عنوان: Markup Extensions در XAML
نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۵/۰۷ ۲۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: WPF, XAML

Markup Extensionها برای مواردی استفاده می‌شوند که قرار است مقداری غیر از یک مقدار ثابت و یک نوع قابل شناسایی در XAML برای یک value تنظیم شود. تمام مواردی در XAML که درون {} قرا می‌گیرند همان Markup Extensionها هستند. مانند Binding و یا StaticResoiurces.
علاوه بر Markup Extensionهای از پیش تعریف شده در XAML، می‌توان Markup Extensionهای شخصی را نیز تولید کرد. در واقع به زبان ساده‌تر Markup Extension برای تولید ساده‌ی داده‌های پیچیده در XAML استفاده می‌شوند.

لازم به ذکر است کهMarkup Extension ها می‌توانند به دو صورت Attribute Usage ،درون {} :

```
"{Binding path=something,Mode=TwoWay}”
```

و یا Property Element Usage (همانند سایر Element هایWPF) درون <> استفاده شوند:

```
<Binding Path="Something" Mode="TwoWay"></Binding>
```

برای تعریف یک Markup Extension، یک کلاس ایجاد می‌کنیم که از Markup Extensions ارث بری می‌کند. این کلاس یک Abstract Method به نام ProvideValue دارد که باید پیاده سازی شود. این متد مقدار خصوصیتی که Markup Extensions را فراخوانی کرده به صورت یک Object بر می‌گرداند که یکبار در زمان Load برای خصوصیت مربوطه‌اش تنظیم می‌شود.

```
public abstract Object ProvideValue(IServiceProvider serviceProvider)
```

همانطورکه ملاحظه می‌کنید ProvideValue یک پارامتر IServiceProvider دارد که ازطریق آن می‌توان به IProvideValueTarget دسترسی داشت. ازاین Interface برای گرفتن اطلاعات کنترل(TargetObject) و خصوصیتی (TargetProperty) که فراخوانی را انجام داده در صورت لزوم استفاده می‌شود.

```
var target = serviceProviderGetService(typeof(IProvideValueTarget))as IProvideValueTarget;
var host = targetTargetObject as FrameworkElement;
```

Markup Extensionها می‌توانند پارامترهای ورودی داشته باشند:

```
public class ValueExtension : MarkupExtension
{
  public ValueExtension () { }
  public ValueExtension (object value1)
  {
    Value1 = value1;
  }
   public object Value1 { get; set; }
   public override object ProvideValue(IServiceProvider serviceProvider)
   {
     return Value1;
   }
}
```

و برای استفاده در فایل Xaml:

```
<TextBox  Text="{app:ValueExtension test}" ></TextBox>
```

و یا می‌توان خصوصیت هایی ایجاد کرد و از آنها برای ارسال مقادیر به آن استفاده کرد:

```
<TextBox  Text="{app:ValueExtension Value1=test}" ></TextBox>
```

تا اینجا موارد کلی برای تعریف و استفاده از Markup Extensions گفته شد. در ادامه یک مثال کاربردی می‌آوریم. برای مثال در نظر بگیرید که نیاز دارید DataType مربوط به یک DataTemplate را برابر یک کلاس Generic قرار بدهید:

```
public class EntityBase
{
   public int Id{get;set}
}

public class MyGenericClass<TType> where TType : EntityBase
{
   public int Id{get;set}
   public string Test{  get;set; }

In XAML:

<DataTemplate DataType="{app:GenericType ؟}">
```

برای انجام این کار یک Markup Extensions به صورت زیر ایجاد می‌کنیم که Type را به عنوان ورودی گرفته و یک نمونه از کلاس Generic ایجاد می‌کند:

```
public class GenericType : MarkupExtension
{
  private readonly Type _of;
  public GenericType(Type of)
  {
     _of = of;
  }
  public override object ProvideValue(IServiceProvider serviceProvider)
  {
      return typeof(MyGenericClass<>)MakeGenericType(_of);
 }
}
```

و برای استفاده از آن یک نمونه از MarkupExtension ایجاد شده ایجاد کرده و نوع Generic را برای آن ارسال می‌کنیم:

```
<DataTemplate DataType="{app:GenericType app:EntityBase}">
```

این یک مثال ساده از استفاده از Markup Extensions است. هنگام کار با WPF می‌توان استفاده‌های زیادی از این مفهوم داشت، برای مثال زمانی که نیاز است ItemsSource یک Combobox از Descriptionهای یک Enum پر شود می‌توان به راحتی با نوشتن یک Markup Extensions ساده این عمل و کارهای مشابه زیادی را انجام داد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۸ ۲۳:۵۰

