Silverlight 4

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

فهرست مطالب

فصل ۴ – آشنایی با طرح بندی رابط کاربر در Silverlight

مقدمه

یک موتور طراحی رابط کاربر خوب باید تسهیلاتی را جهت قرار دادن اشیاء مختلف بر روی صفحه و تنظیم محل قرارگیری آن‌ها ارائه دهد. به این امکانات، طرح بندی و Layout گفته می‌شود. کار آن نیز کنترل اندازه و مکان نسبی عناصر بصری قرار گرفته بر روی صفحه‌ی جاری برنامه است.
یک موتور طرح بندی رابط کاربر خوب باید دارای شرایط ذیل باشد:

- باید بتواند خود را با اندازه‌های متفاوت مرورگر کاربر هماهنگ نماید.
- باید در برابر تغییرات اندازه‌ی قلم برنامه مقاوم باشد.
- مکان‌های نسبی بین اشیاء را در هر حالت و اندازه‌ای بخاطر سپرده و رعایت کند.

همانطور که در فصل قبل نیز ذکر شد، فایل‌های XAML از اصول کاری فایل‌های XML پیروی می‌کنند؛ بنابراین تنها دارای یک عنصر ریشه می‌توانند باشند که در Silverlight عموما User controls آن‌را تشکیل می‌دهند. همچنین هر User control نیز تنها می‌تواند دارای یک عضو فرزند باشد. بنابراین بهترین انتخاب در اینجا اعضایی هستند که می‌توانند چندین عضو را درون خود جای دهند:

- Panels : دربرگیرنده‌های سیستم طرح بندی Silverlight بوده و به خودی خود ظاهری را ارائه نمی‌دهند.
- ItemControls : امکان نمایش چندین عضو مقید شده (Binding) به آن‌ها وجود دارد. این اشیاء در حقیقت نوعی کنترل بوده که می‌توان ظاهر آن‌ها را با کمک قالب‌ها کاملا دگرگون ساخت.

کلاس System.Windows.Controls.Panel ، کلاس پایه‌ی تمامی دربرگیرنده‌های سیستم طرح بندی Silverlight است. Silverlight به صورت پیش فرض دارای تعدادی Panel استاندارد می‌باشد مانند:

- StackPanel
- Grid
- Canvas

همچنین Silverlight toolkit نیز تعدادی Panel را به این مجموعه افزوده است که شامل موارد ذیل هستند:

- WrapPanel

- DockPanel

لازم به ذکر است که امکان ایجاد Panels سفارشی نیز وجود داشته و محدودیتی از این لحاظ وجود ندارد. جزئیات این Panels در ادامه‌ی فصل بررسی خواهند شد.

تنظیم رنگ زمینه و حاشیه‌ی یک Panel

به زمینه‌ی یک Panel می‌توان انواع و اقسام Brush های تعریف شده در Silverlight را مانند قلم‌های رنگی، گرادیان‌های خطی، حلقوی و تصاویر، انتساب داد. لطفا به مثال بعد دقت بفرمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication9.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">
    <!-- Panels do not have borders, use the Border decorator instead-->
    <Border BorderBrush='Orange'>
        <Grid x:Name="LayoutRoot"
          Background="Azure">
          <!-- borders work around other elements too -->
          <Border Margin='50'
            BorderBrush='DarkOrange'
            BorderThickness='2,10'
            CornerRadius='10'>
              <TextBlock Text='Border:  Properties, BorderThickness, BorderBrush,
CornerRadius' />
          </Border>
        </Grid>
    </Border>
</UserControl>
```

در حین ایجاد یک User control جدید در Silverlight ، به صورت پیش فرض دربرگیرنده‌ی طرح‌بندی از نوع Grid جهت تعریف دربرگیرنده‌ی سایر اشیاء صفحه اضافه می‌شود. کلیه‌ی Panels در Silverlight دارای حاشیه (Border) نیستند. برای این منظور مطابق مثال فوق می‌توان از شیء Border استفاده نمود. همچنین از Borders جهت مزین سازی سایر المان‌های صفحه نیز می‌توان کمک گرفت که نمونه‌ای از آن جهت مزین سازی

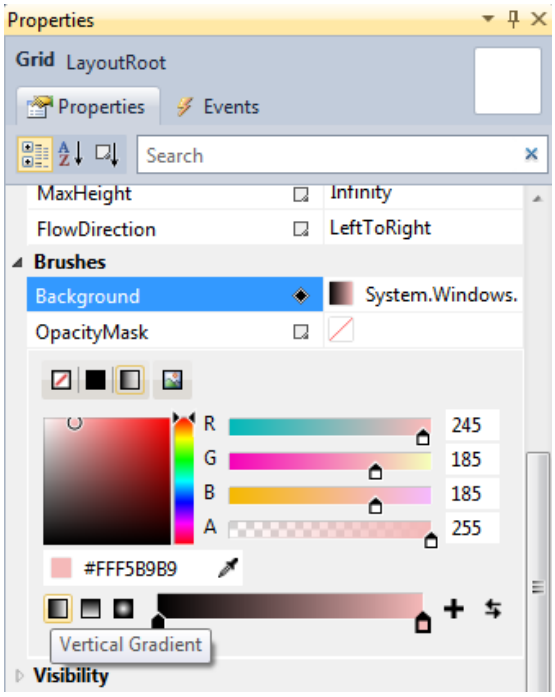
یک TextBlock ساده در اینجا قابل مشاهده است. ویژگی Margin، فاصله‌ی بین دو شیء را تنظیم کرده و همچنین میزان ضخامت و شعاع دایره‌ای که گوشه‌های این حاشیه را تشکیل می‌دهند نیز تعیین گردیده است. نحوه‌ی تنظیم رنگ زمینه‌ی Grid با کمک ویژگی Background هم در این مثال ذکر گردیده است. Brush مورد استفاده در اینجا از نوع SolidColorBrush است.

اگر علاقمند باشید که از انواع دیگر Brush های تعریف شده نیز استفاده نمائید، با بهبودهای حاصل شده در طراح XAML موجود در VS.NET 2010، این امر بسیار ساده شده است (شکل ۱). در اینجا تنها کافی است به برگه‌ی خواص Grid مراجعه کرده و بر روی دکمه‌ی مقابل گزینه‌ی Background کلیک کرد. پس از آن قسمت طراح انواع Brush های مهیا در دسترس خواهد بود. برای مثال Brush از نوع Gradient Brush را انتخاب نموده و سپس ترکیب رنگ دلخواه خود را اعمال نمائید. به این صورت بلافاصله کدهای XAML مربوطه نیز به پروژه اضافه خواهند شد:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication10.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" >
        <Grid.Background>
            <LinearGradientBrush EndPoint="1,0.5" StartPoint="0,0.5">
                <GradientStop Color="Black" Offset="0" />
                <GradientStop Color="#FFF5B9B9" Offset="1" />
            </LinearGradientBrush>
        </Grid.Background>
    </Grid>
</UserControl>
```

