo { set; get; }
        public SiteInfoViewModel()
        {
            SitesInfo = new ObservableCollection<SiteInfo>
                 {
                    new SiteInfo
                    {
                       Title = "Yahoo",
                       LogoUrl =
                            "http://l.yimg.com/a/i/ww/met/yahoo_logo_us_061509.png"
                    },
                    new SiteInfo
                    {
                       Title = "Google",
                       LogoUrl =
                            "http://www.google.com/intl/en_ALL/images/srpr/logo1w.png"
                    }
                };
        }
    }
}
```

و در پایان نحوه‌ی استفاده از این اطلاعات را در View برنامه مشاهده می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication70.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"
    xmlns:vm="clr-namespace:SilverlightApplication70.ViewModels"
    xmlns:cv="clr-namespace:SilverlightApplication70.Converter"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
        <vm:SiteInfoViewModel x:Key="vmSiteInfoViewModel" />
        <cv:UriToImageSourceConverter
              x:Key="cvUriToImageSourceConverter" />
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White"
```

```
            DataContext=
             "{Binding Source={StaticResource vmSiteInfoViewModel}}"
            >
        <ListBox ItemsSource="{Binding SitesInfo}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Margin="5">
                       <TextBlock Text="{Binding Title}"
                             FontWeight="Bold"/>
                        <Image
                            Source="{Binding LogoUrl,
                            Converter={StaticResource
                                  cvUriToImageSourceConverter}}" />
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </Grid>
</UserControl>
```

در کدهای XAML این View، ابتدا منابع ثابت صفحه معرفی شده و سپس DataContext گرید، مقدار دهی گردیده است. برای نمایش قالب ویژه‌ی مورد نظر، یک DataTemplate جدید را برای کنترل استاندارد ListBox تعریف کرده‌ایم. خاصیت Source تصویر، در ابتدا اطلاعات خود را از طریق خاصیت LogoUrl دریافت کرده و سپس آنها را به Converter تعریف شده در قسمت تعاریف Binding ارسال خواهد کرد. حاصل نهایی دریافتی، اطلاعاتی از نوع ImageSource می‌باشد.

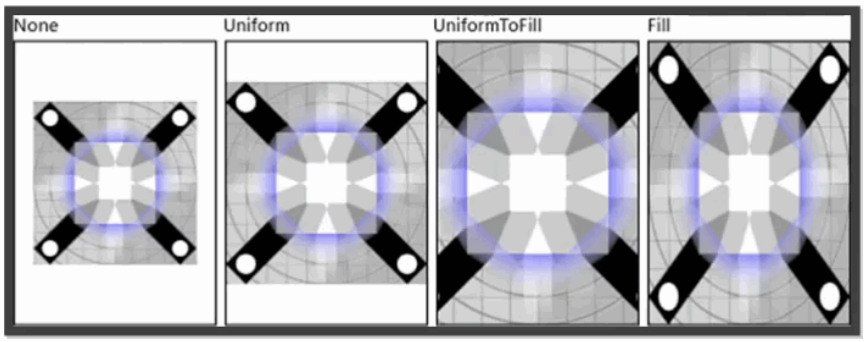
آشنایی با خاصیت‌های Stretch و Clip یک کنترل نمایش تصویر

نحوه‌ی نمایش و میزان کشیدگی یک تصویر جهت پر کردن فضای یک کنترل تصویری با استفاده از خاصیت Stretch آن مشخص می‌شود و مقدار پیش فرض آن مساوی Uniform است.

در شکل بعد نمایی از اثرات مقادیر مختلف این خاصیت را بر نحوه‌ی نمایش نهایی یک تصویر ملاحظه می‌نمائید.