یک مثال جالب در این مورد

DelayBinding: a custom WPF Binding

عنوان: **Debug کردن Binding در XAML**
نویسنده: بزمی
تاریخ: ۱۳۹۲/۰۵/۰۹ ۱۵:۰
آدرس: www.dotnettips.info
برچسب‌ها: Debugging, Silverlight, WPF, XAML, Binding

اغلب در حین Bind کردن Propertyها در XAML به مشکل Bind نشدن بر می‌خوریم. من معمولا از روش زیر استفاده می‌کنم:

```
public class DatabindingDebugConverter : IValueConverter
    {
        #region IValueConverter Members

        public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            Debugger.Break();
            return value;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            Debugger.Break();
            return value;
        }

        #endregion IValueConverter Members
    }
```

و در XAML :

```
<DataTemplate.Resources>
            <debug:DatabindingDebugConverter x:Key="databindingDebugConverter"/>
</DataTemplate.Resources>
<DataGrid ItemsSource="{Binding myViewModel,Converter={StaticResource databindingDebugConverter}}" />
```

و حال دو حالت می‌تواند اتفاق بیفتد :

1 - Break Point Hit نمی‌شود:

در این حالت مقدار myViewModel خالی (null) است و یا اصلا myViewModel در DataContext مربوط به DataGrid وجود ندارد در این صورت همچنین در پنجره Out Put Visual Studio:

```
System.Windows.Data Error: 35 : BindingExpression path error: 'X' property not found ...
```

و با search متن "System.Windows.Data Error: 35 : BindingExpression path error:" در Out Put میتوان متوجه آن شد.

2 - Break Point Hit می‌شود:

در این حالت باید value را Watch کنیم (Shift+F9) تا ببینیم علت Bind نشدن چیست؟ شاید (در این مورد خاص) نوع myViewModel از IEnumerable نباشد ...

در حین بررسی و Debug ، شاید گاهی مسئاله لاینحل به نظر برسد ، ولی به نظر من معمولا با کم و زیاد کردن آدرس (Binding Path) به یکی از دو حالت بالا خواهیم رسید ،
مثلا زمانی که Path به صورت myViewModel.MyProperty.MyInnerPtoperty است ، باید Path را با حالات زیر توسط Converter مذکور تست کنیم:

```
Binding"{Path=myViewModel.MyProperty.MyInnerPtoperty ,Converter="{StaticResource debugger}}"
Binding"{Path=myViewModel.MyProperty,Converter="{StaticResource debugger}}"
Binding"{Path=myViewModel,Converter="{StaticResource debugger}}"
Binding"{Path=.,Converter="{StaticResource debugger}}"
```

امیدوارم از Binding تان لذت ببرید.

عنوان: **Bind کردن Enum به ItemsSource در XAML**
نویسنده: بزمی
تاریخ: ۱۳۹۲/۰۵/۱۱ ۸:۰
آدرس: www.dotnettips.info
برچسب‌ها: C#, Silverlight, WPF, XAML, Binding

شاید تا به حال در یک برنامه سازمانی نیاز به Bind کردن یک Enum به کنترل‌های XAML به چشمتان خورده باشد ، روشی که من برای این کار استفاده می‌کنم توسط یک Markup Extension به صورت زیر است :

```
public class ByteEnumerationExtention : MarkupExtension
    {
        public ByteEnumerationExtention(Type enumType)
        {
            this.enumType = enumType;
        }

        private Type enumType;

        public Type EnumType
        {
            get { return enumType; }
            private set
            {
                enumType = value;
            }
        }

        public override object ProvideValue(IServiceProvider serviceProvider)
        {
            return (from x in Enum.GetValues(EnumType).OfType<Enum>()
                    select new EnumerationMember
                    {
                        Value = GetValue(x),
                        Description = GetDescription(x)
                    }).ToArray();
        }

        private byte? GetValue(object enumValue)
        {
            return Convert.ToByte(enumValue.GetType().GetField("value__").GetValue(enumValue));
        }

        public object GetObjectValue(object enumValue)
        {
            return enumValue.GetType().GetField("value__").GetValue(enumValue);
        }

        public string GetDescription(object enumValue)
        {
            var descAttrib = EnumType.GetField(enumValue.ToString())
                .GetCustomAttributes(typeof(DescriptionAttribute), false)
                .FirstOrDefault() as DescriptionAttribute;
            return descAttrib != null ? descAttrib.Description : enumValue.ToString();
        }
    }

    public class EnumerationMember
    {
        public string Description { get; set; }

        public byte? Value { get; set; }
    }
```

: XAML

```
<ComboBox ItemsSource="{Binding Source={ Extensions:ByteEnumerationExtention {x:Type type:MyEnum} }}"
DisplayMemberPath="Description"
 SelectedValuePath="Value" SelectedValue="{Binding SelectedItemInViewModel}"/>
```