کدهای فوق که بیانگر پس زمینه‌ای از نوع گرادیان خطی می‌باشند توسط طراح VS.NET به صورت خودکار تولید شده‌اند و همانطور که ملاحظه می‌کنید از روش property-element syntax جهت معرفی و انتساب یک شیء پیچیده به ویژگی پس‌زمینه Grid استفاده نموده است.



شکل ۱– طراح VS.NET و امکان استفاده از انواع Brush های تعریف شده

معرفی StackPanel

StackPanel یکی از ساده‌ترین Panel های Silverlight و WPF است و کار آن نمایش اشیاء قرار گرفته در آن به صورت یک پشته می‌باشد. این پشته می‌تواند جهت عمودی یا افقی داشته باشد. لطفا به مثال‌های ذیل دقت بفرمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication11.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="800">
    <Grid x:Name="LayoutRoot" Margin='50'>
        <Border BorderBrush='DarkGoldenrod'
            BorderThickness='3' >
            <StackPanel Orientation="Horizontal">
                <TextBlock Text='StackPanel Sample' />
                <Button Content='First' />
                <Button Content='Second' />
```

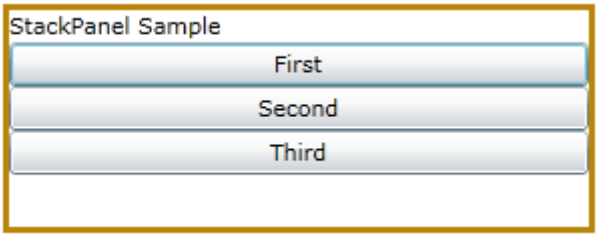
```
                <Button Content='Third' />
            </StackPanel>
        </Border>
    </Grid>
</UserControl>
```

شکل ۲– نمایش اعضای قرار گرفته در یک StackPanel در جهت افقی

در این مثال یک StackPanel به همراه یک برچسب و سه دکمه داخل آن معرفی شده‌اند. جهت StackPanel به شکل صریح به حالت افقی تنظیم شده است. در صورت عدم ذکر آن، جهت پیش فرض چیدمان عناصر در یک StackPanel به صورت عمودی می‌باشد (همانند مثال ذیل) :

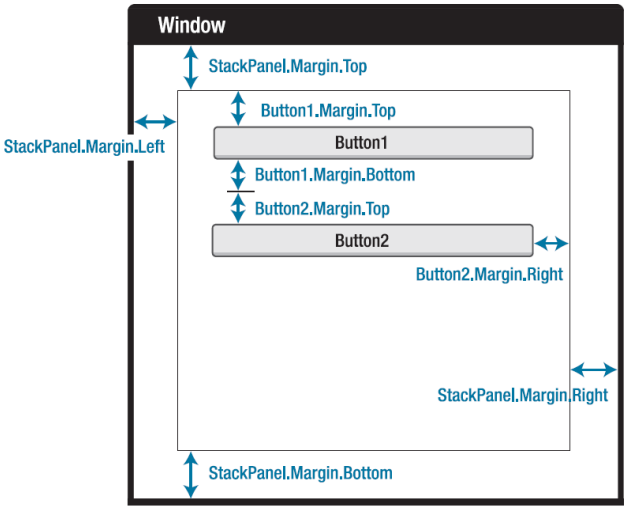
MainPage.xaml

