Silverlight 4

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

فهرست مطالب

فصل ۶ – استفاده از کنترل‌های Silverlight

مقدمه

تا اینجا با اصول کاری XAML ، روش‌های مختلف طرح بندی و کار با رویدادها آشنا شدیم. اکنون زمان آن است که با المان‌ها و کنترل‌های مهیا در Silverlight جهت ساخت رابط‌های کاربری ساده و پیچیده، بیشتر آشنا گردیم.

در این فصل یک سری از کنترل‌های اصلی Silverlight و همچنین تعدادی از کنترل‌های Silverlight toolkit معرفی خواهند شد.

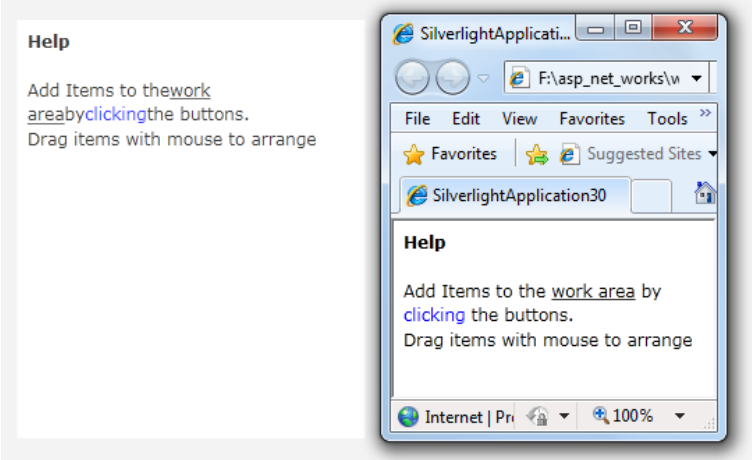
نمایش متن به کاربر

برای نمایش متون فقط خواندنی به کاربر از کنترل‌ها Label و TextBlock استفاده می‌شود. کنترل Label در فضای نام System.Windows.Controls قرار داشته و بیشتر جهت مباحث Data Binding کاربرد دارد که در طی فصول آینده به آن پرداخته شد. در ادامه در طی یک مثال، توانمندی‌های TextBlock ارائه می‌گردند:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication30.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="250">
    <Grid x:Name="LayoutRoot" Background="White" Margin="7">
     <TextBlock Margin="0" FontSize="12" TextWrapping='Wrap'>
       <Run FontWeight="Bold">Help</Run><LineBreak/><LineBreak/>
      Add Items to the <Run TextDecorations='Underline'>work area</Run>
      by <Run Foreground='Blue'>clicking</Run> the buttons.
       <LineBreak /> Drag items with mouse to arrange
     </TextBlock>
    </Grid>
</UserControl>
```

و خروجی این مثال در شکل ۱ نمایش داده شده است. در سمت چپ شکل ۱، خروجی طراح VS.NET را مشاهده می‌نمائید که متون را در هم فرو رفته نشان داده است. خروجی نهایی در مرورگر (همانند تصویر سمت راست شکل ۱) مشکلی نخواهد داشت.



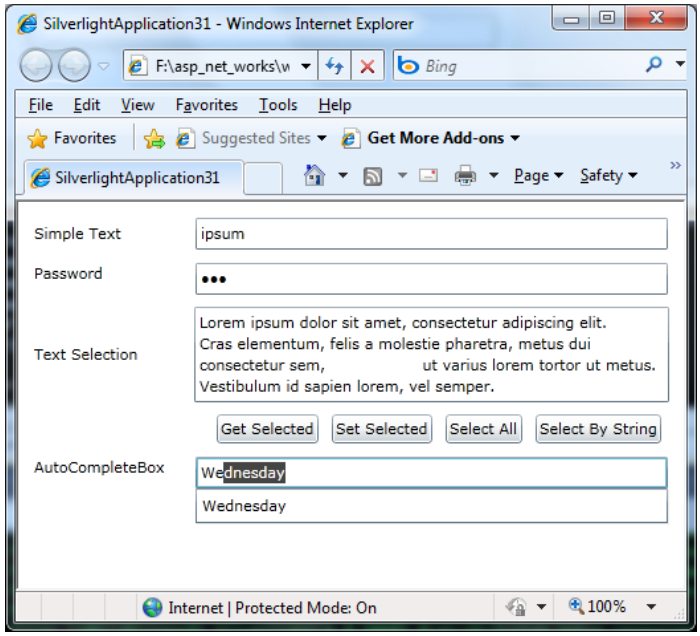
شکل ۱– استفاده از TextBlock جهت ارائه‌ی متون فقط خواندنی به کاربر

در این مثال با دو ویژگی مهم TextBlock آشنا شدیم : استفاده از تگ‌های Run (همانند تگ‌های Span در HTML) جهت تعیین قالب متن در برگیرنده‌ی آن مانند رنگ، ضخامت و غیره و همچنین استفاده از تگ LineBreak جهت نمایش ادامه‌ی متن در سطری دیگر (همانند تگ BR در HTML).

برای اعمال این تگ‌ها نیاز است تا متن ارائه شده مانند کدهای XAML قبل، بین تگ‌های TextBlock قرار گیرد. با تنظیم ویژگی TextWrapping به Wrap ، متن ارائه شده با اندازه‌های متفاوت مرورگر خود را تنظیم کرده و ادامه‌ی هر سطر را در سطر دیگر نمایش خواهد داد. در غیر اینصورت، ادامه‌ی سطر به صورت خودکار حذف گردیده و نمایش داده نخواهد شد. خاصیت دیگری به این کنترل در Silverlight 4 اضافه شده است به نام TextTrimming که جهت حذف ادامه‌ی یک سطر طولانی و برای مثال نمایش سه نقطه به صورت خودکار در پایان آن، کاربرد دارد.

دریافت متن از کاربر

سه کنترل مهم TextBox ، PasswordBox و AutoCompleteBox جهت دریافت ورودی‌های متنی از کاربر استفاده می‌شوند. در ادامه قصد داریم در طی یک مثال، این موارد را در عمل بررسی نمائیم (شکل ۲).



شکل ۲– نمایی از برنامه‌ی کنترل‌های دریافت متن از کاربر

استفاده از کنترل AutoCompleteBox نیاز به افزودن ارجاع‌هایی به اسمبلی‌های استاندارد System.Windows.Controls و System.Windows.Controls.Input دارد که به دو صورت می‌توان این ارجاعات را به برنامه افزود:

۱. استفاده از منوی Project → Add Reference
۲. و یا کشیدن کنترل AutoCompleteBox از جعبه ابزار VS.NET بر روی فرم برنامه که سبب افزوده شدن ارجاعات لازم به برنامه و همچنین فضاهای نام متناظر به فایل XAML جاری برنامه می‌شود.

کدهای XAML برنامه‌ی دریافت ورودی از کاربر را در ذیل مشاهده خواهید نمود. در این مثال نحوه‌ی تعریف و استفاده از سه کنترل TextBox ، PasswordBox و AutoCompleteBox جهت دریافت ورودی‌های متنی از کاربر به تفصیل بیان شده است:

MainPage.xaml

```
<UserControl
    xmlns:
    my="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Input"
    x:Class="SilverlightApplication31.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="500">
<Grid x:Name="LayoutRoot"Margin="7">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="25*" />
        <ColumnDefinition Width="75*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto"  />
        <RowDefinition Height="*"  />
    </Grid.RowDefinitions>

    <TextBlock HorizontalAlignment="Left"
            VerticalAlignment="Center"
            Text="Simple Text"
            Grid.Row="0"
            Margin="5" />
    <TextBox x:Name="simpleTextBox"
            Text=""
            Grid.Row="0"
            Grid.Column="1"
            Margin="5" />

    <TextBlock HorizontalAlignment="Left"
           VerticalAlignment="Top"
            Text="Password"
            Grid.Row="1"
            Margin="5" />
    <PasswordBox x:Name="passwordBox"
            Grid.Row="1"
            Grid.Column="1"
            Margin="5"
            Grid.RowSpan="1"
           VerticalAlignment="Top" />