XAML

```
<Image x:Name="MyImage" Source="Images/GiantSeaTurtle.jpg"
    Stretch="Uniform" />
```



شکل ۴– نمایی از تاثیرات مقادیر مختلف خاصیت Stretch

معانی این مقادیر مختلف به شرح زیر است:

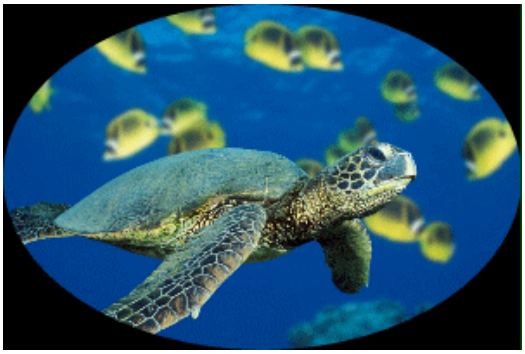
- None : تصویر در همان اندازه‌ی اصلی خودش نمایش داده خواهد شد. در این حالت با استفاده از خواص VerticalAlignment و HorizontalAlignment می‌توان محل قرارگیری تصویر را تعیین نمود.
- Uniform : باحفظ نسبت طول و عرض تصویر، سعی خواهد نمود تا سطح کنترل تصویر را پوشش دهد.
- UniformToFill : باحفظ نسبت طول و عرض تصویر، سعی خواهد نمود تا سطح کنترل تصویر را پوشش دهد. در این حالت ممکن است قسمتی از تصویر را از دست دهیم اما بهترین کیفیت نمایشی حاصل خواهد شد.
- Fill : بدون حفظ طول و عرض تصویر، تمام ناحیه کنترل تصویر را پوشش خواهد داد.

یکی دیگر از ویژگی‌های جالب کنترل Image در Silverlight امکان تنظیم ناحیه‌ی قابل رویت یک تصویر می‌باشد. لطفا به مثال بعد در این زمینه دقت بفرمائید.

XAML

```
<Image x:Name="MyImage" Source="Images/GiantSeaTurtle.jpg"
    Stretch="UniformToFill">
    <Image.Clip>
       <EllipseGeometry x:Name="Ellipse" RadiusX="150" RadiusY="100"
            Center="150,100"/>
    </Image.Clip>
</Image>
```

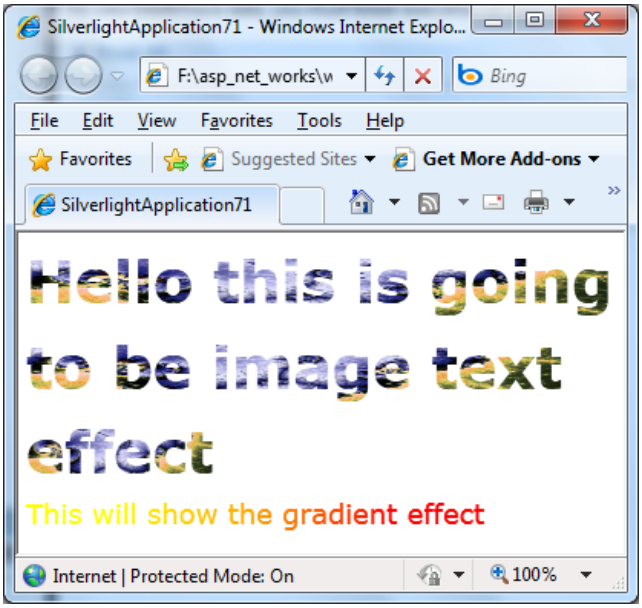
در اینجا با استفاده از ویژگی Clip ، ناحیه‌ی قابل رویت تصویر را توسط شیء EllipseGeometry مشخص نموده‌ایم؛ که نتیجه‌ی حاصل را در شکل بعد ملاحظه می‌نمائید:



شکل ۵– استفاده از ویژگی Clip یک کنترل Image جهت مشخص ساختن ناحیه‌ی قابل رویت

نقاشی بر روی رابط گرافیکی کاربر به کمک ImageBrush

تاکنون با SolidColorBrush و LinearGradientBrush در طول کتاب جاری آشنا شده‌ایم. نوع دیگری از قلم‌ها در Silverlight موجود است به نام ImageBrush که توسط آن می‌توان تصاویر را بر روی هر کدام از UI Element موجود، نقاشی کرد و نمایش داد. در ادامه در طی یک مثال قصد داریم تصویری را به حروف نمایش داده شده توسط یک TextBlock اعمال نمائیم.