```
<UserControl x:Class="SilverlightApplication11.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="800">
    <Grid x:Name="LayoutRoot" Margin='50'>
        <Border BorderBrush='DarkGoldenrod'
            BorderThickness='3' >
            <StackPanel>
                <TextBlock Text='StackPanel Sample' />
                <Button Content='First' />
                <Button Content='Second' />
                <Button Content='Third' />
            </StackPanel>
        </Border>
    </Grid>
</UserControl>
```



شکل ۳- نمایش اعضای قرار گرفته در یک StackPanel در جهت پیش فرض عمودی

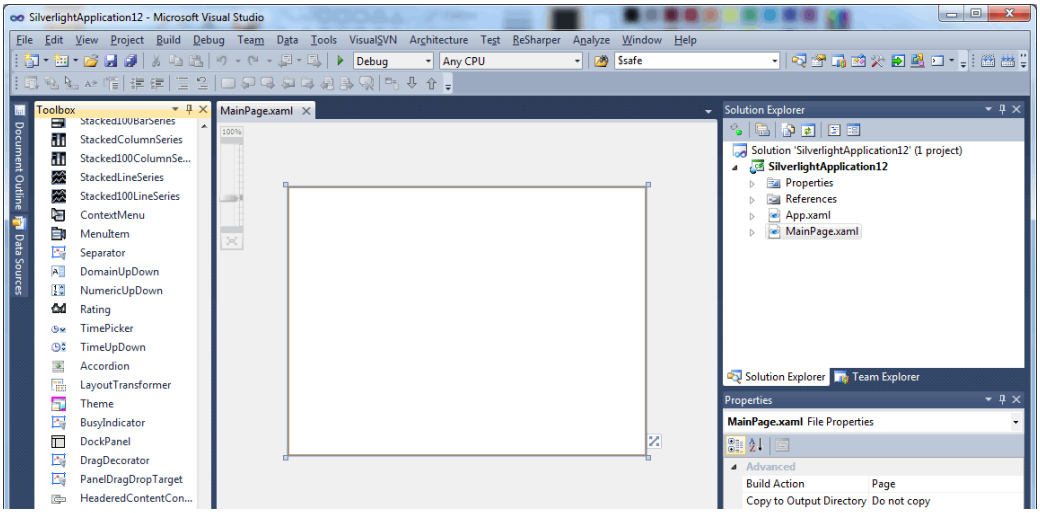
اگر از کنار هم قرار گرفتن عناصر در این حالت ناراضی هستید، می‌توان با کمک خاصیت Margin ، فاصله‌ی بین عناصر را تعیین نمود (شکل زیر).



شکل ۴– نمایی از اثرات تعریف Margins در یک StackPanel

معرفی DockPanel

بهترین مثال برای معرفی DockPanel که هر یک از اجزای معرفی شده در آن به قسمتی از صفحه لنگر خواهند انداخت، سیستم طرح بندی VS.NET است (شکل ۵). برای مثال در VS.NET ، جعبه ابزار کنترل‌ها به سمت چپ صفحه، صفحه‌ی خواص عناصر مختلف به سمت راست و نوار ابزار به بالای صفحه لنگر انداخته‌اند (Docking) .



شکل ۵– سیستم طرح بندی اجزای مختلف VS.NET

DockPanel و WrapPanel جزو مجموعه‌ی Silverlight toolkit می‌باشند که از آدرس ذیل قابل دریافت هستند:

http://silverlight.codeplex.com/

پس از دریافت و نصب آن، یک پروژه‌ی جدید Silverlight را آغاز کرده و سپس DockPanel را از جعبه ابزار کنترل‌ها بر روی صفحه‌ی فرم، کشیده و رها کنید. در این حالت سه عملیات به صورت همزمان و خودکار رخ خواهند داد:

۱. اضافه شدن ارجاعات لازم به اسمبلی‌های Silverlight toolkit در قسمت References در Solution explorer.
۲. اضافه شدن فضای نام Silverlight toolkit به کدهای XAML صفحه‌ی جاری
۳. اضافه شدن تگ ابتدایی مرتبط با DockPanel به صفحه

لطفا به مثالی در این زمینه دقت بفرمائید:

MainPage.xaml

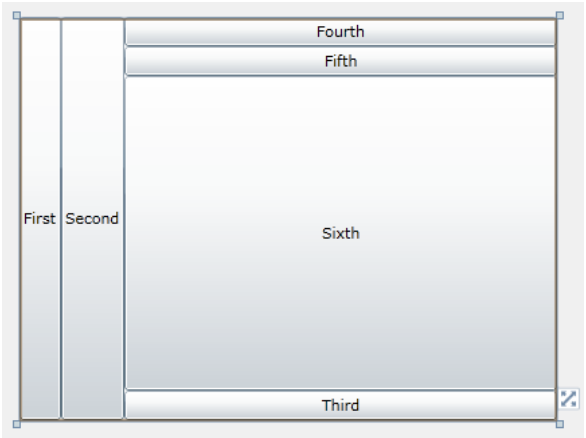
```
<UserControl x:Class="SilverlightApplication12.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
      "http://schemas.openxmlformats.org/markup-compatibility/2006"
    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:DockPanel LastChildFill='True'>
            <Button Content='First' toolkit:DockPanel.Dock='Left' />
            <Button Content='Second' toolkit:DockPanel.Dock='Left' />
            <Button Content='Third' toolkit:DockPanel.Dock='Bottom' />
            <Button Content='Fourth' toolkit:DockPanel.Dock='Top' />
            <Button Content='Fifth' toolkit:DockPanel.Dock='Top' />
            <Button Content='Sixth' toolkit:DockPanel.Dock='Right' />
        </toolkit:DockPanel>
    </Grid>
</UserControl>
```

در مثال فوق یک DockPanel به صفحه اضافه شده و سپس شش دکمه به قسمت‌های چپ، پایین، بالا و سمت راست صفحه لنگر انداخته‌اند (شکل ۶). محل قرارگیری دکمه‌ها توسط Attached properties کلاس DockPanel معرفی شده‌اند (که در انتهای فصل قبل در مورد آن‌ها توضیح داده شد).

در این مثال اگر ویژگی LastChildFill به False تنظیم شود، آخرین دکمه (ششمین دکمه) همانند شکل ۶، باقیمانده‌ی فضای خالی را پر نکرده و به شکل قرینه‌ی دکمه‌ی یک قرار خواهد گرفت.

همچنین باید دقت داشت که عناصر معرفی شده در DockPanel دارای حق تقدم می‌باشند (از شماره یک به بعد؛ تقدم بیشتر به کمتر)؛ به همین جهت دکمه‌ی سوم مثال معرفی شده تمام ناحیه‌ی پایین صفحه را پوشش نمی‌دهد (زیرا دکمه‌های یک و دو حق تقدم بیشتری دارند). اگر علاقمند باشید که دکمه‌ی سوم تمام ناحیه‌ی پایین صفحه را پوشش دهد، آن‌را به عنوان اولین دکمه‌ی معرفی شده در DockPanel تعریف کنید.



شکل ۶– مثالی از نحوه‌ی طرح بندی با DockPanel .

معرفی WrapPanel

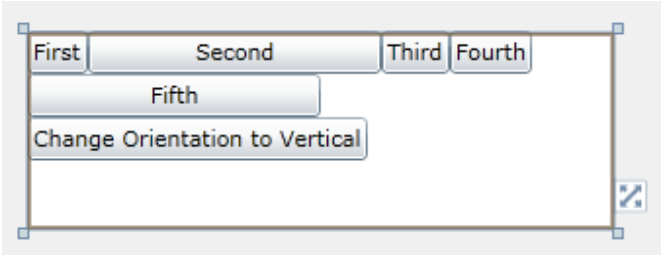
عملکرد WrapPanel مجموعه‌ی Silverlight toolkit همانند StackPanel است با این تفاوت که طول و عرض جاری صفحه‌ی نمایش را جهت معرفی عناصر خود محاسبه کرده و اگر برای نمایش عناصر خود در ردیف یا ستون جاری با کمبود جا مواجه گردد، عناصر باقیمانده را به سطر یا ستون بعدی منتقل می‌کند (سطر یا ستون بر اساس جهت عمودی یا افقی معرفی شده تنظیم می‌شوند).

بهترین روش برای قرار دادن یک WrapPanel در صفحه‌ی جاری، کشیدن و رها کردن آن از جعبه ابزار کنترل‌های VS.NET است تا به این صورت افزودن ارجاعات لازم به اسمبلی‌های آن و همچنین فضای نام مرتبط، به صورت خودکار صورت گیرد.

لطفا به مثالی در این مورد دقت بفرمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication13.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="300"
    xmlns:toolkit=
   "http://schemas.microsoft.com/winfx/2006/xaml/presentation/toolkit">
    <Grid x:Name="LayoutRoot" Background="White">
        <toolkit:WrapPanel  Orientation='Horizontal'>
            <Button Content='First' />
            <Button Content='Second' Width='150'/>
            <Button Content='Third' />
            <Button Content='Fourth' />
            <Button Content='Fifth' Width='150'/>
            <Button Content='Change Orientation to Vertical' />
        </toolkit:WrapPanel>
    </Grid>
</UserControl>
```