    <TextBlock HorizontalAlignment="Left"
            VerticalAlignment="Center"
            Text="Text Selection"
            Grid.Row="2"
            Margin="5" />
    <TextBox x:Name="selectionTextbox"
            Grid.Row="2"
            Grid.Column="1"
            Margin="4"
            AcceptsReturn="True"
```

```
            Text="Lorem ipsum dolor sit amet, consectetur adipiscing elit.
   Cras elementum, felis a molestie pharetra, metus dui consectetur sem,
            ut varius lorem tortor ut metus.
            Vestibulum id sapien lorem, vel semper."
               TextWrapping="Wrap" />
        <StackPanel Margin="5"
              Grid.Column="1"
              Grid.Row="3"
              Orientation="Horizontal"
              d:LayoutOverrides="Width"
              HorizontalAlignment="Right">

            <Button x:Name="getSelectedButton"
                      Content="Get Selected"
                      Click="getSelectedButton_Click"
                      Margin="5,0" />
            <Button x:Name="setSelectedButton"
                      Content="Set Selected"
                      Click="setSelectedButton_Click"
                      Margin="5,0" />
            <Button x:Name="selectAllButton"
                      Content="Select All"
                      Click="selectAllButton_Click"
                      Margin="5,0" />
            <Button x:Name="selectByStringButton"
                      Content="Select By String"
                      Click="selectByStringButton_Click"
                      Margin="5,0" />
        </StackPanel>
        <my:AutoCompleteBox Margin="5"
               x:Name='autoCompleteBox'
               Grid.Column="1"
               Grid.Row="4"
               Height='Auto'
               FilterMode='StartsWith'
               IsTextCompletionEnabled="True"
             VerticalAlignment="Top" />
        <TextBlock HorizontalAlignment="Left"
               VerticalAlignment="Top"
               Text="AutoCompleteBox"
               Grid.Row="4"
               Margin="5" />
    </Grid>
</UserControl>
```

و کدهای متناظر برنامه‌ی دریافت متن از کاربر به شرح بعد هستند:

MainPage.xaml.cs

```
using System.Collections.Generic;
using System.Windows;
namespace SilverlightApplication31
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            var days = new List<string>
            {
             "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"
            };
            autoCompleteBox.ItemsSource = days;
        }
        private void getSelectedButton_Click(object sender,
               RoutedEventArgs e)
        {
            string selected = selectionTextbox.SelectedText;
            simpleTextBox.Text = selected;
        }
        private void setSelectedButton_Click(object sender,
            RoutedEventArgs e)
        {
            selectionTextbox.Focus();
            selectionTextbox.Select(12, 30);
        }

        private void selectAllButton_Click(object sender,
             RoutedEventArgs e)
        {
            selectionTextbox.Focus();
            selectionTextbox.SelectAll();
        }

        private void selectByStringButton_Click(object sender,
            RoutedEventArgs e)
        {
            selectionTextbox.Focus();
            if (selectionTextbox.SelectionLength > 0)
            {
                selectionTextbox.SelectedText = "dolor sit amet";
            }
        }
    }
}
```

توضیحات:

دو کنترل TextBox و PasswordBox جزو متداول ترین کنترل‌های دریافت ورودی متنی از کاربر می‌باشند. مثالی پیشرفته از بکارگیری آن‌ها، در TextBox سوم قرار گرفته بر روی فرم بیان شده است. با تنظیم ویژگی AcceptsReturn آن به True امکان ورود چند سطری اطلاعات فراهم گردیده است (امکان استفاده از دکمه‌ی Enter برای وارد کردن سطر بعدی اطلاعات). همچنین ویژگی TextWrapping آن به Wrap تنظیم گردیده است تا ادامه‌ی سطرهای طولانی، در سطر بعدی مشخص شوند.

در ادامه، چهار دکمه بر روی فرم قرار گرفته‌اند که کار آن‌ها دریافت متن انتخاب شده در TextBox سوم (استفاده از خاصیت SelectedText)، انتخاب قسمتی از متن (استفاده از متد Select) و یا انتخاب کل متن (به کمک متد SelectAll) می‌باشد.

کنترل AutoCompleteBox بر اساس یک سری واژه‌ی از پیش تعریف شده ، کار پر کردن خودکار TextBox را در حین ورود متن توسط کاربر، انجام می‌دهد. این واژه‌ها توسط یک Generic List رشته‌ای به نام day در متد سازنده‌ی کلاس، به خاصیت ItemsSource کنترل AutoCompleteBox برنامه انتساب داده شده است. بدیهی است این اطلاعات را می‌توان از یک بانک اطلاعاتی و یا یک Web service نیز دریافت کرد.

استفاده از قلم‌های متفاوت

هر المانی در Silverlight که توانایی نمایش متون را داشته باشد، از ویژگی Font نیز برخوردار است که توسط آن می‌توان نوع قلم، ضخامت، اندازه و بسیاری خواص دیگر را نیز تنظیم نمود. اما نکته‌ای را که باید در اینجا به آن دقت داشت محدودیت تعداد قلم‌های استاندارد آن است. این قلم‌های از پیش مهیا شامل Arial ، Arial Black ، Comic Sans MS ، Lucia ، Times New Roman ، Trebuchet MS و Verdana می‌باشند. در صورت استفاده از سایر قلم‌ها، باید آن‌ها را در برنامه‌ی خود مدفون (Embed) نمود. برای مدفون سازی قلم‌ها به صورت زیر می‌توان عمل کرد:

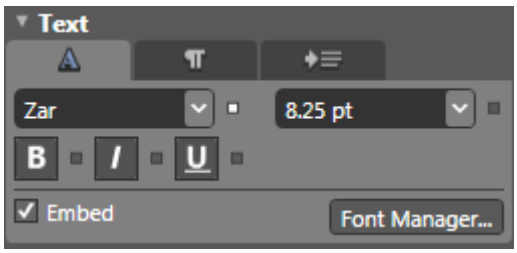
۱. یک پوشه‌ی جدید به نام fonts به پروژه اضافه کنید.
۲. سپس قلم مورد نظر خود را در این پوشه کپی کرده و به پروژه اضافه نمائید (کلیک راست بر روی پوشه، انتخاب Add → New Item)
۳. به خواص این فونت در VS.NET مراجعه نموده و از تنظیم خاصیت Build Action آن به Resource اطمینان حاصل نمائید.

سپس برای استفاده از این فونت در برنامه تنها کافی است خاصیت FontFamily به صورت FileName#fontName مقدار دهی گردد که این مورد را در مثال بعد می‌توان مشاهده نمود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication32.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" FlowDirection="RightToLeft">
        <StackPanel>
            <StackPanel.Background>
                <LinearGradientBrush EndPoint="0.5,1"
                                     StartPoint="0.5,0">
                    <GradientStop Color="#FFF2DF69" Offset="0.869" />
                    <GradientStop Color="#FFC1C197" Offset="0.057" />
                </LinearGradientBrush>
            </StackPanel.Background>
            <TextBlock FontFamily="fonts/zar.ttf#zar" Margin="10"
                       HorizontalAlignment="Center" FontSize="30">
            بررسی قلم زر در سیلورلایت
            </TextBlock>
        </StackPanel>
    </Grid>
</UserControl>
```

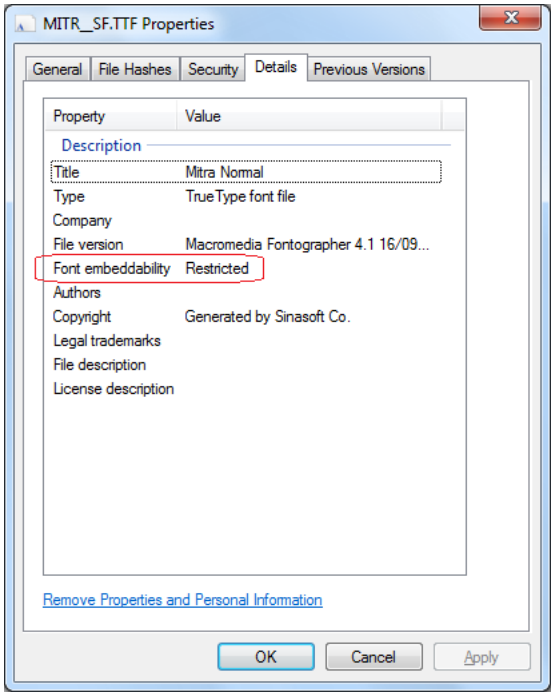
مدفون سازی قلم‌ها توسط برنامه‌ی MS Expression Blend بسیار ساده است. برای این منظور تنها کافی است قلم مورد استفاده انتخاب گردیده و سپس گزینه‌ی Embed انتخاب گردد (شکل ۳). سایر موارد به صورت خودکار صورت خواهند گرفت.