شکل ۶– نمایی از اعمال دو نوع قلم مختلف بر متن نمایش داده شده

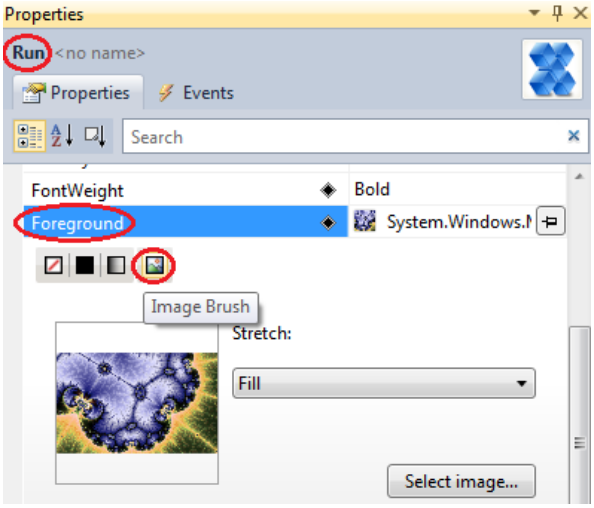
در کدهای XAML بعد نحوه‌ی تعریف یک ImageBrush و اعمال آن‌را به متن نمایش داده شده توسط یک TextBlock ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication71.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" Margin="5">
        <TextBlock Foreground="Red" TextWrapping="Wrap" >
            <Run FontWeight="Bold"
                  Text="Hello this is going to be image text effect"
                 FontSize="38">
                <Run.Foreground>
                    <ImageBrush ImageSource="images/Curls.png" />
                </Run.Foreground>
            </Run>
            <LineBreak />
            <Run
               Text="This will show the gradient effect" FontSize="18">
                <Run.Foreground>
                  <LinearGradientBrush StartPoint="0,0" EndPoint="1,1" >
                      <GradientStop Color="Yellow"
                          Offset="0.2"></GradientStop>
                        <GradientStop Color="Orange"
                          Offset="0.5"></GradientStop>
                        <GradientStop Color="Red"
                          Offset="0.8"></GradientStop>
                    </LinearGradientBrush>
                </Run.Foreground>
            </Run>
        </TextBlock>
    </Grid>
</UserControl>
```

جهت تکمیل بحث ، از یک LinearGradientBrush نیز جهت اعمال گرادیان به قسمتی از متن نمایش داده شده، کمک گرفته شد.

همچنین با بهبودهای حاصل شده در ویرایشگر XAML مربوط به VS.NET 2010 ، به سادگی می‌توان به خاصیت Foreground یک شیء Run قرار گرفته در TextBlock ، یک تصویر دلخواه را انتساب داد (شکل ۷).



شکل ۷– امکان اعمال ساده‌ی یک ImageBrush به کمک VS.NET 2010 .

نمایش فایل‌های چند رسانه‌ای به کمک Silverlight

بسیاری از ابتدایی‌ترین برنامه‌های تهیه شده با Silverlight صرفا Video player بوده‌اند؛ زیرا مهم‌ترین هدف نگارش‌های اولیه Silverlight نیز تسهیل در ارائه‌ی فایل‌های چندرسانه‌ای بوده است. همانطور که در قسمت تاریخچه‌ی نگارش‌های مختلف Silverlight در ابتدای کتاب جاری نیز عنوان گردید، Silverlight از قالب‌های چند رسانه‌ای بسیار متنوعی پشتیبانی کرده و همچنین بهره‌گیری از GPU Acceleration جهت نمایش آن‌ها را نیز باید مد نظر داشت که در نمونه‌های مشابه مانند Flash، تا سال ۲۰۱۰ خبری از آن نبوده است. لیست کامل قالب‌های صوتی و تصویری پشتیبانی شده توسط Silverlight را در ادامه ملاحظه خواهید نمود:

Video

- Raw Video
- YV12 - YCrCb(4:2:0)
- RGBA - 32 bit Alpha Red, Green, Blue
- WMV1: Windows Media Video 7
- WMV2: Windows Media Video 8
- WMV3: Windows Media Video 9
  - Supports Simple and Main Profiles.
  - Supports only progressive (non-interlaced) content.
- WMVA: Windows Media Video Advanced Profile, non-VC-1
- WVC1: Windows Media Video Advanced Profile, VC-1
  - Supports Advanced Profile.