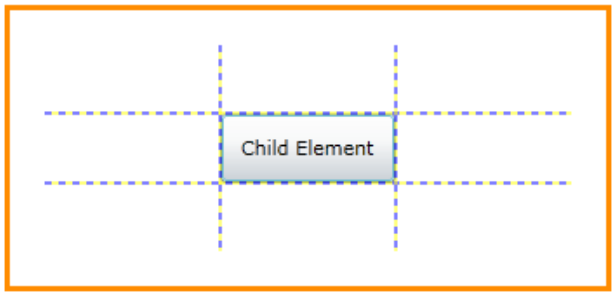
شکل ۷– نمایی از نحوه‌ی چیدمان در یک WrapPanel

همانطور که ملاحظه می‌فرمائید در این مثال، ۶ دکمه داخل یک WrapPanel تعریف شده‌اند که تعداد دکمه‌های هر سطر بر اساس طول صفحه‌ی جاری محاسبه شده و باقیمانده در ردیف‌های دیگر نمایش داده شده‌اند. برای تمرین، جهت این WrapPanel را به Vertical تنظیم کرده و عرض صفحه را در حین نمایش کم و زیاد نمائید (کوچکتر و یا بزرگتر کردن عرض مرورگر) تا با نحوه‌ی چیدمان در این حالت نیز آشنا گردید.

معرفی Grid

پرکاربردترین و قویترین سیستم طرحبندی در Silverlight ، Grid است که در حین افزودن یک User control جدید به صفحه به صورت خودکار به عنوان Panel اصلی در برگیرنده‌ی اشیاء معرفی می‌شود. روش کارکردن با آن هم بر اساس تعریف سطرها و ستون‌های یک Grid و سپس انتساب اشیاء مختلف به هر یک از سلول‌های حاصل می‌باشد.

در ادامه قصد داریم یک Grid جدید را تعریف نموده و دکمه‌ای را در سطر و ستون دوم آن قرار دهیم (شکل ۸). برای این منظور لطفا به مثال بعد دقت بفرمائید:



شکل ۸– قرار دادن یک دکمه در سطر و ستون دوم یک Grid

MainPage.xaml

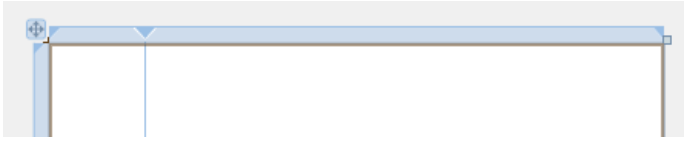
```
<UserControl x:Class="SilverlightApplication14.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
     "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <Border BorderBrush='DarkOrange' Margin="50"
            BorderThickness='3'>
            <Grid Margin='20' ShowGridLines="True">
                <Grid.RowDefinitions>
                    <RowDefinition />
                    <RowDefinition />
                    <RowDefinition />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <!-- Rows and Columns numbers start at zero -->
                <Button  Content='Child Element' Grid.Row="1"
                       Grid.Column="1" />
            </Grid>
        </Border>
    </Grid>
</UserControl>
```

در این مثال ابتدا یک حاشیه‌ی جدید با فاصله‌ی ۵۰ از لبه‌ی صفحه تعریف شده است و سپس درون آن یک Grid جدید با سه سطر و سه ستون تعریف گردیده است. همانطور که ملاحظه می‌نمائید نحوه‌ی تعریف سطرها و ستون‌ها با کمک Property-element syntax است. در یک گرید، سطر و ستون‌ها از شماره‌ی صفر شروع می‌شوند. بنابراین جهت معرفی یک دکمه در سطر و ستون دوم باید به شکل زیر عمل نمود:

```
<Button  Content='Child Element' Grid.Row="1" Grid.Column="1" />
```

در اینجا با کمک سیستم Attached properties کلاس Grid ، سطرها و ستون‌های دربرگیرنده‌ی دکمه‌ی این مثال تعریف شده‌اند. برای اینکه این نحوه‌ی قرارگیری بهتر نمایش داده شود، ویژگی ShowGridLines مربوط به Grid به True تنظیم گشته است.

روش ساده‌تر ایجاد سطرها و ستون‌ها در طراح VS.NET (بدون کد نویسی دستی آن‌ها) به این صورت است که ابتدا داخل صفحه‌ی طراح یکبار کلیک نمائید تا Grid جاری انتخاب و فعال شود؛ سپس اشاره‌گر Mouse خود را به حاشیه‌ی آبی رنگی که ظاهر شده است نزدیک نمائید تا راهنماهای ایجاد سطر و ستون‌ها ظاهر شوند. اکنون با هر کلیک Mouse ، یک سطر یا ستون جدید به Grid انتخابی اضافه خواهند شد (شکل ۹).



شکل ۹– استفاده از طراح VS.NET جهت ایجاد یک ستون جدید.

ترکیب شیوه‌های طرح بندی مختلف با یکدیگر

یکی از روش‌های متداول کار با Grid در Silverlight ، ترکیب این شیوه‌ی طرح بندی با سایر شیوه‌های موجود است. برای مثال در ادامه قصد داریم یک Grid را با همراه سه ستون و یک سطر، ایجاد نموده و سپس چهار دکمه را توسط یک StackPanel در ستون سوم آن نمایش دهیم (شکل ۱۰). لطفا به کدهای XAML مثال بعد دقت بفرمائید:

شکل ۱۰– تعریف یک StackPanel داخل یک Grid.