شکل ۳– استفاده از Expression Blend جهت مدفون سازی قلم مورد استفاده

نکته‌ای در مورد مجوزهای قلم‌های گوناگون

مجاز به مدفون نمودن تمامی قلم‌های در دسترس نیستید. برای مشاهده‌ی مجوز قلم‌ها، به قسمت Fonts در کنترل پنل ویندوز مراجعه نموده ، روی فونت مورد نظر کلیک راست کرده و به صفحه‌ی خواص آن مراجعه نمائید. در برگه‌ی جزئیات قلم، خاصیت Font embeddability آن قلم مورد نظر نباید Restricted و یا محدود شده باشد (شکل ۴).



شکل ۴– امکان مدفون نمودن قلم Mitra مطابق مجوز آن، محدود شده است.

اگر علاقمند باشید که با استفاده از کد نویسی از قلم‌های مدفون شده استفاده نمائید باید به نحو ذیل عمل کرد:

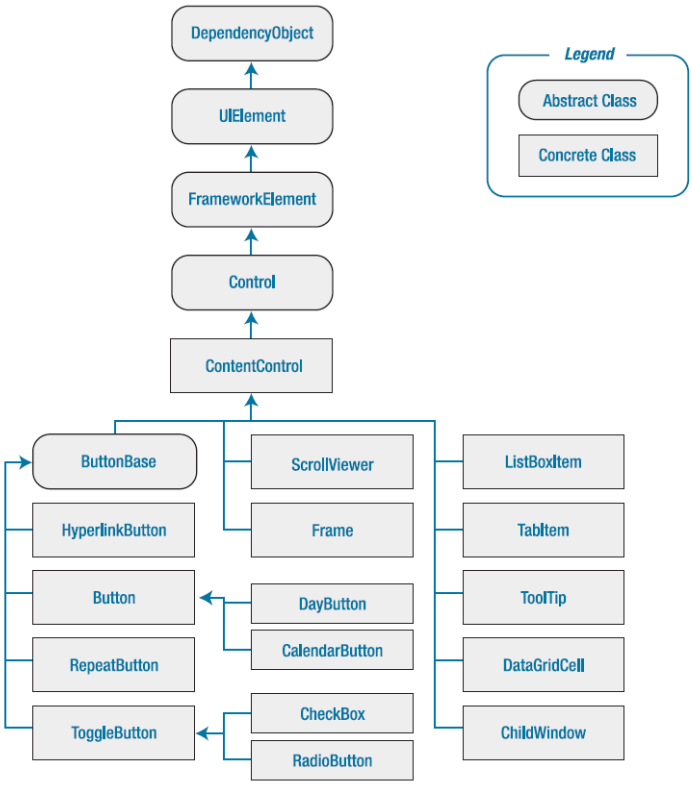
C#

```
//Format = AssemblyName;component/FontResourceName
StreamResourceInfo sri =
     Application.GetResourceStream(
     new Uri("FontTest;component/BayernFont.ttf", UriKind.Relative));
lbl.FontSource = new FontSource(sri.Stream);
lbl.FontFamily = new FontFamily("Bayern");
```

ابتدا باید مراحل اضافه کردن فونت به صورت resource به برنامه انجام شده و سپس از کدهای فوق استفاده گردد.

ایجاد کنترل‌های ترکیبی در Silverlight

تعداد زیادی از کنترل‌های Silverlight از کلاس ContentControl مشتق می‌شوند (شکل ۵). این مورد بدین معنا است که تمامی اینگونه کنترل‌ها می‌توانند شامل یک المان فرزند نیز باشند. بدیهی است که این المان فرزند را می‌توان به صورت یکی از اشیاء طرح بندی، مانند StackPanel نیز معرفی کرد و به این ترتیب چندین شیء را به عنوان محتوای یک کنترل ارائه داد. به این صورت دیگر لازم نیست تا طراحان، منتظر نگارش بعدی Silverlight باشند که برای مثال در آن امکان قرار دادن تصویر در یک کنترل دکمه اضافه شده است. طراح به سادگی می‌تواند چندین المان فرزند را به کمک سیستم‌های طرح بندی Silverlight به یک کنترل دکمه‌ی ساده اضافه نموده و کنترلی پیچیده را خلق نماید. به این ترتیب نیاز طراحان Silverlight به مجموعه‌ای از کامپوننت‌های تجاری شرکت‌های دیگر جهت بهبود ظاهر کنترل‌های استاندارد موجود، به شدت کاهش می‌یابد. در ادامه مثالی را در این زمینه مرور خواهیم کرد.

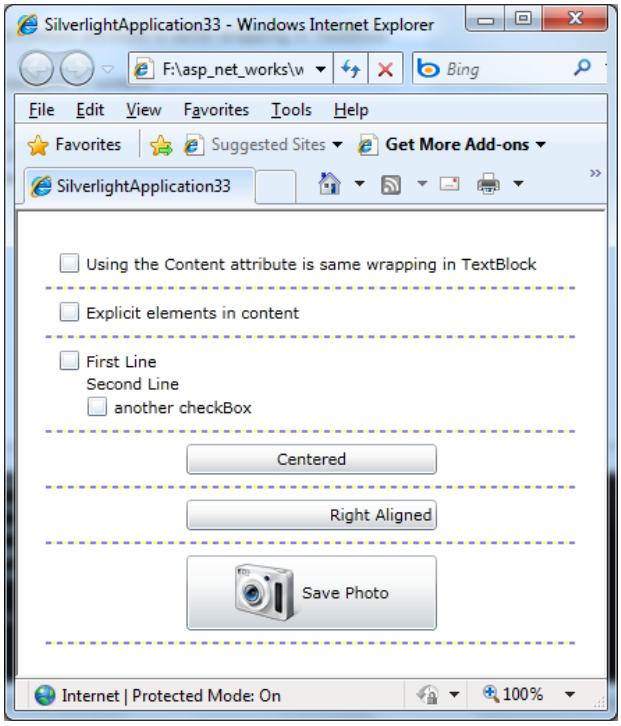


شکل ۵– سلسله مراتب کنترل‌هایی که می‌توانند دربرگیرنده‌ی کنترل‌های دیگر نیز باشند.

در ادامه مثالی را در مورد ترکیب کنترل‌ها با یکدیگر مرور خواهیم کرد (شکل ۶). کدهای XAML این مثال در ادامه ذکر شده‌اند. هدف از این مثال نمایش توانایی‌های Silverlight در ترکیب کنترل‌های مختلف است و هدف آن نیست که برای مثال در برنامه‌های خود CheckBox را با TextBox ترکیب کنید و کنترل جدیدی را به این شکل خلق نمائید!

در این مثال ابتدا ویژگی Content یک CheckBox مقدار دهی شده است، سپس در CheckBox بعدی جهت نمایش متن منتسب به CheckBox از یک TextBlock استفاده گردیده است که از لحاظ ظاهری نمای یکسانی را با روش قبل ارائه می‌دهد. سپس در CheckBox بعدی با توجه به اینکه هر ContentControl یک فرزند بیشتر نمی‌تواند داشته باشد، از یک StackPanel جهت ارائه‌ی چندین فرزند کمک گرفته شده است.