  - Supports only progressive (non-interlaced) content.
- H264 (ITU-T H.264 / ISO MPEG-4 AVC)
  - Supports H.264 and MP43 codecs.
  - Supports Base, Main, and High Profiles
  - Supports only progressive (non-interlaced) content.
  - Supports only 4:2:0 chroma subsampling profiles.

- Supports PlayReady DRM with Mp4 (H264 and AAC-LC)

Audio

- “1”. This is Linear 8 or 16 bit Pulse Code Modulation. Roughly speaking, this is WAV format.
- "353" - Microsoft Windows Media Audio v7, v8 and v9.x Standard (WMA Standard)
- "354" - Microsoft Windows Media Audio v9.x and v10 Professional (WMA Professional)
  - Supports full fidelity decoding of WMA 10 Professional Low Bit Rate (LBR) modes in the 32-96 kbps range.
  - Multichannel (5.1 and 7.1 surround) audio content is automatically mixed down to stereo.
  - 24 bit audio will return silence.
  - Sampling Rates beyond 48000 return an invalid format error code in same-domain and a 4001 in cross-domain scenarios.
- "85" - ISO MPEG-1 Layer III (MP3)
- "255" - ISO Advanced Audio Coding (AAC)
  - Supports Low Complexity (AAC-LC) decoding at full fidelity (up to 48 kHz).
  - High Efficiency (HE-AAC) encoded content will decode only at half fidelity (up to 24 kHz).
  - Multichannel (5.1 surround) audio content is not supported.

بررسی امکانات کنترل MediaElement

در ادامه مروری خواهیم داشت بر توانایی‌های کنترل MediaElement جهت نمایش فایل‌های چندرسانه‌ای. کاربر برنامه برای استفاده از کنترل MediaElement ، نیازی به نصب هیچگونه media player خاصی نداشته و به محض نصب افزونه‌ی Silverlight ، امکانات آن قابل استفاده خواهند بود. ساده‌ترین کاربرد این کنترل در مثال XAML بعد ذکر شده است:

XAML

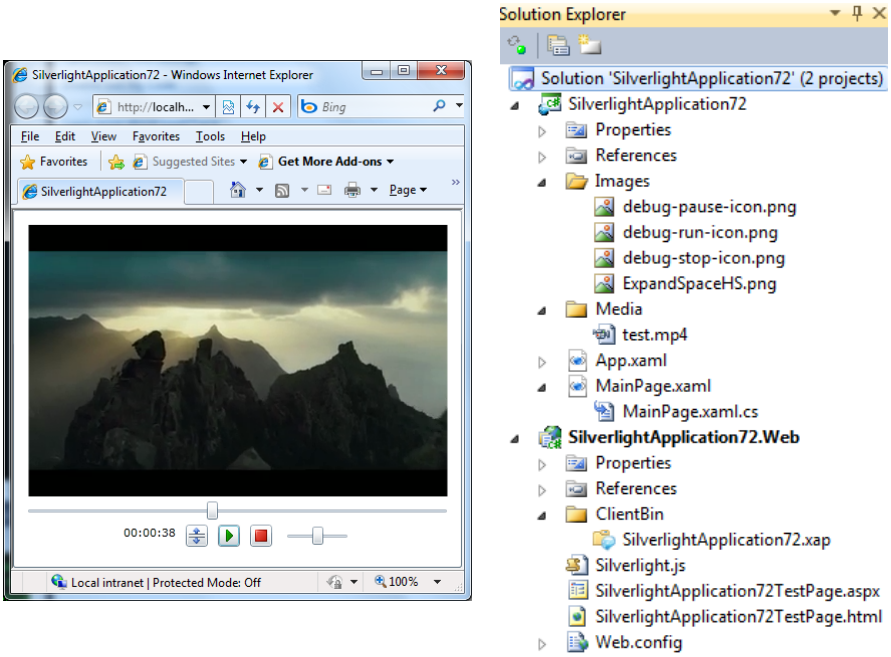
```
<MediaElement
          Width="472" Height="328" Source="video.wmv"
          Stretch="Fill" x:Name="Video1" />
```

به این ترتیب یک کنترل MediaElement با اندازه‌ی مشخص ، فایل ویدیویی video.wmv را نمایش خواهد داد. در اینجا نیز هنگام افزودن فایل video.wmv به پروژه، به خواص آن در VS.NET مراجعه نموده و Build Action آن را بر روی Resource قرار دهید؛ تا امکان پخش و نمایش آن فراهم گردد.

قسمت بصری این کنترل صرفا کار نمایش را عهده دار است و از کنترل‌های توقف نمایش، از سرگیری و غیره در آن خبری نیست که باید به کمک برنامه نویسی فراهم شوند. برای این منظور یک پروژه‌ی جدید Silverlight را به همراه ASP.NET Web site آن آغاز نمائید. کدهای XAML و کدهای C# متناظر آن در ادامه ذکر شده‌اند. در لابلای کدها نیز جهت معرفی عملکرد هر قسمت از Comments مناسب استفاده گردیده است.

در اینجا علت استفاده از ScrollViewer ، بکارگیری دکمه‌ی افزایش طول و عرض صفحه‌ی نمایش فایل چند رسانه‌ای است. در این حالت در صورت نیاز Scrollbar ایی به صورت خودکار نمایش داده خواهد شد. همچنین

در کدهای متناظر این View ، از مقادیر متفاوت Stretch که پیشتر در مورد آن‌ها توضیح داده شد جهت تغییر اندازه‌ی قسمت نمایشی استفاده گردیده است.