MainPage.xaml

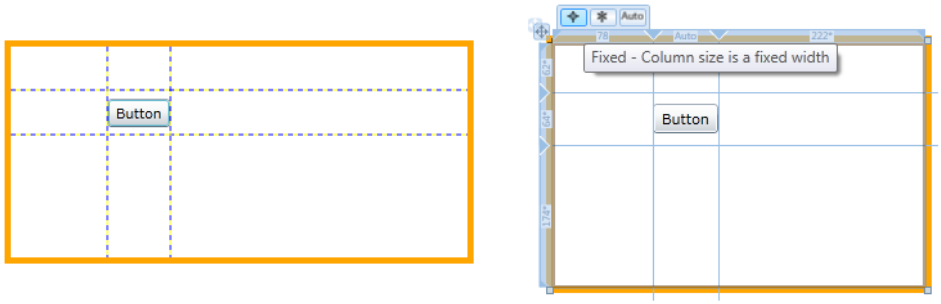
```
<UserControl x:Class="SilverlightApplication15.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
     "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
```

```
        <Border BorderBrush='DarkOrange' Margin="50"
                                        BorderThickness='3'>
            <Grid Margin='20' ShowGridLines='True'>

                <!--  a one row, three column grid-->
                <Grid.RowDefinitions>
                    <RowDefinition />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <StackPanel Grid.Column='2'>
                    <Button  Content='Child Element' />
                    <Button  Content='Child Element' />
                    <Button  Content='Child Element' />
                    <Button  Content='Child Element' />
                </StackPanel>
            </Grid>
        </Border>
    </Grid>
</UserControl>
```

در این مثال StackPanel تعریف شده توسط ویژگی Attached properties به ستون سوم Grid الحاق شده است. اگر Attached properties مورد نیاز جهت تعیین محل قرارگیری یک عنصر در سطرها و ستون‌های مورد نظر ذکر نشوند، آن عنصر در سطر و ستون اول (اولین سلول یک Grid) قرار خواهد گرفت.

آشنایی با روش‌های اندازه گذاری سطرها و ستون‌های یک Grid

فرض کنید می‌خواهیم یک Grid دارای سه ستون مشخص باشد. اندازه‌ی ستون اول آن ثابت بوده، ستون دوم به صورت خودکار به اندازه‌ی محتوای خود تغییر اندازه دهد و ستون سوم مابقی فضای باقیمانده را استفاده نماید (شکل ۱۱). برای پیاده سازی این مثال لطفا به کدهای XAML بعد دقت بفرمائید :



شکل ۱۱– روش‌های متفاوت اندازه گذاری سطرها و ستون‌های یک Grid

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication16.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">
    <Border BorderThickness="5" BorderBrush="Orange" Margin="50">
      <Grid x:Name="LayoutRoot"  ShowGridLines="True">
        <Grid.RowDefinitions>
            <RowDefinition Height="62*" />
            <RowDefinition Height="64*" />
            <RowDefinition Height="174*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="78" />
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Button Content="Button" Grid.Row="1" Grid.Column="1"
                Height="23" Width="51" />
      </Grid>
    </Border>
</UserControl>
```

در هنگام معرفی ستون‌های این Grid از سه روش اندازه گذاری متفاوت استفاده شده است:

- اندازه‌ی ثابت :

```
<ColumnDefinition Width="78" />
```

- اندازه‌ی خودکار که بر اساس اندازه‌ی محتویات آن تنظیم خواهد شد:

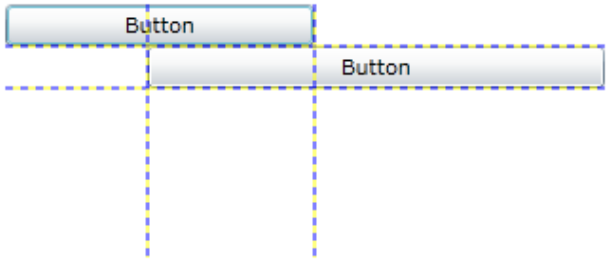
```
<ColumnDefinition Width="Auto" />
```

- اندازه‌ی نسبی که با ستاره مشخص شده و باقیمانده‌ی فضای خالی را به خود اختصاص خواهد داد:

```
<ColumnDefinition Width="*" />
```

قرار دادن اشیاء بر روی دو یا چند سلول Grid و پوشاندن آن‌ها (Spanning)

تا اینجا با نحوه‌ی تعریف محل قرارگیری اشیاء در سلول‌های مختلف یک Grid آشنا شدیم. گاهی از اوقات در عمل نیاز خواهد بود تا اشیاء تعریف شده، چندین سلول را پوشش دهند (شکل بعد). برای این منظور از ویژگی RowSpan استفاده می‌شود. لطفا به مثال بعد دقت نمائید:



شکل ۱۲– پوشاندن دو سلول مجاور در یک Grid .

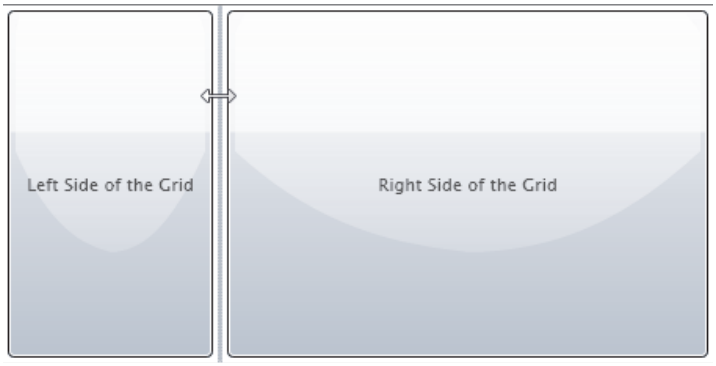
MainPage.xaml