در دو دکمه‌ای که بر روی فرم قرار گرفته‌اند، محل قرارگیری TextBlock واقع شده در آن‌ها به صورت صریح مشخص شده است و در آخرین دکمه‌ی قرار گرفته بر روی صفحه، یک دکمه‌ی ترکیبی از متن و تصویر ایجاد گردیده است:



شکل ۶– نمایی از ترکیب کنترل‌ها در Silverlight

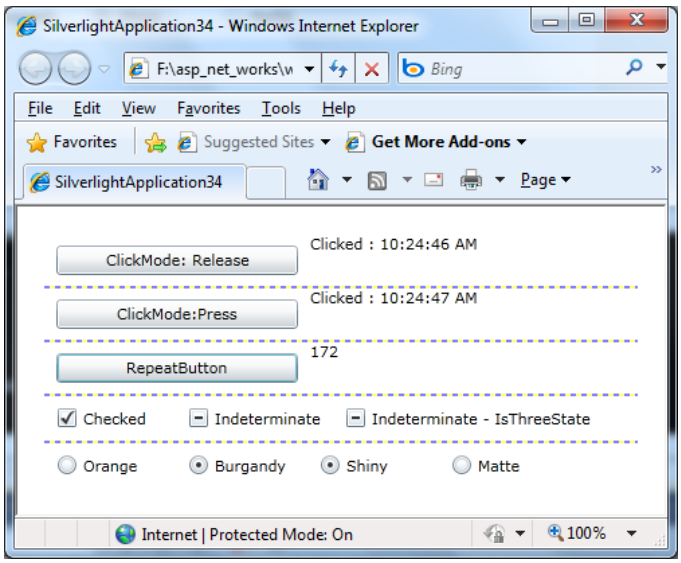
MainPage.xaml

```
<UserControl x:Class="SilverlightApplication33.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" Margin='20' ShowGridLines='True'>
        <Grid.RowDefinitions>
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
```

```
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
        </Grid.RowDefinitions>
        <!-- simple textblock as content-->
        <CheckBox  Margin='9'
        Content='Using the Content attribute is same wrapping in TextBlock'>
        </CheckBox>
        <!-- explicit textblock-->
        <CheckBox  Margin='9' Grid.Row='1'>
            <TextBlock>Explicit elements in content</TextBlock>
        </CheckBox>
        <!-- only single child allowed as content -->
        <CheckBox  Margin='9' Grid.Row='2'>
            <StackPanel>
                <TextBlock Text='First Line' />
                <TextBlock Text='Second Line' />
                <CheckBox Content="another checkBox"  />
            </StackPanel>
        </CheckBox>
        <!-- content alignment-->
        <Button Grid.Row='3' Margin='9' Width='180'
       HorizontalContentAlignment='Center'>
            <TextBlock Text='Centered' />
        </Button>
        <Button Grid.Row='4' Margin='9' Width='180'
       HorizontalContentAlignment='Right'>
            <TextBlock Text='Right Aligned' />
        </Button>
        <!-- customize content with images or vector art -->
        <Button Grid.Row='5'
                     Margin='9'
                     Width='180'
                     HorizontalAlignment="Center"
                     VerticalAlignment="Center">
            <StackPanel Orientation="Horizontal">
                <Image Height="48" Source="images/Camera.png"
                       Stretch="Fill" />
                <TextBlock Text='Save Photo'
        HorizontalAlignment="Center"
        VerticalAlignment="Center" />
            </StackPanel>
        </Button>
    </Grid>
</UserControl>
```

انواع دکمه‌ها در Silverlight

در Silverlight ، CheckBox و RadioButton نیز جزو خانواده‌ی دکمه‌ها محسوب می‌شوند. در ادامه قصد داریم توسط یک مثال (شکل ۷)، کنترل‌های دکمه، وضعیت‌های منتسب به آن‌ها، کنترل RepeatButton ، کنترل‌های CheckBox و RadioButton و نکات مرتبط به آن‌ها را بررسی نمائیم.



شکل ۷– نمایی از بررسی برنامه‌ی انواع دکمه‌ها در Silverlight

کدهای XAML این مثال را در ادامه مشاهده می‌نمائید. نکته‌ی جدیدی که در مورد کنترل‌های استاندارد دکمه در Silverlight مشاهده خواهید نمود، ویژگی ClickMode آن‌ها است. اگر این ویژگی به Release تنظیم شود (همانند دکمه‌ی اول مثال)، تنها پس از رها شدن دکمه‌ی سمت چپ Mouse ، روال رویدادگردان آن فراخوانی می‌گردد. اگر مانند دکمه‌ی دوم قرار گرفته بر روی فرم، ویژگی ClickMode به Press تنظیم شود، روال رویدادگردان مرتبط با آن، بلافاصله پس از فشرده شده دکمه سمت چپ Mouse عمل خواهد کرد.

دکمه‌ی جدیدی به نام RepeatButton به Silverlight اضافه شده است که کار آن فراخوانی پی در پی روال رویدادگردان مرتبط زمانیکه دکمه‌ی چپ Mouse فشرده می‌شود، می‌باشد (دکمه‌ی سوم قرار گرفته بر روی فرم). توسط خاصیت‌های Delay و Interval این کنترل مشخص خواهیم ساخت که کار فراخوانی پی در پی روال رویدادگران آن پس از چه تاخیری از زمان فشرده شدن دکمه‌ی سمت چپ Mouse شروع شود و همچنین در چه فواصل زمانی نیز فراخوانی گردد.

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication34.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="600">
    <Grid x:Name="LayoutRoot"
        Margin='20'
        ShowGridLines='True'>
        <Grid.RowDefinitions>
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
        </Grid.RowDefinitions>

        <!--  default click behavior is to press and release -->
        <StackPanel Orientation='Horizontal'
                Grid.Row='0'>
            <Button
              ClickMode="Release"
              Content='ClickMode: Release'
              Grid.Row='0' MinWidth='180' Margin='9'
              x:Name='btn1'
              Click='btn1_Click' />
            <TextBlock x:Name='tb1' />
        </StackPanel>
        <!--  this causes the click to fire on mouse down -->
        <StackPanel Orientation='Horizontal'
                Grid.Row='1'>
            <Button
              ClickMode="Press"
              Content='ClickMode:Press'
              MinWidth='180' x:Name='btn2' Margin='9'
              Click='btn2_Click' />
            <TextBlock x:Name='tb2' />
        </StackPanel>

        <StackPanel Orientation='Horizontal'
                Grid.Row='2'>
            <RepeatButton
              Delay="1000"
              Interval="50"
              Content='RepeatButton'
              MinWidth='180' x:Name='btn3' Margin='9'
```

```
                Click='btn3_Click' />
            <TextBlock x:Name='tb3' />
        </StackPanel>

        <StackPanel Orientation='Horizontal'
                Grid.Row='3'>
            <CheckBox Content='Checked'
                IsChecked='True'
                MinWidth='80' x:Name='btn4' Margin='9' />
            <CheckBox Content='Indeterminate'
                IsChecked='{x:Null}'
                MinWidth='80' x:Name='btn5' Margin='9' />
            <!-- allow the user to choose the inderminate state with IsThreeState -->
            <CheckBox Content='Indeterminate - IsThreeState'
                IsChecked='{x:Null}'
                IsThreeState='True'
                MinWidth='80' x:Name='btn6' Margin='9' />
        </StackPanel>
        <StackPanel Orientation='Horizontal'
                Grid.Row='4'>
            <!-- use GroupName to link radio buttons together.-->

            <RadioButton Content='Orange'
                   IsChecked='False'
                   GroupName='TrimColor'
                   MinWidth='80' Margin='9' />
            <RadioButton Content='Burgandy'
                   IsChecked='True'
                   GroupName='TrimColor'
                   MinWidth='80' Margin='9' />

            <RadioButton Content='Shiny'
                   IsChecked='True'
                   GroupName='TrimFinish'
                   MinWidth='80' Margin='9' />