شکل ۸– نمایی از ساختار فایل‌ها و پروژه‌ی نمایش فایل‌های چند رسانه‌ای

کدهای XAML صفحه‌ای اصلی برنامه:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication72.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"
    Name="ThisPrv"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <ScrollViewer
            HorizontalAlignment="Stretch"
            VerticalScrollBarVisibility="Auto"
            VerticalAlignment="Stretch"
            VerticalContentAlignment="Stretch"
            HorizontalContentAlignment="Stretch">
            <StackPanel Margin="5"  >
                <MediaElement Margin="5"
                      HorizontalAlignment="Center"
                      VerticalAlignment="Center"
```

```
                        Height="auto" Width="auto"
                    MediaOpened="mediaElement1_MediaOpened"
                    MediaEnded="mediaElement1_MediaEnded"
                    Name="mediaElement1"
                    Source="media/test.mp4"
                    Stretch="None"
                    Loaded="mediaElement1_Loaded"
                    AutoPlay="True"
                    MediaFailed="mediaElement1_MediaFailed" />
            <Slider Name="seekBar"
                ValueChanged="seekBar_ValueChanged"
                 />
            <StackPanel Orientation="Horizontal"
                    HorizontalAlignment="Center">
                <TextBlock Margin="5"
                       Name="durationLbl"
                       Text="{Binding ElementName=ThisPrv,
                              Path=MediaLen}"  />

                <Button ToolTipService.ToolTip="Expand"
                    Name="btnExpand" Margin="5"
                    Click="btnExpand_Click" >
                    <Image Width="16" Height="16"
                       Source="images/ExpandSpaceHS.png"  />
                </Button>

                <Button ToolTipService.ToolTip="Play/Pause"
                    Name="btnPlay" Margin="5"
                    Click="btnPlay_Click" >
                    <Image Name="imgPaly"
                         Source="images/debug-run-icon.png"  />
                </Button>

                <Button Name="btnStop" ToolTipService.ToolTip="Stop"
                    Margin="5" Click="btnStop_Click">
                    <Image Source="images/debug-stop-icon.png"  />
                </Button>

                <Slider Name="volumeSlider"
                    ToolTipService.ToolTip="Volume"
                    Margin="5"  VerticalAlignment="Center"
                    ValueChanged="volumeSlider_ValueChanged"
                    Minimum="0" Maximum="1" Value="0.5"
                    Width="70"/>
            </StackPanel>
          </StackPanel>
       </ScrollViewer>
   </Grid>
</UserControl>
```