```
<UserControl x:Class="SilverlightApplication22.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" Margin="20"
             ShowGridLines="True" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="95*" />
            <ColumnDefinition Width="111*" />
            <ColumnDefinition Width="194*" />
        </Grid.ColumnDefinitions>
        <Button Content="Button" Grid.ColumnSpan="2"  />
        <Button Content="Button"
                Grid.Column="1"
                Grid.Row="1"
                Grid.ColumnSpan="2"
                Grid.RowSpan="1" />
    </Grid>
</UserControl>
```

در این مثال ساده، یک Grid با سه سطر و سه ستون تعریف شده است. با کمک Attached properties از نوع ColumnSpan و RowSpan کلاس Grid، کار پوشاندن سلول‌های مجاور صورت گرفته است.

تغییر اندازه‌ی طول و عرض سلول‌های یک Grid به کمک GridSplitter

ممکن است در برنامه‌ی خود بخواهید به کاربران این امکان را بدهید که بتوانند طول و عرض سلول‌ها و یا سطر و ستون‌های یک Grid را تغییر بدهند. برای مثال در VS.NET می‌توان به سادگی عرض و یا طول Solution explorer و موارد مشابه را به کمک Mouse تغییر داد (شکل زیر).



شکل ۱۳– نمایی از یک GridSplitter .

کنترل GridSplitter در اسمبلی System.Windows.Controls قرار دارد. بنابراین یا باید به صورت دستی ارجاعی را به آن افزود و سپس فضای نام مرتبط را نیز ضمیمه نمود و یا به سادگی می‌توان این کنترل را از جعبه ابزار VS.NET بر روی فرم کشیده و رها کرد. به این صورت کار افزودن ارجاع ذکر شده و همچنین افزودن فضای نام sdk به XAML جاری برنامه به صورت خودکار انجام خواهد شد. لطفا به مثال بعد در این زمینه دقت بفرمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication17.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"
    xmlns:sdk=
     "http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">
    <Grid x:Name="LayoutRoot" Background="White">
        <Border BorderBrush='DarkOrange' Margin="20"
                                          BorderThickness='3'>
            <Grid Margin='40'>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width='120' />
```

```
                    <ColumnDefinition  Width='Auto' />
                <ColumnDefinition Width='120' />
                <ColumnDefinition  Width='Auto' />
                <ColumnDefinition Width='120' />
            </Grid.ColumnDefinitions>

            <sdk:GridSplitter Grid.Column='1'
                   VerticalAlignment='Stretch'
                   HorizontalAlignment='Center'
                   Width='6' />

          <sdk:GridSplitter  Grid.Column='3'
                 VerticalAlignment='Stretch'
                 HorizontalAlignment='Center'
                 Width='6'
                 ShowsPreview='True' />

            <StackPanel Background='Orange'
                 Grid.Column='0' HorizontalAlignment='Stretch'>
            </StackPanel>
            <StackPanel Background='Red' Grid.Column='2' />
            <StackPanel Background='Orange' Grid.Column='4' />
        </Grid>
    </Border>
</Grid>
</UserControl>
```

در این مثال یک Grid به همراه ۵ ستون ایجاد شده است. از ستون‌های اختصاصی دوم و چهارم جهت معرفی محل قرارگیری کنترل‌های GridSplitter استفاده خواهیم کرد (هر چند الزامی نیست اما ظاهر بهتری حاصل خواهد شد). به همین جهت عرض آن‌ها نیز به Auto تنظیم شده است. سپس با کمک Attached properties ، دو GridSplitter معرفی شده به ستون‌های دوم و چهارم الحاق شده‌اند.

در GridSplitter دوم، ویژگی ShowsPreview به True تنظیم شده است. در این حالت سایه‌ای خاکستری از GridSplitter در حین کشیدن و رها کردن آن در صفحه مشاهده خواهد شد. در غیراینصورت به محض تغییر مکان GridSplitter ، اندازه‌ی سلول‌ها نیز تغییر خواهد نمود.

معرفی Canvas

روش قرارگیری المان‌ها در سیستم طرح بندی Canvas بر اساس مختصات مطلق آن‌ها است. روش قدیمی ساخت رابط کاربر در WinForms نیز بر همین مبنا است. مناسبت‌ترین کاربرد این روش طرح بندی غیرمنعطف در Silverlight ، رسم اشیاء هندسی پیچیده است و به هیچ عنوان از آن برای طراحی رابط کاربر استفاده نکنید.

در مثال بعد صرفا جهت نمایش نحوه‌ی کاربرد این روش طرح بندی، چند دکمه را بر روی صفحه‌ی جاری قرار داده‌ایم:

MainPage.xaml

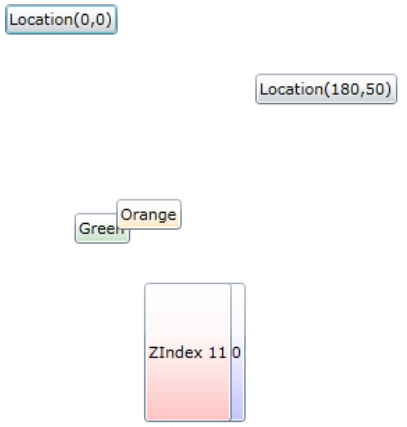
```
<UserControl x:Class="SilverlightApplication18.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
     "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">
    <Canvas x:Name="LayoutRoot" Background="White">
        <Button Content='Location(0,0)' />

        <Button Content='Location(180,50)'
            Canvas.Left='180'
            Canvas.Top='50' />

        <Button Content='Green'
            Canvas.Left='50'
            Background='Green'
            Canvas.Top='150' />
        <Button Content='Orange'
            Background='Orange'
            Canvas.Left='80'
            Canvas.Top='140' />
        <Button x:Name='redButton'
            Content='ZIndex 11'
            Canvas.Left='100'
            Canvas.Top='200'
            Canvas.ZIndex='11'
            Height='100'
            Background='Red' />
        <Button  x:Name='blueButton'
             Content='ZIndex 10'
             Canvas.Left='110'
             Canvas.Top='200'
             Canvas.ZIndex='10'
             Height='100'
             Background='Blue'
             Click='Button_Click' />
    </Canvas>
</UserControl>
```

و کدهای مرتبط با این User control به شرح بعد هستند:

MainPage.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
namespace SilverlightApplication18
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            Canvas.SetZIndex(blueButton, 90);
        }
    }
}
```

حاصل نهایی استفاده از Canvas در این مثال را در شکل زیر می‌توان مشاهده نمود :

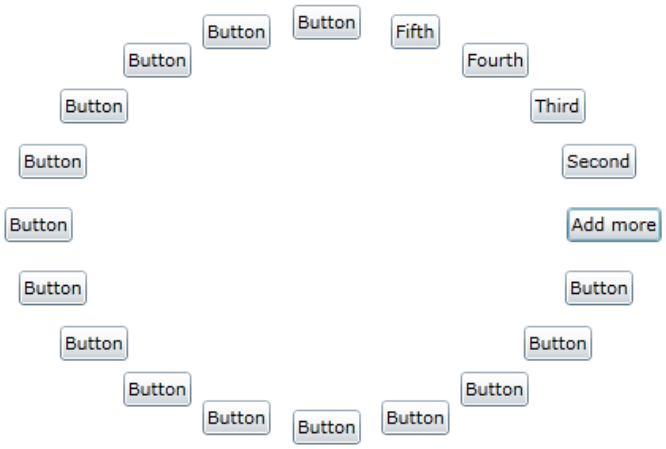


شکل ۱۴– استفاده از Canvas جهت تعیین مختصات یک سری دکمه در برنامه

هنگام استفاده از سیستم طرح بندی Canvas ، اگر مختصات محل قرارگیری شیء مورد نظر ذکر نگردد، از مختصات صفر و صفر استفاده خواهد شد. در سایر حالات باید به کمک Attached properties ، مختصات سمت چپ و بالای هر شیء را مشخص نمود. نکته‌ی دیگری که در این مثال به آن پرداخته شده است، مبحث ZIndex می‌باشد. برای مثال، قسمتی از دو دکمه‌ی پرتغالی و سبز، بر روی یکدیگر قرار گرفته‌اند. شیءایی که ZIndex بیشتری دارد، بر روی شیء دیگر قرار خواهد گرفت. توسط متد Button_Click نیز نحوه‌ی مقدار دهی این خاصیت در کدهای برنامه ذکر گردیده است.

ایجاد یک Panel سفارشی

در Silverlight محدود به Panels پیش فرض موجود نیستیم و می‌توان نسبت به طراحی Panels سفارشی نیز اقدام نمود. در ادامه قصد داریم یک Panel حلقوی را ایجاد کنیم به طوریکه اعضای قرار گرفته در آن حول محیط یک دایره چیده شوند (شکل ۱۵).



شکل ۱۵– نمایی از کاربرد پنل سفارشی حلقوی ایجاد شده

برای ایجاد یک Panel سفارشی باید کلاس Panel را همانند کدهای کلاس CircularPanel بعد بسط داد و سپس دو متد MeasureOverride و ArrangeOverride آن‌را تحریف (override) نمود:

CircularPanel.cs

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

namespace SilverlightApplication19
{
    public class CircularPanel : Panel
    {
        protected override Size MeasureOverride(Size availableSize)
        {
            // called by parent element
            // our opportunity to measure our children
            foreach (UIElement elem in Children)
            {
                elem.Measure(new Size(
                    double.PositiveInfinity,
```

```
                    double.PositiveInfinity));
            }
            return base.MeasureOverride(availableSize);
        }

        protected override Size ArrangeOverride(Size finalSize)
        {
            // called by our parent element
            // our opportunity to arrange our children
            if (Children.Count == 0)
                return finalSize;

            double angle = 0;

            //Degrees converted to Radian by multiplying with PI/180
            double angularCounter =
                 (360.0 / Children.Count) * (Math.PI / 180);

            // hacky way to find radiuses (radii?)
            double radiusX = finalSize.Width / 2.4;
            double radiusY = finalSize.Height / 2.4;

            foreach (UIElement elem in Children)
            {
                // position each child
                var childPoint = new Point(
                    Math.Cos(angle) * radiusX,
                    -Math.Sin(angle) * radiusY);
                var actualChildPoint = new Point(
                   finalSize.Width / 2 + childPoint.X - elem.DesiredSize.Width / 2,
                   finalSize.Height / 2 + childPoint.Y - elem.DesiredSize.Height / 2);

                elem.Arrange(new Rect(
                    actualChildPoint.X,
                    actualChildPoint.Y,
                    elem.DesiredSize.Width,
                    elem.DesiredSize.Height));

                angle += angularCounter;
            }

            return finalSize;
        }
    }
}
```

توسط متد MeasureOverride ، تمامی اجزای یک Panel اندازه‌گیری می‌شوند. سپس توسط متد ArrangeOverride این امکان را خواهیم یافت که چیدمان دلخواه خود را به این عناصر اندازه گیری شده اعمال نمائیم.

در ادامه نحوه‌ی استفاده از این کلاس در صفحه‌ی اصلی برنامه پس از افزودن فضای نام مرتبط به شکل زیر می‌باشد:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication19.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:local="clr-namespace:SilverlightApplication19"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <local:CircularPanel x:Name='circPanel'>
            <Button Content='Add more' Click='Button_Click'/>
            <Button Content='Second' />
            <Button Content='Third' />
            <Button Content='Fourth' />
            <Button Content='Fifth' />
        </local:CircularPanel>
    </Grid>
</UserControl>
```

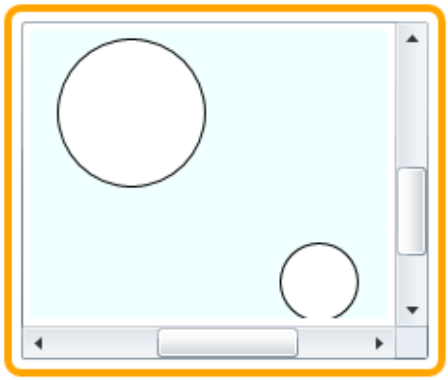
و کدهای متناظر با این صفحه در ادامه ذکر شده‌اند:

MainPage.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
namespace SilverlightApplication19
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
        private void Button_Click(object sender, RoutedEventArgs e)
        {
          circPanel.Children.Add(new Button { Content = "Button" });
        }
    }
}
```

استفاده از ScrollViewer

هیچکدام از سیستم‌های طرح بندی موجود امکان ارائه عناصر داخل خود را به صورت یک طومار ندارند. برای مثال سیستم طرح بندی Canvas را در نظر بگیرید که توسط آن دو دایره بر روی صفحه‌ی جاری ترسیم شده‌اند. اگر اندازه‌ی طول و عرض مرورگر را کاهش دهیم (شکل ۱۶)، پس از اندکی تغییر دیگر امکان مشاهده‌ی هر دو دایره یا یکی از آن‌ها را نخواهیم داشت (بسته به طول و عرض جدید انتخابی). برای رفع این مشکل، شیء ScrollViewer در Silverlight ارائه شده است که مثالی از کاربرد آن‌را در ادامه مرور خواهیم کرد.