            <RadioButton Content='Matte'
                   IsChecked='False'
                   GroupName='TrimFinish'
                   MinWidth='80' Margin='9' />
        </StackPanel>
    </Grid>
</UserControl>
```

در ردیف چهارم این مثال، سه CheckBox بر روی صفحه قرار گرفته‌اند. نکته‌ی جالب CheckBox ها در Silverlight ، امکان انتساب Null به خاصیت IsChecked آن‌ها است. در این حالت وضعیت یک CheckBox نامشخص بوده و انتخاب شده یا انتخاب نشده فرض نخواهد شد. اگر خاصیت IsThreeState یک CheckBox

به True تنظیم شود، یک کاربر می‌تواند هر سه حالت انتخاب شده، انتخاب نشده و نامعین را انتخاب و تنظیم نماید.

تنها نکته‌ی مهم RadioButton ها، تعیین ویژگی GroupName آن‌ها می‌باشد. اگر دو یا چند RadioButton در یک گروه قرار گیرند، تنها می‌توان یکی از آن‌ها را در این گروه انتخاب نمود.

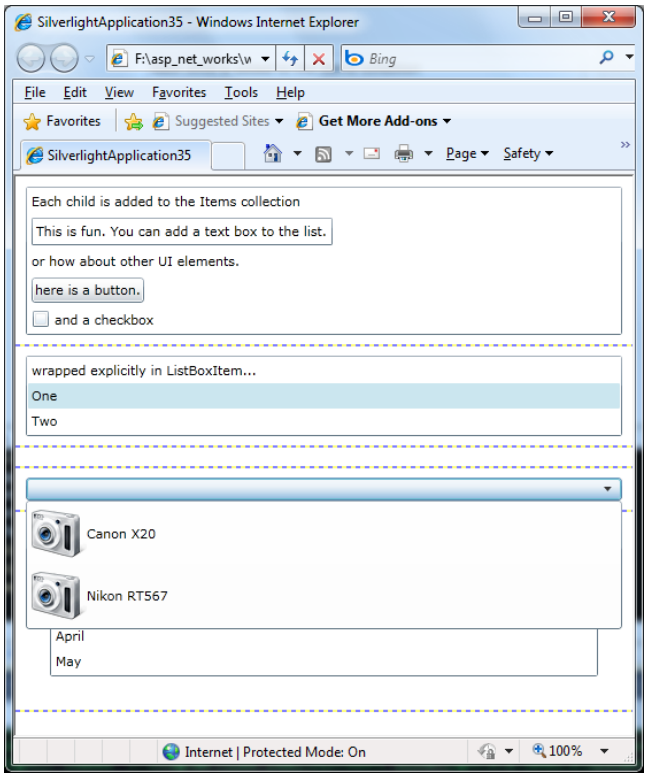
و کدهای متناظر با این فایل XAML در ذیل ذکر شده‌اند:

MainPage.xaml.cs

```
using System;
using System.Windows;

namespace SilverlightApplication34
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void btn1_Click(object sender, RoutedEventArgs e)
        {
            tb1.Text = "Clicked : " + DateTime.Now.ToLongTimeString();
        }

        private void btn2_Click(object sender, RoutedEventArgs e)
        {
            tb2.Text = "Clicked : " + DateTime.Now.ToLongTimeString();
        }

        long _counter;
        private void btn3_Click(object sender, RoutedEventArgs e)
        {
            ++_counter;
            tb3.Text = _counter.ToString();
        }
    }
}
```

معرفی ItemControls

همواره در انواع برنامه‌ها لیستی از اطلاعات، جهت نمایش به کاربران وجود دارد. در Silverlight کنترل‌هایی که برای نمایش اینگونه لیست‌ها تدارک دیده شده‌اند، ItemControls نام دارند و شامل ListBox ، ComboBox و موارد دیگری که در Silverlight toolkit تعریف شده‌اند، می‌باشند. در مثال بعد قصد داریم به بررسی روش‌های متفاوت مقدار دهی ListBox و ComboBox بپردازیم (شکل ۸).



شکل ۸– نمایی از برنامه‌ی معرفی ItemControls

کدهای XAML این مثال را در ادامه ملاحظه می‌فرمائید. در ابتدا یک ListBox در صفحه قرار گرفته است و آیتم‌های مختلف آن به صورت مستقیم در XAML تعریف شده‌اند. این آیتم‌ها هر نوع UIElement معتبری می‌توانند باشند. در ListBox دوم، با کمک ListBoxItem ، هر کدام از آیتم‌های مورد نظر اضافه شده‌اند. مزیت این روش امکان اعمال یک سری از رفتارها و ویژگی‌ها مانند تعیین آیتم انتخاب شده است.

در ادامه یک ComboBox در صفحه تعریف شده است که لیستی از دوربین‌های موجود را به همراه یک آیکون در کنار آن‌ها نمایش می‌دهد. همانطور که ملاحظه می‌کنید جهت تعریف هر آیتم از یک StackPanel برای ترکیب

اشیاء مورد نظر کمک گرفته شده است. در پایان فرم، دو ListBox بر روی صفحه قرار گرفته‌اند که هدف از ارائه‌ی آن‌ها ، معرفی روش مقدار دهی این نوع کنترل‌ها در کدهای صفحه می‌باشد.

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication35.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" ShowGridLines='True'
                            Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
            <RowDefinition Height='Auto' />
        </Grid.RowDefinitions>
        <ListBox Margin='9' Grid.Row='0'>
            <!-- add children directly-->
     <TextBlock Text='Each child is added to the Items collection' />
     <TextBox Text='This is fun. You can add a text box to the list.' />
     <TextBlock Text='or how about other UI elements.' />
     <Button Content='here is a button.' />
     <CheckBox Content='and a checkbox' />
           <!-- and you can add other UI elements here...-->
       </ListBox>
       <ListBox Margin='9' Grid.Row='1' x:Name='lb2'>
           <!-- All items in the ListBox.Items collection are
                        wrapped in a ListBoxItem. -->
           <!-- ListBoxItem adds behavior to contained items,
                        example:IsSelected -->
           <ListBoxItem>
            <TextBlock Text='wrapped explicitly in ListBoxItem...' />
           </ListBoxItem>
           <ListBoxItem IsSelected='True'>
               <TextBlock Text='One' />
           </ListBoxItem>
           <ListBoxItem>
               <TextBlock Text='Two' />
           </ListBoxItem>
       </ListBox>
       <ComboBox Grid.Row='3' Margin='9'>
```

```
                <StackPanel Orientation='Horizontal'>
                <Image Source='images/camera.png' />
                <TextBlock Text='Canon X20' VerticalAlignment='Center'/>
            </StackPanel>
            <StackPanel Orientation='Horizontal'>
                <Image Source='images/camera.png' />
                <TextBlock Text='Nikon RT567'
                   VerticalAlignment='Center' />
            </StackPanel>
        </ComboBox>
        <TextBlock x:Name='tbResult' Grid.Row='2' Margin='9' />
        <ListBox Margin='30' Grid.Row='4' x:Name='listBox1' />
        <ListBox Margin='30' Grid.Row='5' x:Name='listBox2' />
    </Grid>
</UserControl>
```

و کدهای متناظر با این صفحه‌ی XAML جهت مقدار دهی عناصر listBox1 و listBox2 به شرح ذیل می‌باشند:

MainPage.xaml.cs

```
using System.Collections.Generic;
using System.Windows;

namespace SilverlightApplication35
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            this.Loaded += FromCollectionInCode_Loaded;
        }

        void FromCollectionInCode_Loaded(object sender,
            RoutedEventArgs e)
        {
            listBox1.Items.Add("January");
            listBox1.Items.Add("February");
            listBox1.Items.Add("March");
            listBox1.Items.Add("April");
            listBox1.Items.Add("May");

            var days = new List<string>
                           {
                             "Monday", "Tuesday", "Wednesday",
                             "Thursday", "Friday", "Saturday", "Sunday"
                           };
           listBox2.ItemsSource = days;
        }
    }
}
```

در این کدها دو روش مختلف مقدار دهی عناصر ListBox ها را ملاحظه می‌فرمائید. ابتدا از متد Add کلاس Items استفاده شده و در روش دوم یک لیست به خاصیت ItemsSource کنترل ListBox دوم انتساب داده شده است.