: Enum

```
public enum MyEnum : short
    {
        [Description("گزینه 1")]
        Item1 = 1,

        [Description("گزینه 2")]
        Item1 = 2,

        [Description("گزینه 3")]
        Item1 = 3,

        [Description("گزینه 4")]
        Item1 = 4,

        [Description("گزینه 5")]
        Item1 = 5,
.
.
.
    }
```

SelectedItem در ViewModel :

```
short? selectedItemInViewModel;
        public short? SelectedItemInViewModel
        {
            get
            {
                return selectedItemInViewModel;
            }
            set
            {
                selectedItemInViewModel = value;
                 RaisePropertyChanged("SelectedItemInViewModel");
               //do calculations if needed
            }
        }
```

عنوان: **دسترسی به فیلد های Static در XAML**
نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۵/۱۵ ۲۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: WPF, XAML

MarkupExtension ها قبلا در اینجا توضیح داده شده اند. یکی از MarkupExtensionهای از پیش تعریف شده x:Static است که برای مقداردهی یک خصوصیت در XAML با یک مقدار استاتیک استفاده می‌شود.اگر بخواهید از یک ثابت(constant)،یک خصوصیت استاتیک(static property)، یا یک مقدار از یک enumeration، برای مقداردهی یک خصوصیت در XAML استفاده کنید باید از این MarkupExtension استفاده کنید.

برای مثال برای یک استفاده از یک خصوصیت استایک به صورت زیر عمل می‌کنیم:

```
namespace Test
 {
   public class Constants
   {
      public static readonly string ConstantString = "Test";
   }
 }
```

توجه داشته باشید که برای استفاده از این ثابت باید ابتدا فضای نام مربوط به آن را تعریف کنید.

```
xmlns:test="clr-namespace:ItemTest "
<Label Content="{x:Static test:Constants.ConstantString}" />
```

و یا برای مقدار دهی از طریق یک Enumeration

```
namespace Test
{
  public enum VisiblityEnum
     {
         Collapse,
         Hidden,
         Visible
     };
}
```

و در فایل XAML:

```
xmlns:test="clr-namespace:Test"
<Label Content="{x:Static test:VisiblityEnum.Collapse}" />
```

برای استفاده از یک ثابت نیز به همین صورت عمل می‌کنیم.

عنوان: **حرکت روی سلول های دیتا گرید با فشردن کلید Enter در برنامه های WPF**
نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۱۱/۲۳ ۱۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: MVVM, WPF, XAML

کاربران بیشتر برنامه‌های فارسی تمایل دارند که توسط کلیدEnter درون فرم‌ها حرکت کنند.در برنامه‌های WPF و مخصوصا زمانی که شما از الگوی MVVM استفاده می‌کنید ،انجام این کار اگر از روش‌های مناسب استفاده نکنید تا حدودی سخت می‌شود.
برای حرکت روی TextBoxها و کنترل‌های مشابه می‌توانید این کار را به راحتی با Register کردن رویداد مربوط به آن نوع کنترل‌ها توسط EventManager یک بار در ابتدای برنامه انجام دهید.

```
public partial class App : Application
{
EventManager.RegisterClassHandler(typeof(TextBox), TextBox.KeyDownEvent, new
KeyEventHandler(TextBox_KeyDown));
     ...
}
 private void TextBox_KeyDown(object sender, KeyEventArgs e)
   {
         if (e.Key != Key.Enter)
             return;
         var focusedElement = Keyboard.FocusedElement as TextBox;
          focusedElement.MoveFocus(new TraversalRequest(FocusNavigationDirection .Next));
   }
```

اما همانطور که در عنوان مطلب آورده شده است در این مطلب تصمیم دارم حرکت روی سلول‌های دیتا گرید توسط کلید Enter را شرح بدهم.
برای این کار نیز یک راه حل ساده وجود دارد و آن شبیه سازی فراخوانی کلید Tab هنگام فشرده شدن کلید Enter است.چون همانطور که می‌دانید کلید Tab به صورت پیش فرض حرکت روی سلول‌ها را انجام می‌دهد.برای انجام آن کافی ست دیتاگرید خود را سفارشی کرده و در متد OnPreviewKeyDown عملیات زیر را انجام دهید:

```
public class CommonDataGrid : DataGrid
{
        protected override void OnPreviewKeyDown(KeyEventArgs e)
        {
            base.OnPreviewKeyDown(e);
            if (e.Key != Key.Enter || Keyboard.PrimaryDevice.ActiveSource == null) return;
            this.CommitEdit();
            var args = new KeyEventArgs
                (Keyboard.PrimaryDevice,
                Keyboard.PrimaryDevice.ActiveSource, 0, Key.Tab) { RoutedEvent = Keyboard.KeyDownEvent
};
            InputManager.Current.ProcessInput(args);
        }
}
```