عنوان: **آشنایی با WPF قسمت پنجم : DataContext بخش اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۲۱ ۲۲:۵۰
آدرس: www.dotnettips.info
گروه‌ها: MVVM, WPF

یکی از مهمترین قسمت‌های برنامه، کار با داده‌های بانک اطلاعاتی (یا در کل منابع اطلاعاتی) است. اینکه چگونه با آن‌ها ارتباط برقرار کنیم و آن‌ها را در یک قالب کاربر پسند به کاربران برنامه نشان دهیم. افزودن شیء DataContext و مفاهیمی چون DataBinding باعث ارتباط سریع‌تر و راحت‌تری با منبع داده‌ها شده است. همچنین این قابلیت وجود دارد که هر گونه به روز آوری در اطلاعات دریافت شده، شما را با خبر سازد تا بتوانید طبق آن چه که می‌خواهید اطلاعات نمایشی را به روز کنید. در این مقاله به نحوه‌ی ارتباط بین منبع داده با DataContext و سپس کنترل‌هایی را چون Grid و ListBox و ... در رابطه با این منابع داده بررسی می‌کنیم.

در مورد بررسی ارتباط با داده‌ها در WPF باید سه مورد را بشناسیم:

DataContext: این شیء اتصالش را به منبع داده‌ها برقرار کرده و هر موقع داده‌ای را نیاز داریم، از طریق این شیء تامین می‌شود.
DataBinding: یک واسطه بین DataContext و هر آن چیزی است که قرار است از داده‌ها تغذیه کند. در تعریفی رسمی‌تر می‌گوییم: روشی ساده و قدرتمند بوده و واسطی است بین مدل تجاری و رابط کاربری. هر زمانی که داده‌ای تغییر کند، ما را آگاه می‌سازد که می‌تواند یک ارتباط یک طرفه یا دو طرفه باشد.

DataTemplate: نحوه‌ی فرمت بندی و نمایش داده‌ها را تعیین می‌کند.
ابتدا کار را با یک مثال ساده آغاز می‌کنیم. قصد داریم فرمی را که در قسمت قبلی ساختیم، با استفاده از یک منبع داده پر کنیم:
ابتدا قبل از هر چیزی کلاس فرم قبلی را پیاده سازی می‌کنیم. در این پیاده سازی از یک enum برای انتخاب زمینه‌های کاری هم کمک گرفته ایم و هچنین با یک متد ایستا، منبع داده‌ی تک رکوردی را جهت تست برنامه آماده کرده‌ایم:

```
public enum FieldOfWork
    {
        Actor=0,
        Director=1,
        Producer=2
    }
    public class Person
    {
        public string Name { get; set; }

        public bool Gender { get; set; }

        public string ImageName { get; set; }

        public string Country { get; set; }

        public DateTime Date { get; set; }

        public IList<FieldOfWork> FieldOfWork { get; set; }
        public static Person GetPerson()
        {
            return new Person()
            {
                Name = "Leo",
                Gender = true,
                ImageName ="man.jpg",
                Country = "Italy",
                Date = DateTime.Now
            };
        }
    }
```

حالا لازم است که این منبع داده را در اختیار DataContext بگذاریم. وارد بخش کد نویسی شده و در سازنده‌ی پنجره کد زیر را می‌نویسیم:

```
DataContext = Person.GetPerson();
```

```
public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            DataContext = Person.GetPerson();
        }
    }
```

با این کار، ارتباط شما با منبع داده آغاز می‌شود و طبق درخواست‌هایی که از DataBinding به آن می‌رسد، اطلاعات را تحویل DataBinding می‌دهد. برای نمایش داده‌ها در کنترل‌ها و استفاده از DataBinding، به سراغ خصوصیات وابسته می‌رویم. در حال حاضر فعلا برنامه را با دو کنترل عکس و نام که رشته‌ای هستند آغاز می‌کنیم؛ چون بقیه‌ی کنترل‌ها کمی متفاوت هستند.
همانطور که می‌دانید متن کنترل TextBox توسط خصوصیت Text پر می‌شود و برای همین در این خصوصیت می‌نویسیم:

```
Text="{Binding Name}"
```

علامت {} را باز کرده و در ابتدا نام Binding را می‌آوریم. سپس بعد از یک فاصله، نام پراپرتی کلاسی را که حاوی اطلاعات مدنظر است، می‌نویسیم و بدین صورت اتصال برقرار می‌شود. برای کنترل عکس هم وضعیت به همین صورت است:

```
Source="{Binding ImageName}"
```

حال برنامه را اجرا کرده و دو کنترل textbox و Image را بررسی می‌کنیم:

کلمه‌ی Leo داخل کادر متنی قرار گرفته و عکس اینبار به صورت ایستا خوانده نشده، بلکه نام عکس از طریق یک منبع داده برای آن فراهم شده است.

اطلاع از به روزرسانی در منبع داده‌ها:
حال این نکته پیش می‌آید که اگر همین اطلاعات دریافت شده در مدل منبع داده تغییر کند، چگونه می‌توانیم از این موضوع مطلع شده و همین اطلاعات به روز شده را که نمایش داده‌ایم، تغییر دهیم. بنابراین جهت اطلاع از این مورد، کد را به شکل زیر تغییر می‌دهیم.