معرفی کنترل‌های Silverlight toolkit

همانطور که در فصل‌های قبل نیز ذکر گردید، Silverlight toolkit مجموعه‌ای سورس باز و همچنین رایگان قرار گرفته در آدرس http://silverlight.codeplex.com می‌باشد که توسط تیم اصلی Silverlight توسعه یافته است. این مجموعه شامل تعداد زیادی کنترل همانند TreeView ، TabControl ، Chart Controls ، DataGrid ، تقویم و بسیاری موارد دیگر می‌باشد. هدف از ارائه‌ی این کنترل‌ها به صورت سورس باز، دریافت بازخورد کاربران و آزمودن آن‌ها پیش از یکپارچگی با نگارش‌های اصلی بعدی Silverlight است که یکی از روش‌های موفق توسعه‌ی نرم افزار به شمار می‌رود. بهترین روش آشنایی با کنترل‌های این مجموعه، مراجعه به مسیر ذیل در ویندوز است:

Start → All programs → Microsoft Silverlight 4 Toolkit → Toolkit samples

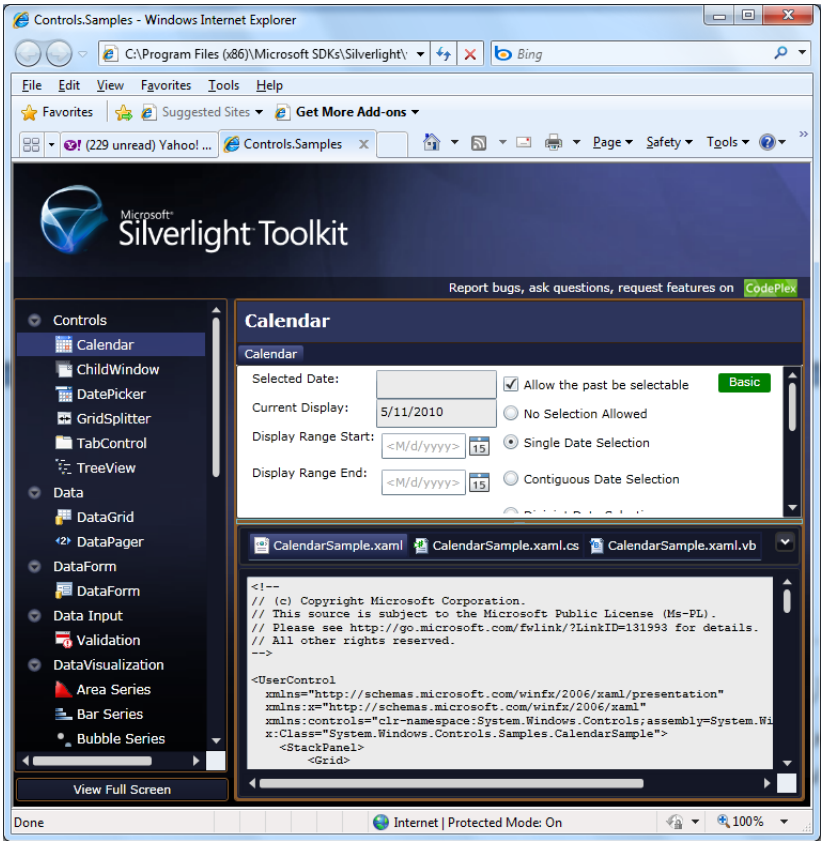
پس از آن با یک برنامه‌ی زیبای تهیه شده با Silverlight توسط مایکروسافت جهت معرفی مشروح انواع و اقسام کنترل‌های مهیا، به همراه مثال‌هایی به زبان‌های C# و VB.NET مواجه خواهیم شد (شکل ۹).

پس از نصب این مجموعه، کنترل‌های آن به صورت خودکار به کنترل‌های استاندارد جعبه ابزار (Toolbox) VS.NET اضافه می‌شوند. ساده‌ترین روش برای استفاده از آنها نیز کشیدن آن‌ها از جعبه ابزار و رها کردن آن‌ها بر روی فرم برنامه می‌باشد. به این صورت ابتدا ارجاعی به اسمبلی‌های مورد نیاز به برنامه اضافه شده و همچنین فضای نام مرتبط نیز به فایل XAML جاری برنامه به صورت خودکار افزوده خواهد شد. بدیهی است بهترین روش برای استفاده از این مجموعه فعال کردن گزینه ذیل در تنظیمات پروژه است که در فصل‌های قبل نیز به آن اشاره گردید:

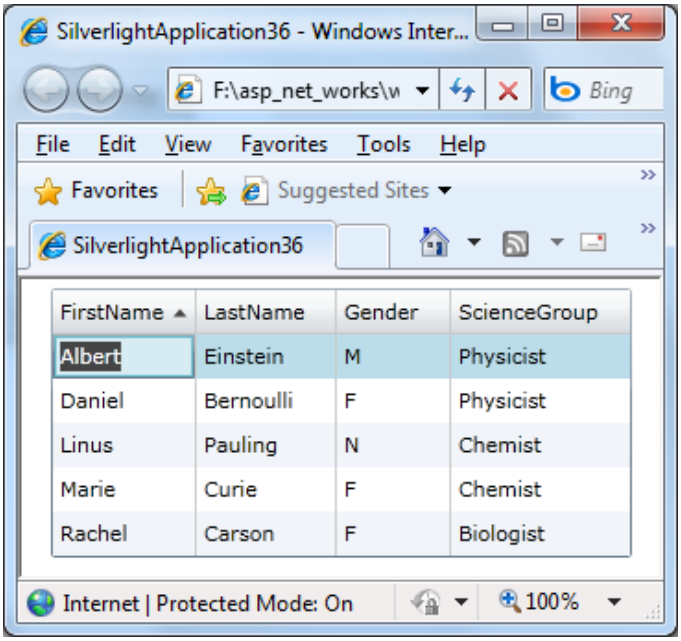
Reduce XAP size by using application library caching

به این ترتیب کاربران هر بار مجبور نخواهند شد تا اسمبلی‌های مرتبط با Silverlight toolkit را دریافت کنند و پس از اولین بار دریافت آن‌ها توسط مرورگر کاربر، Cache خواهند شد.

با تعداد زیادی از این کنترل‌ها در طی فصول قبلی آشنا شده‌ایم؛ مانند سیستم‌های طرح بندی WrapPanel ، کنترل AutoCompleteBox و موارد دیگر. در ادامه قصد داریم نگاهی داشته باشیم به توانایی‌های مقدماتی کنترل بسیار پرکاربرد DataGrid (شکل ۱۰). DataGrid یکی از کنترل‌هایی است در نگارش‌های اخیر Silverlight از Silverlight toolkit به مجموعه‌ی SDK آن نیز منتقل شده است.



شکل ۹– مرورگر مثال‌های متنوع مجموعه‌ی Silverlight toolkit مایکروسافت



شکل ۱۰– نمایی از مثال اولین کاربرد DataGrid

جهت پیاده سازی مثال اولین کاربرد DataGrid برای نمایش لیستی از مشخصات افراد، ابتدا پوشه‌ی Model را به برنامه افزوده و سپس دو فایل Person.cs و Persons.cs را مطابق کدهای بعد به این پوشه اضافه نمائید. به این طریق منبع داده‌ای مورد نیاز جهت Binding به DataGrid تعریف و تهیه خواهد شد.