کدهای #C متناظر با اعمال مختلف منتسب به اشیاء صفحه و MediaElement :

MainPage.xaml.cs

```
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Threading;

namespace SilverlightApplication72
{
    public partial class MainPage : INotifyPropertyChanged
    {
        #region Fields (3)

        private bool _isPlaying;
        string _mediaLen;
        private DispatcherTimer _timer;

        #endregion Fields

        #region Constructors (1)

        public MainPage()
        {
            InitializeComponent();
            initTimer();
        }

        #endregion Constructors

        #region Properties (3)

        //نمایش خودکار مدت زمان کل و مدت زمان پخش شده
        public string MediaLen
        {
            get { return _mediaLen; }
            set
            {
                if (_mediaLen == value) return;
                _mediaLen = value;
                raisePropertyChanged("MediaLen");
            }
        }

        //این تصاویر در حین پخش باید تغییر کنند
```

```
       static BitmapImage pauseImg
{
    get
    {
        return new BitmapImage
        {
            UriSource =
                   new Uri("Images/debug-pause-icon.png",
                   UriKind.Relative)
        };
    }
}

//دریافت تصویر از منابع پروژه
static BitmapImage runImg
{
    get
    {
        return new BitmapImage
        {
            UriSource =
            new Uri("Images/debug-run-icon.png",
            UriKind.Relative)
        };
    }
}

#endregion Properties

#region Delegates and Events (1)

// Events (1)

public event PropertyChangedEventHandler PropertyChanged;

#endregion Delegates and Events

#region Methods (13)

// Private Methods (13)

//نمایش میزان پیشرفت پخش مدیا
void _timer_Tick(object sender, EventArgs e)
{
    seekBar.Value = mediaElement1.Position.TotalSeconds;
    if (mediaElement1.NaturalDuration.HasTimeSpan)
    {
        var remaningVal = mediaElement1.NaturalDuration.TimeSpan
                            - mediaElement1.Position;
```

```
            MediaLen = string.Format(
                "{0}:{1}:{2}",
                remaningVal.Hours.ToString("00"),
                remaningVal.Minutes.ToString("00"),
                remaningVal.Seconds.ToString("00"));
        }
        else
            Debug.WriteLine("!NaturalDuration.HasTimeSpan");
    }

    //تغییر اندازه‌ی صفحه‌ی نمایش
    private void btnExpand_Click(object sender, RoutedEventArgs e)
    {
        mediaElement1.Stretch =
            mediaElement1.Stretch == Stretch.UniformToFill
            ? Stretch.None
            : Stretch.UniformToFill;
    }

    //درخواست مکث یا شروع به کار
    //به همراه تغییر آیکون دکمه‌ها
    private void btnPlay_Click(object sender, RoutedEventArgs e)
    {
        if (!_isPlaying)
        {
            mediaElement1.Play();
            _isPlaying = true;
            imgPaly.Source = runImg;
        }
        else
        {
            mediaElement1.Pause();
            _isPlaying = false;
            imgPaly.Source = pauseImg;
        }
    }

    //درخواست توقف نمایش
    private void btnStop_Click(object sender, RoutedEventArgs e)
    {
        _isPlaying = false;
        mediaElement1.Stop();
        imgPaly.Source = runImg;
    }

    //نمایش طول مدت پخش یک فیلم یا مدیا
    string getMediaLen()
```

```
        {
    return !mediaElement1.NaturalDuration.HasTimeSpan
                ? "00:00:00"
