عنوان: آشنایی با WPF قسمت هفتم: DataContext بخش چهارم
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۷/۰۵ ۰:۳۵
آدرس: www.dotnettips.info
گروه‌ها: WPF

تا قسمت قبلی کنترل لیست را پر نمودیم. در این مقاله قصد داریم آخرین کنترلT یعنی تقویم را بایند کرده و یک نکته از Binding را جهت تکمیل کردن بحث بیان کنیم.

تقویم

در دروس گذشته اطلاعات را از متدی به نام GetPerson دریافت می‌کردیم که اطلاعات آن به شرح زیر است:

```
public static Person GetPerson()
        {

            return new Person()
            {
                Name = "Leo",
                Gender = true,
                ImageName = "man.jpg",
                Country = new Country()
                {
                    Id = 3, Name = "Angola"
                },
                FieldOfWork = new FieldOfWork[] { test.FieldOfWork.Actor, test.FieldOfWork.Producer },
                Date = DateTime.Now.AddMonths(-3)
            };
        }
```

تاریخ ثبت شده در بالا، به سه ماه قبل از تاریخ فعلی بر می‌گردد و حالا این تاریخ را به خصوصیت DisplayDate تقویم انتساب می‌دهیم:

```
Calendar DisplayDate="{Binding Date}" Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left"
Margin="10">
```

اگر از برنامه اجرا بگیرد می‌بینید که تقویم روی سه ماه پیش قرار گرفته است؛ ولی تاریخی روی صفحه انتخاب نشده است و دلیل آن هم این است که این خصوصیت، تقویم را به جایی میبرد که آن تاریخ در آن ذکر شده است، ولی تاریخی روی صفحه انتخاب نمی‌کند. به همین علت در اکثر موارد در کنار خاصیت DisplayDate، از خاصیت SelectedDate هم استفاده می‌شود. این خاصیت بر خلاف خاصیت قبلی، تقویم را حرکت نمی‌دهد ولی تاریخ را انتخاب می‌کند. پس در این حالت ما هر دو گزینه را بایند می‌کنیم که هم تقویم به محل تاریخ حرکت کرده و هم تاریخ مد نظر انتخاب شود:

```
<Calendar DisplayDate="{Binding Date}" SelectedDate="{Binding Date}" Grid.Row="4" Grid.Column="1"
HorizontalAlignment="Left" Margin="10">
```

ادامه مفاهیم بایندینگ

در قسمت پنجم، دیدیم که چطور می‌توانیم با استفاده از متد OnPropertyName، برنامه را از تغییراتی که در سطح مدل می‌گذرد، آگاه کنیم و این تغییرات جدید را دریافت کرده و اطلاعات نمایش داده شده را به روز کنیم. در اینجا قصد داریم خلاف اینکار را با استفاده از همان متد انجام دهیم. یعنی مدل را از تغییراتی که در سطح UI می‌گذرد، آگاه کنیم.

این مثال را روی خصوصیت Name مدل اجرا می‌کنیم:

در Xaml Editor تگTextBox مربوط به نام شخص را به شکل زیر تغییر می‌دهیم:

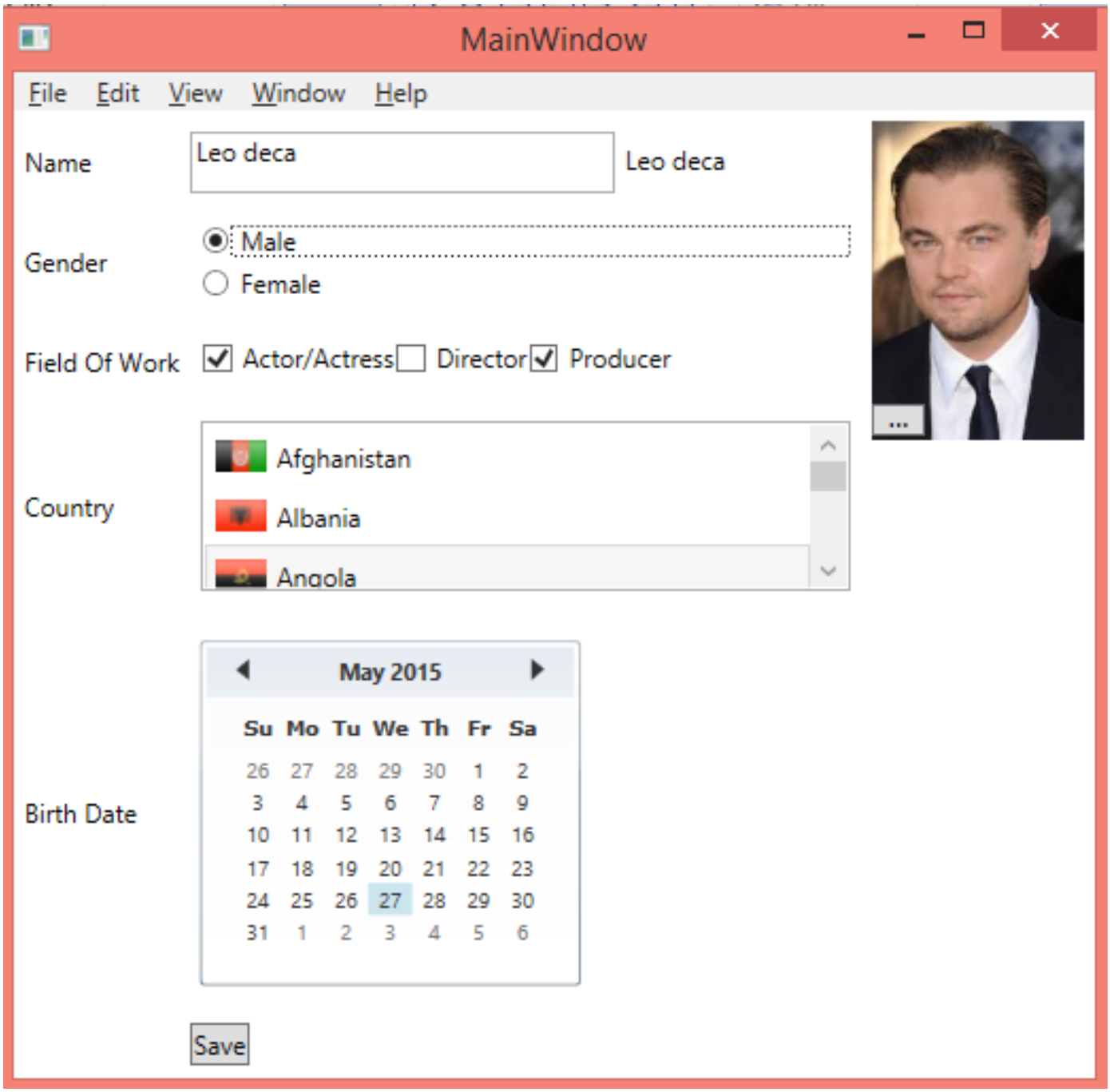
```
<TextBox Grid.Row="0" Grid.Column="1" Name="Txtname" Text="{Binding Path=Name,Mode=TwoWay}"
HorizontalAlignment="Left" Margin="5" Width="200" ></TextBox>
```

تغییری که در این حالت رخ داده است، افزودن ویژگی به نام Mode است که روی گزینه TwoWay تنظیم شده است. در قسمت‌های قبلی تمامی بایندینگ‌ها به طور پیش فرض روی حالت یک طرفه OneWay قرار داشتند، ولی در اینجا ما بایندینگ را دو طرفه اعمال کرده‌ایم. حال به همین سادگی هر تغییری که در این TextBox رخ دهد به مدل هم اعمال خواهد شد.

حال برای تست این مورد، عنصر زیر را در کنار نام شخص به صفحه اضافه می‌کنیم. یک برچسب متنی که به خاصیت Name متصل است و از تغییراتی که در سطح مدل داده می‌شود، آگاه است:

```
<TextBlock Grid.Column="1" Text="{Binding Path=Name}"  Grid.Row="0" VerticalAlignment="Center"
HorizontalAlignment="Left" Margin="210,10,0,13" RenderTransformOrigin="0.555,1.283" ></TextBlock>
```

اینک برنامه را اجرا می‌کنیم و فیلد متنی نام را ویرایش می‌کنیم. اگر فوکوس را از این کنترل بگیریم، می‌بینید که فیلد متنی هم به مقدار جدید تغییر می‌کند. اتفاق جدیدی که در اینجا افتاد این بود که مدل از تغییراتی که در سطح UI رخ داده بود، آگاه شد و بعد از آن فیلد متنی همانطور که قبلا با آن آشنا شده‌ایم از تغییری که در مدل رخ داده است آگاه شده است.



از دیگر مقادیر Mode می‌توان به جدول زیر اشاره کرد:

OneWayToSource	در این حالت، مدل از تغییرات سطح UI آگاه می‌شود ولی بقیه کنترل‌ها یا المان‌ها را از تغییرات خود آگاه نمی‌کند.
OneTime	در این حالت تنها یکبار مدل داده‌های خود را کنترل کرده (همان پر کردن اولیه داده‌ها) و دیگر هیچ نوع تغییراتی را رصد نمی‌کند.

تا به اینجا یک سری پیش نیازها را یاد گرفتیم. ولی روشی را که تا به اینجا استفاده کرده‌ایم یک روش اشتباه و قدیمی است که در winform هم انجام می‌دادیم. یعنی هنوز وابستگی بین رابط کاربری و منطق برنامه وجود دارد. در قسمت بعدی در مورد M-V-VM صحبت خواهیم کرد و از طریق viewmodel ارتباط بین مدل و ویو را ایجاد خواهیم کرد. در این روش دیگر نیازی نیست که بدانید کنترلی به اسم textbox1 وجود دارد یا خیر یا حتی اصلا اسمی دارد یا خیر و این یعنی جدایی رابط کاربری و منطق برنامه و اصل هدف WPF.

دانلود مثال