Person.cs

```
namespace SilverlightApplication36.Model
{
    public class Person
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Gender { get; set; }
        public string ScienceGroup { get; set; }
    }
}
```

Persons.cs

```
using System.Collections.Generic;

namespace SilverlightApplication36.Model
{
    public class Persons : List<Person>
    {
        public Persons()
        {
            this.Add(new Person
                        {
                            FirstName = "Albert",
                            LastName = "Einstein",
                            Gender = "M",
                            ScienceGroup = "Physicist"
                        });
            this.Add(new Person
                        {
                            FirstName = "Marie",
                            LastName = "Curie ",
                            Gender = "F",
                            ScienceGroup = "Chemist"
                        });
            this.Add(new Person
                        {
                            FirstName = "Daniel ",
                            LastName = "Bernoulli",
                            Gender = "F",
```

```
                                ScienceGroup = "Physicist"
                        });

            this.Add(new Person
                        {
                            FirstName = "Rachel",
                            LastName = "Carson",
                            Gender = "F",
                            ScienceGroup = "Biologist"
                        });
            this.Add(new Person
                        {
                            FirstName = "Linus",
                            LastName = "Pauling",
                            Gender = "M",
                            ScienceGroup = "Chemist"
                        });
        }
    }
}
```

تا اینجا کار تعریف Model برنامه به پایان می‌رسد. اکنون نوبت به تعریف آن در فایل XAML و انقیاد (Bind) آن به DataGrid می‌باشد. برای این منظور ابتدا کنترل DataGrid را از جعبه ابزار VS.NET کشیده و بر روی فرم رها کنید تا ارجاعات لازم به اسمبلی‌های مورد نیاز و همچنین فضای نام مربوطه در فایل XAML جاری اضافه شوند. سپس نیاز است تا کلاس Persons را بتوان به صورت یک منبع داده معرفی کرد. بنابراین نیاز است تا فضای نام در برگیرنده‌ی آن (local در اینجا) به فایل XAML افزوده شود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication36.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
     "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400"
    xmlns:sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk"
    xmlns:local="clr-namespace:SilverlightApplication36.Model">
    <UserControl.Resources>
        <local:Persons x:Key="PersonsDataSource"/>
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White">
        <sdk:DataGrid AutoGenerateColumns="True"
                      HorizontalAlignment="Center"
                      Margin="5"
```

```
                    Name="dataGrid1"
                    VerticalAlignment="Top"
                    ItemsSource=
                "{Binding Source={StaticResource PersonsDataSource}}"
                />
    </Grid>
</UserControl>
```

نحوه‌ی تعریف یک StaticResource را در قسمت UserControl.Resources فوق ملاحظه می‌نمائید. اکنون جهت انقیاد آن به خاصیت ItemsSource متعلق به DataGrid ، به خواص این کنترل در VS.NET مراجعه کرده و همانند شکل بعد، در برگه‌ی Source ابتدا UserControl.Resources را از قسمت StaticResource انتخاب نموده و سپس PersonsDataSource ظاهر شده را با دوبار کلیک بر روی آن انتخاب نمائید. به این صورت سطر مربوط به تعریف Binding به صورت خودکار ایجاد خواهد شد :

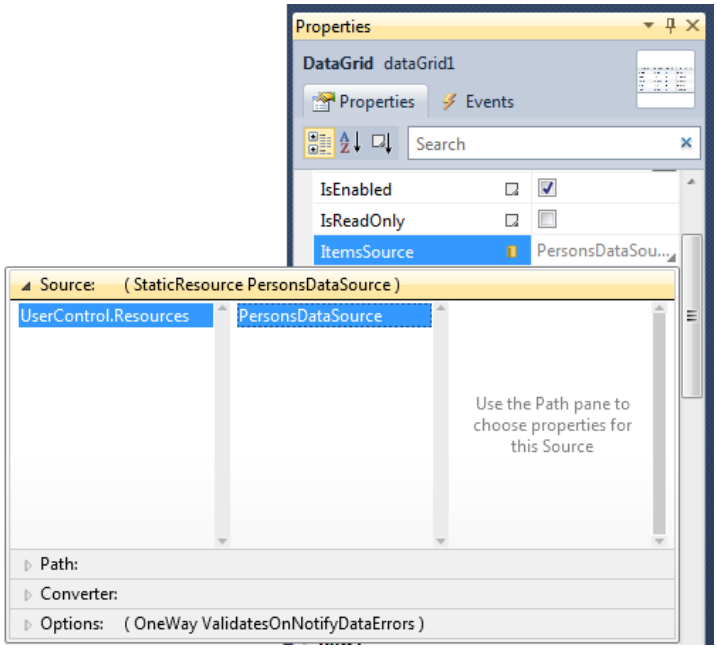
```
ItemsSource = "{Binding Source={StaticResource PersonsDataSource}}"
```

پس از این تعریف، بلافاصله Grid نهایی را در طراحی VS.NET به همراه اطلاعات موجود در آن می‌توان مشاهده کرد. تنظیم خاصیت AutoGenerateColumns به True، سبب ایجاد ستون‌های داده به صورت خودکار شده است. اگر این خاصیت به False تنظیم شود، می‌توان نحوه‌ی ارائه‌ی اطلاعات را به کمک DataTemplate به شکلی سفارشی ارائه داد که این مباحث موضوعات فصل‌های آتی کتاب جاری هستند.

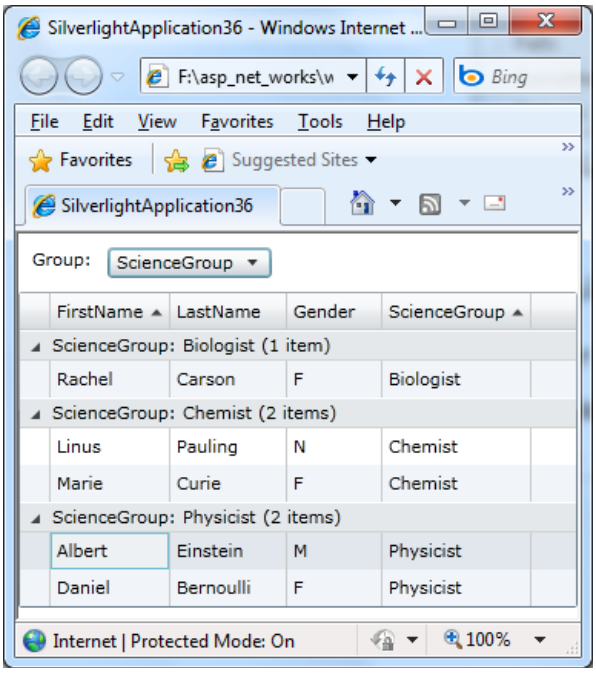
DataGrid برنامه در زمان اجرا به صورت پیش فرض دارای یک سری ویژگی‌های پیشرفته مانند مرتب سازی خودکار و یا امکان ویرایش سلول‌های آن می‌باشد.

در مثال دوم استفاده از توانایی‌های DataGrid قصد داریم اطلاعات کلاس Persons را به صورت گروه بندی شده نمایش دهیم (شکل ۱۲). برای مثال گروه بندی بر اساس گروه علمی افراد تعریف شده و غیره. برای این منظور یک User control جدید را به نام GroupingData به پروژه‌ی جاری اضافه کنید (از منوی پروژه، گزینه‌ی Add new item و انتخاب Silverlight User Control).

کدهای XAML این User control را در ادامه مشاهده خواهید نمود که از یک ComboBox برای انتخاب گروه مورد نظر و یک DataGrid جهت نمایش اطلاعات گروه بندی شده بر اساس گروه انتخابی، تشکیل شده است.



شکل ۱۱– بهبودهای حاصل شده در VS.NET 2010 جهت تعریف ساده‌تر Binding



شکل ۱۲– نمایی از مثال دوم کاربردهای DataGrid جهت گروه بندی داده‌ها

کدهای XAML مرتبط با فایل GroupingData.xaml :

GroupingData.xaml