شکل ۱۶– استفاده از ScrollViewer به همراه سیستم طرح بندی Canvas

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication20.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">
    <Border Margin="30" BorderBrush="Orange"
            CornerRadius="7" BorderThickness="4" >
    <ScrollViewer Margin="5" HorizontalScrollBarVisibility="Auto">
        <Canvas Background="Azure"
            ScrollViewer.HorizontalScrollBarVisibility="Auto"
            ScrollViewer.VerticalScrollBarVisibility="Visible"
            Margin="5"
            MinHeight="400"
            MinWidth="400">
            <Ellipse Fill="White" Stroke="Black" Height="79"
                     Width="79" Canvas.Left="145" Canvas.Top="192"/>
            <Ellipse Fill="White" Stroke="Black" Height="42"
```

```
                    Width="42" Canvas.Left="263" Canvas.Top="300"/>
        </Canvas>
    </ScrollViewer>
</Border>
</UserControl>
```

در این مثال یک Canvas ، داخل شیء ScrollViewer قرار گرفته است. تنظیم ویژگی HorizontalScrollBarVisibility به Auto در ScrollViewer به این معنا است که در صورت نیاز، ScrollBar مربوطه نمایش داده شود؛ در غیر اینصورت لزومی به نمایش آن نیست. همچنین این خواص را به صورت Attached properties نیز در Canvas می‌توان تنظیم نمود. تنظیم ویژگی VerticalScrollBarVisibility به Visible به این معنا است که ScrollBar عمودی همواره باید نمایان باشد.

نکته‌ی جدید دیگری در این مثال معرفی شده است و آن استفاده از ویژگی‌های MinHeight و MinWidth می‌باشد که بدون وجود آن‌ها ScrollViewer معرفی شده در اینجا به درستی کار نخواهد کرد. استفاده از ویژگی‌های متداول Width و Height به معنای مقید نمودن اندازه‌ی طول و عرض اشیاء در Silverlight است که عموما توصیه نمی‌شود زیرا اشیاء قرار گرفته در صفحه‌ی مرورگر باید بتوانند با اندازه‌های متفاوت آن و یا اندازه‌های متفاوت قلم‌های (fonts) تعریف شده سازگار باشند. اما گاهی از اوقات نیاز است تا بتوان یک حداقل و یا حداکثری را جهت طول و عرض اشیاء تعریف نمود. به همین منظور ویژگی‌های MaxWidth ، MinWidth ، MinHeight و MaxHeight در Silverlight ارائه شده‌اند. برای مثال MinWidth به این معنا است که با تغییر اندازه‌ی مرورگر و کوچکتر شدن طول و عرض صفحه‌ی نمایشی آن، شیء مورد نظر حداقل به مقدار تعیین شده در این خاصیت باید قابل مشاهده باشد (همین تنظیم سبب پدیدار شدن ScrollBar افقی می‌شود). MaxWidth هم اگر مقدار دهی می‌شد، سبب می‌گردید با طول و عرض بالای یک مرورگر، اندازه‌ی عرض Canvas از مقدار تعیین شده بیشتر نشود.

نمایش صفحه‌ی جاری در حالت تمام صفحه

به سادگی با استفاده از کد نویسی می‌توان نحوه‌ی نمایش صفحه‌ی جاری را تبدیل به حالت تمام صفحه نمود که در مثال بعد به این مورد خواهیم پرداخت.

شکل ۱۷– نمایی از مثال نمایش تمام صفحه

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication21.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"
    FontFamily="Tahoma"
    d:DesignHeight="300" d:DesignWidth="400">
    <Border BorderBrush="CornflowerBlue" BorderThickness="1"
            CornerRadius="5"
            FlowDirection="RightToLeft"
            MaxWidth="300" MaxHeight="80">
        <Grid ShowGridLines="True" Background="Azure"
          HorizontalAlignment="Center"
          MinWidth="300"
          VerticalAlignment="Center">
            <Grid.RowDefinitions>
                <RowDefinition Height="*"></RowDefinition>
                <RowDefinition Height="Auto"></RowDefinition>
            </Grid.RowDefinitions>

            <TextBlock Margin="10" Grid.Row="0" Text="نمایش تمام صفحه:" />

            <StackPanel
            Grid.Row="1"
            HorizontalAlignment="Right"
            Orientation="Horizontal">
                <Button Name="btnFull" Margin="10,10,2,10"
                    Padding="3" Content="ورود" Click="btnFull_Click" />
                <Button Name="btbClose" Margin="2,10,10,10" Padding="3"
                    Content="خروج" Click="btnClose_Click" />
            </StackPanel>
        </Grid>
    </Border>
</UserControl>
```

کدهای متناظر با این صفحه جهت انتقال به حالت تمام صفحه و یا بازگشت به حالت قبل به شرح زیر هستند (به دلایل امنیتی تنها در حالت کلیک و فرمان مستقیم یک کاربر، انتقال به حالت تمام صفحه وجود دارد و سایر حالات ندید گرفته می‌شوند. همچنین دسترسی به صفحه کلید نیز در این حالت محدود است):

MainPage.xaml.c

```
using System.Windows;
namespace SilverlightApplication21
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
        private void btnFull_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.Host.Content.IsFullScreen = true;
        }
        private void btnClose_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.Host.Content.IsFullScreen = false;
        }
    }
}
```

نکته‌ی جدیدی که به رابط کاربر این مثال افزوده شده است تنظیم ویژگی FlowDirection به حالت راست به چپ می‌باشد که از نگارش چهارم Silverlight به این مجموعه برای پشتیبانی از زبان‌های راست به چپ، همانند زبان فارسی افزوده گردیده است.

طراحی رابط کاربر این مثال نیز ترکیبی است از Grid و StackPanel . از HorizontalAlignment برای تعیین محل قرارگیری StackPanel استفاده شده و چون FlowDirection مجموعه به راست به چپ تنظیم گردیده است، محل StackPanel در سمت چپ Grid قرار خواهد گرفت.

اگر به خواص ارتفاع سطرها دقت نمائید، یکی از آن‌ها با * مقدار دهی شده است و دیگری با Auto .

```
<RowDefinition Height="*"></RowDefinition>
<RowDefinition Height="Auto"></RowDefinition>
```

این نحوه‌ی تنظیم بدین معنا است که پس از مشخص شدن ارتفاع سطر دوم بر اساس عناصر قرار گرفته در آن، مقدار باقیمانده‌ی فضا در اختیار ردیف اول قرار خواهد گرفت.