                : string.Format("{0}:{1}:{2}",
  mediaElement1.NaturalDuration.TimeSpan.Hours.ToString("00"),
  mediaElement1.NaturalDuration.TimeSpan.Minutes.ToString("00"),
mediaElement1.NaturalDuration.TimeSpan.Seconds.ToString("00"));
}

//آغاز به  کار یک تایمر جهت بررسی میزان پیشرفت کار
private void initTimer()
{
    _timer = new DispatcherTimer {
            Interval = TimeSpan.FromMilliseconds(200) };
    _timer.Tick += _timer_Tick;
}

//فایل ویدیویی بارگذاری شده است
private void mediaElement1_Loaded(object sender,
             RoutedEventArgs e)
{
    _isPlaying = true;
    imgPaly.Source = pauseImg;
    imgPaly.ImageFailed += (send, er)
        => Debug.WriteLine(er.ErrorException.ToString());
}

//پایان کار پخش
private void mediaElement1_MediaEnded(object sender,
               RoutedEventArgs e)
{
    btnStop_Click(this, null);
}

//آیا مشکلی رخ داده است؟
private void mediaElement1_MediaFailed(object sender,
                ExceptionRoutedEventArgs e)
{
    Debug.WriteLine(e.ErrorException.ToString());
}

//شروع یک تایمر جهت بررسی میزان پیشرفت پخش فیلم
private void mediaElement1_MediaOpened(object sender,
                  RoutedEventArgs e)
{
    _timer.Stop();
    if (mediaElement1.NaturalDuration.HasTimeSpan)
    {
        var ts = mediaElement1.NaturalDuration.TimeSpan;
```

```
                seekBar.Maximum = ts.TotalSeconds;
                seekBar.SmallChange = 1;
                seekBar.LargeChange = Math.Min(10, ts.Seconds / 10);

                MediaLen = getMediaLen();
            }
            _timer.Start();
        }

        //جهت اعمال انقیاد داده‌ها و آگاه سازی خودکار
        void raisePropertyChanged(string propertyName)
        {
            var handler = PropertyChanged;
            if (handler == null) return;
            handler(this, new PropertyChangedEventArgs(propertyName));
        }

        //تغییر نقطه‌ی در حال پخش
        private void seekBar_ValueChanged(object sender,
            RoutedPropertyChangedEventArgs<double> e)
        {
            if (seekBar != null)
            {
                var fromSeconds = TimeSpan.FromSeconds(seekBar.Value);
                if (mediaElement1.Position.CompareTo(fromSeconds) != 0)
                    mediaElement1.Position = fromSeconds;
            }
        }

        //تغییر شدت صدای در حال پخش
        private void volumeSlider_ValueChanged(object sender,
            RoutedPropertyChangedEventArgs<double> e)
        {
            if (volumeSlider != null)
                mediaElement1.Volume = volumeSlider.Value;
        }

        #endregion Methods
    }
}
```

در این کدها بررسی مفصلی از توانایی‌های شیء MediaElement ارائه گردیده است (شروع، مکث و توقف نمایش؛ تغییر میزان صدا، امکان تغییر دستی نقطه‌ی در حال نمایش، نمایش میزان پیشرفت و غیره). همچنین به کمک یک تایمر میزان پیشرفت نمایش بررسی شده و به همین ترتیب مقدار seekBar برنامه مشخص و مقدار دهی می‌گردد.

اگر دقت کرده باشید این مثال با همان روش متداول تعریف روال‌های رخدادگردان در فایل Code behind یک View پیاده سازی شد. برای پیاده سازی کامل آن با کمک الگوی MVVM می‌توان از قابلیت EventToCommand تعریف شده در کتابخانه‌ی MVVM Light toolkit استفاده کرد (که ریز جزئیات بکارگیری آن در فصل مربوطه ارائه گردید)؛ زیرا امکان انقیاد مستقیم به رخدادهایی مانند ValueChanged، MediaEnded ، MediaOpened و امثال آن وجود ندارد و انقیاد به Commands که به صورت پیش فرض در Silverlight 4 مهیا است، تنها محدود به پیاده سازی اینترفیس ICommand می‌باشد.