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