```
<UserControl
    xmlns:sdk="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
    x:Class="SilverlightApplication36.GroupingData"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
      "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel>
            <StackPanel Orientation="Horizontal" Margin='9'>
                <TextBlock Text="Group:" Margin="0,0,10,0" />
                <ComboBox x:Name="GroupNames"
                     SelectionChanged="GroupNames_SelectionChanged">
                    <ComboBox.Items>
                        <ComboBoxItem Content="ScienceGroup"
                                  IsSelected="True" />
                        <ComboBoxItem Content="Gender" />
                    </ComboBox.Items>
                </ComboBox>
            </StackPanel>
            <sdk:DataGrid  x:Name="personGrid" />
        </StackPanel>
    </Grid>
</UserControl>
```

و کدهای این صفحه جهت گروه بندی اطلاعات به شرح زیر می‌باشند. همانطور که ملاحظه می‌نمائید در متد Load صفحه، یک PagedCollectionView بر اساس اطلاعات شیء Persons ایجاد می‌شود. استفاده از PagedCollectionView جهت مقاصد گروه بندی اطلاعات در یک Grid و یا مرتب سازی یا فیلتر کردن آن‌ها کاربرد دارد.

سپس در سطر بعدی، گروه پیش فرض ابتدایی مساوی ScienceGroup قرار گرفته و در پایان ItemsSource مربوط به DataGrid به این PagedCollectionView مقید (Bind) خواهد شد.

در متد GroupNames_SelectionChanged ، تعاریف گروه‌های موجود حذف شده و سپس گروه جدیدی بر اساس مقدار انتخابی ComboBox اضافه می‌شود.

هر گونه تغییری در یک PagedCollectionView ، به صورت خودکار در UI برنامه نیز منعکس خواهد شد، به همین جهت اینجا دیگر نیازی به فراخوانی مجدد خواص و متدهای Binding نیست.

GroupingData.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using SilverlightApplication36.Model;

namespace SilverlightApplication36
{
    public partial class GroupingData
    {
        public GroupingData()
        {
            InitializeComponent();
            Loaded += MainPage_Loaded;
        }

        PagedCollectionView _pagedView;
        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            // get the underlying source
            _pagedView = new PagedCollectionView(new Persons());

            _pagedView.GroupDescriptions.Add(
                new PropertyGroupDescription("ScienceGroup"));

            personGrid.ItemsSource = _pagedView;
        }

        private void GroupNames_SelectionChanged(
            object sender, SelectionChangedEventArgs e)
        {
            if (_pagedView == null) return;
            // comment this next line out to see
            // adding additional groupings.
            _pagedView.GroupDescriptions.Clear();
            var selectedItem = (ComboBoxItem)GroupNames.SelectedItem;

            _pagedView.GroupDescriptions.Add(
                new PropertyGroupDescription(
                    selectedItem.Content.ToString()));
        }
    }
}
```

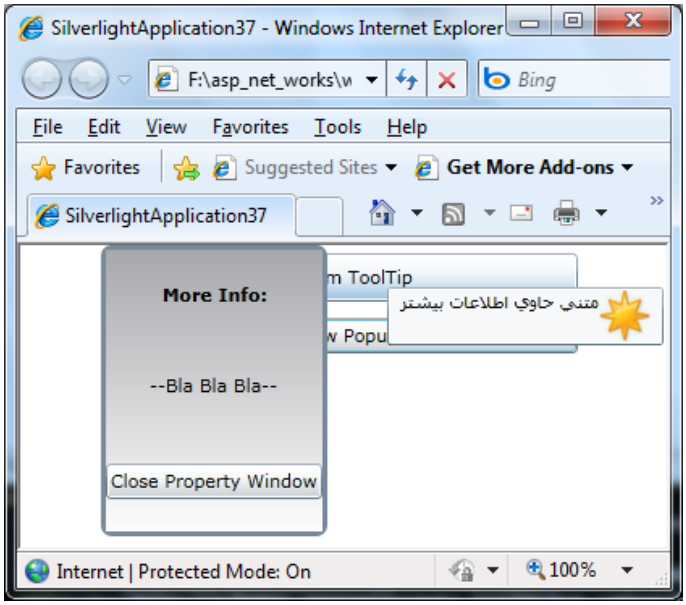
سپس به فایل App.xaml.cs مراجعه نموده و متد Application_Startup آن‌را به شکل زیر تغییر دهید تا User Control جدید، آغاز کننده‌ی برنامه در VS.NET شود:

App.xaml.cs

```
private void Application_Startup(object sender, StartupEventArgs e)
{
    //this.RootVisual = new MainPage();
    this.RootVisual = new GroupingData();
}
```

بهبود ظاهر ToolTip و استفاده از Popups

از ToolTip جهت ارائه‌ی اطلاعات بیشتری در مورد کاربرد یک عنصر در صفحه می‌توان استفاده کرد. خاصیت ToolTip پیش فرض، صرفا یک متن را ارائه می‌دهد اما در Silverlight می‌توان با کمک ToolTipService نسبت به ایجاد ظاهری پیچیده با بهره‌گیری از انواع و اقسام سیستم‌های طرح بندی اقدام نمود. همچنین المان دیگری در Silverlight به نام Popup وجود دارد که بسیار شبیه به ToolTip می‌باشد. مهم‌ترین تفاوت آن با ToolTip امکان کنترل دقیق نمایش و یا بسته شدن آن به همراه امکان تعامل با اجزای درون آن می‌باشد. در طی مثالی قصد داریم با این دو عنصر بیشتر آشنا شویم (شکل ۱۳) .



شکل ۱۳– بهبود ظاهری یک ToolTip و استفاده از Popups

کدهای XAML این مثال را در ذیل ملاحظه می‌نمائید. در این کدها توسط یک Attached property به نام ToolTipService امکان تعریف یک StackPanel و سپس قرار دادن متن و تصویری جهت تعریف یک ToolTip پدید آمده است.

سپس نحوه‌ی تعریف یک Popup را ملاحظه می‌کنید. با کمک سیستم‌های طرح بندی می‌توان انواع و اقسام المان‌های مجاز UI را درون آن تعریف نمود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication37.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" MaxWidth="300" >
        <Grid.RowDefinitions>
            <RowDefinition Height="40" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Button Content="Custom ToolTip" Margin="5" Grid.Row="0">
            <ToolTipService.ToolTip>
                <StackPanel Orientation='Horizontal'
                            FlowDirection="RightToLeft">
           <Image Source='images/burst.png' />
           <TextBlock FontFamily="Tahoma">متني حاوي اطلاعات بيشتر</TextBlock>
                </StackPanel>
            </ToolTipService.ToolTip>
        </Button>
        <Button Content="Show Popup"
                    Margin="5" Grid.Row="1"
                    Click="ShowPopup_Click" />
        <Popup x:Name="propertyPopup"
               MinWidth="300"
               MinHeight="140"
               >
            <Border BorderBrush="LightSlateGray"
                  BorderThickness="3"
                  CornerRadius="5">
                <StackPanel>
                    <StackPanel.Background>
                        <LinearGradientBrush EndPoint="0.5,1"
                                     StartPoint="0.5,0">
                            <GradientStop Color="#FFA8A9AD"
                                Offset="0" />
                            <GradientStop Color="#FFFEFEFE"
```

```
                                Offset="1" />
                        </LinearGradientBrush>
                    </StackPanel.Background>
                    <TextBlock FontWeight="Bold"
                         Margin="0,20"
                     HorizontalAlignment="Center">More Info:</TextBlock>
                <TextBlock  Margin="0,20"
                HorizontalAlignment="Center">--Bla Bla Bla--</TextBlock>

                    <Button Content="Close Property Window"
                      Margin="0,20"
                      Click="closeButton_click"
                      HorizontalAlignment="Center" />
                 </StackPanel>
            </Border>
        </Popup>
    </Grid>
</UserControl>
```

و کدهای متناظر با این صفحه را جهت نمایش و بستن صفحه‌ی Popup ، در ادامه ملاحظه خواهید نمود:

MainPage.xaml.cs

```
using System.Windows;

namespace SilverlightApplication37
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void closeButton_click(object sender,
            RoutedEventArgs e)
        {
            propertyPopup.IsOpen = false;
        }

        private void ShowPopup_Click(object sender,
            RoutedEventArgs e)
        {
            propertyPopup.IsOpen = true;
        }
    }
}
```