کار را از یک کلاس آغاز می‌کنیم. از اینترفیس INotifyPropertyChanged ارث بری کرده و در آن یک رویداد و یک متد را تعریف

می‌کنیم و کمی در هم در تعریف Propertyها دست می‌بریم. فعلا اینکار را فقط برای پراپرتی Name انجام می‌دهیم:

```
private string _name;
        public string Name
        {
            get { return _name; }
            set
            {
                _name = value;
                OnPropertyChanged("Name");
            }
        }
        public event PropertyChangedEventHandler PropertyChanged;

        private void OnPropertyChanged(string property)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(property));
            }
        }
```

در کد بالا یک رویداد از نوع PropertyChangedEventHandler تعریف می‌کنیم که وظیفه‌ی به روزآوری را به عهده دارد؛ ولی صدا زدن این رویداد بر عهده‌ی ماست و خود به خود صدا زده نمی‌شود. پس نیاز است متدی را فراهم کرده تا بدانیم که چه خصوصیتی تغییر یافته‌است و از آن طریق رویداد را فراخوانی کنیم و به رویداد بگوییم که کدام پراپرتی تغییر کرده است. این متد را OnpropertyChanged می‌نامیم که آرگومان ورودی آن نام خصوصیتی است که تغییر یافته است و پس از ارزیابی از صحت آن، رویداد را Invoke می‌کنیم.

در بخش Setter آن خصوصیت هم باید این متد را صدا زده و نام خصوصیت را به آن پاس بدهیم تا موقعی که مدل تغییر پیدا کرد، بگوید که خصوصیت Name بوده است که تغییر کرده است.

برای اینکه بدانیم کد واقعا کار می‌کند و تستی بر آن زده باشیم، فعلا دکمه‌ی Save را به Change تغییر می‌دهیم و کد داخل پنجره را بدین صورت تغییر می‌دهیم:

```
public partial class MainWindow : Window
    {
        private Person person;
        public MainWindow()
        {
            InitializeComponent();
            person = Person.GetPerson();
            DataContext = person;

        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            person.Name = "Leonardo Decaperio";
        }
    }
```

متغیر کلاسی را از حالت محلی Local به عمومی Global تغییر دادم که از طریق دکمه‌ی منبع داده در دسترس باشد. حال در رویداد دکمه نام بازیگر را تغییر می‌دهم. برنامه را اجرا کنید و بر روی دکمه کلیک کنید. باید بعد از یک لحظه‌ی کوتاه، نام بازیگر از Leo به Leonardo Decaperio تغییر کند.

این کد واقعا کدی مفید جهت به روزرسانی است ولی مشکلی دارد که نام پراپرتی باید به صورت String به آن پاس شود که در یک برنامه بزرگ این مورد یک مشکل خواهد شد و اگر نام خصوصیت تغییر کند باید نام داخل آن هم تغییر کند؛ پس کد را به شکل دیگری بازنویسی می‌کنیم:

```
private string _name;
        public string Name
        {
            get { return _name; }
            set
            {
                _name = value;
```

```
                OnPropertyChanged();
            }
        }

 private void OnPropertyChanged([CallerMemberName] string property="")
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(property));
            }
        }
```

در متد OnPropertyChanged در کنار پارامتر اول، ویژگی attribute به نام CallerMemberName را که در فضای نام system.runtime.compilerservice قرار دارد استفاده می‌کنیم (دات نت 4.5). این ویژگی، نام پراپرتی یا متدی که متد OnpropertyChnaged را صدا زده است، به دست می‌آورد. پارامتر اول را هم اختیاری می‌کنیم که سیستم بر ورود پارامتر اجباری نداشته باشد و نهایتا در هر پراپرتی تنها لازم است همانند بالا، خط زیر ذکر شود:

```
OnPropertyChanged();
```

اگر الان یک تست از آن بگیرید، می‌بینید که بدون مشکل کار می‌کند. حالا همین متد را در setter تمام پراپرتی‌هایی که دوست دارید از تغییر آن‌ها آگاه شوید قرار دهید.

کد این قسمت

در قسمت‌های آینده به بررسی تبدیل مقادیر و framework element و کنترل‌ها می‌پردازیم.

نظرات خوانندگان

نویسنده: مرتضی رییسی
تاریخ: ۱۳۹۴/۰۲/۲۲ ۲۳:۱۸

ممنون بابت تمام زحماتتون.
سوال: رخداد PropertyChanged کی و کجا مقدار دهی میشه؟ طبق برنامه شما هیچ وقت این رخداد مقداری نمیگیره و همیشه نال خواهد بود. لطفاً راهنمایی بفرمایید.
در ضمن اگه میشه مثالها و کلاسها رو بصورت کامل (using و namespace و بدنه کامل کلاس) بذارین و به تکه کد کوتاهی بسنده نکنید تا کمتر دچار سردرگمی بشیم.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۲۳ ۰:۱۹

اگه درست منظورتون رو متوجه شدم باشم پاسخ شما در متن بالا قرار داره:
"در متد OnPropertyChanged در کنار پارامتر اول، ویژگی attribute به نام CallerMemberName را که در فضای نام system.runtime.compilerservice قرار دارد استفاده می‌کنیم (دات نت 4.5). این ویژگی، نام پراپرتی یا متدی که متد OnpropertyChnaged را صدا زده است، به دست می‌آورد. "
تعریف ویژگی CallermemberName قبل از پارامتر باعث می‌شود به طور خودکار نام پراپرتی Name به سمت متد ارسال شود. یعنی با کد زیر برابری خواهد کرد:

```
OnPropertyChanged("Name");
```

test-1da6cfbbddbf48e1bd8b88c0c24a2c7b.zip

نویسنده: مرتضی رییسی
تاریخ: ۱۳۹۴/۰۲/۲۳ ۱۸:۴۱

مشکل پیدا شد.

```
public  class Person:INotifyPropertyChanged
```

اینترفیس رو پیاده سازی نکرده بودم. ممنون