WPF Data Template

lifeisforu@naver.com
최도경

- 다음 MSDN 의 내용을 중심으로 요약 보충한 글입니다.
  - Resources Overview
  - Data Templating Overview

Resources

- DataTemplate, Style, ControlTemplate 등을 이해하기 위해서는 resource 에 대한 이해가 선행되어야 한다.
- Resource 란 재사용할 수 있는 object 나 value 들을 의미한다. "어떤 type 의 instance 가 resoruce 다" 라는 정의는 없다는 것이다.
- 크게는 static resource 와 dynamic resource 로 나뉜다.

Example : Resource

```
<Page.Resources>
  <SolidColorBrush x:Key="MyBrush" Color="Gold"/>
```

- Page 는 FrameworkElement 를 상속했고, 이는 Resources 라는 dictionary 를 가지고 있다. 위 code 는 그 Resources 에 "MyBrush" 라는 key 를 사용해 SolidColorBrush 의 object 를 하나 등록해 달라는 의미이다.

Example : Resource(cont)

- Resource 를 사용하기 위해서는 WPF markup extension 을 사용한다.

```
<TextBlock Foreground="{StaticResource MyBrush}"
           Text="Text" Margin="20" />
```

- 이는 Forground 라는 property 의 값으로 "MyBrush" 라는 key 를 가진 static resource 를 할당해 달라는 요청이다.
- 이를 property element 표현으로 바꿔보자. 잘 동작하는가?

Example : Resource(cont)

- WPF markup extension 은 결국 property element 를 특별한 형태로 기술할 수 있도록 하는 것이라는 것을 기억하라.
- 항상 본질을 이해해야 한다.

```
<TextBlock Text="Text" Margin="20">
    <TextBlock.Background>
        <StaticResource ResourceKey="MyBrush"/>
    </TextBlock.Background>
</TextBlock>
```

Dynamic Resource

- 우리가 XAML 에서 등록한 resource 들은 static resource 였다. 정적으로 그 resource 가 생성된다.
- 하지만 Resources 에 실시간에 data 가 할당되는 경우는 어떻게 해야 할까?
- 이 경우에는 DynamicResource markup extension 을 사용해야만 한다.

Dynamic Resource(cont)

- Dynamic resource 추가.

```
public MainWindow()
{
    Resources.Add( "MyBrush", new SolidColorBrush( Colors.Red ) );

    InitializeComponent();
}
```

- Dynamic resource 사용.

```
<TextBlock.Background>
    <DynamicResource ResourceKey="MyBrush"/>
</TextBlock.Background>
```

DataTemplate

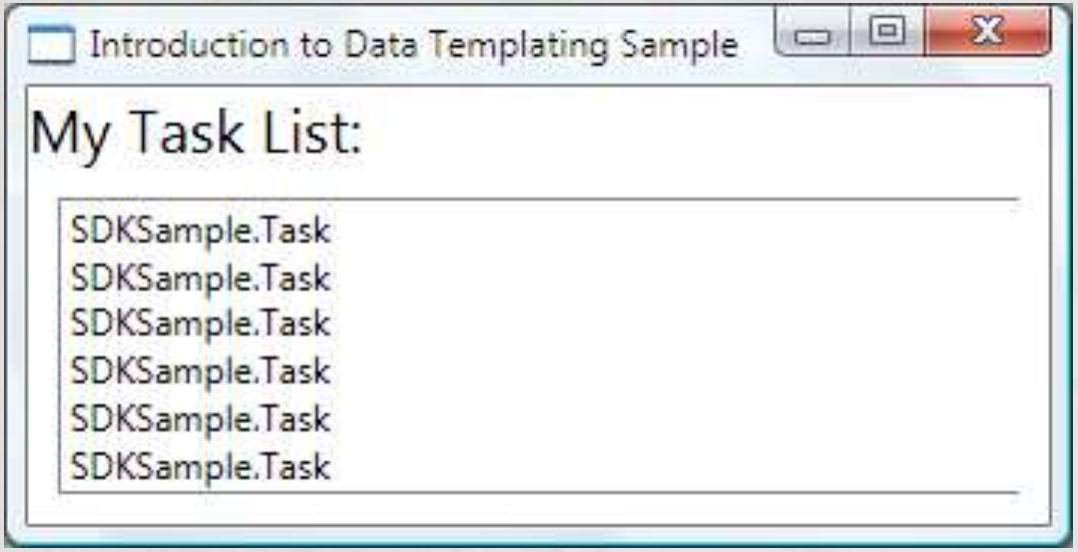
- DataTemplate 은 collection 을 data 로 받았을 때 이를 화면에 어떻게 출력할 것인가를 결정하기 위한 template 이다.
- 아래 예에서 Tasks 는 Task 를 item 으로 가지는 collection 이라 하자. 각 Task object 는 TaskName, Description, Priority 를 property 로 갖고 있다.

```
<Window.Resources>
  <local:Tasks x:Key="myTodoList"/>
```

```
<ListBox Width="400" Margin="10"
         ItemsSource="{Binding Source={StaticResource myTodoList}}"/>
```

Without a DataTemplate

- DataTemplate 을 사용하지 않았을 경우 기본적으로 다음과 같은 결과를 출력한다.

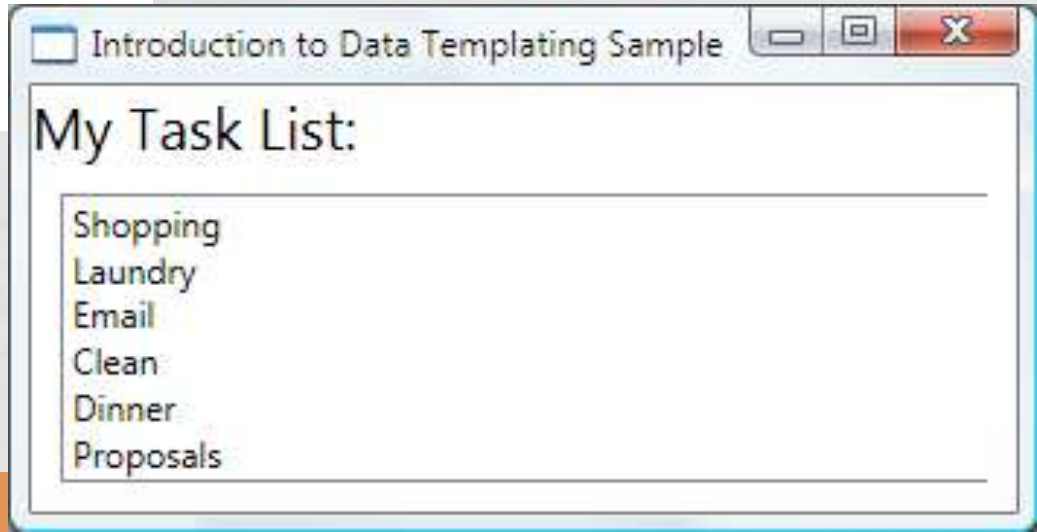


- 각 item 을 ToString() 했을 때의 결과이다.

Without a DataTemplate(cont)

- 만약 TaskName 을 출력해주고자 한다면 Task class 의 ToString() 을 overriding 하면 된다. TaskName 의 field 가 name 이라 가정한다.

```
public override string ToString()
{
    return name.ToString();
}
```



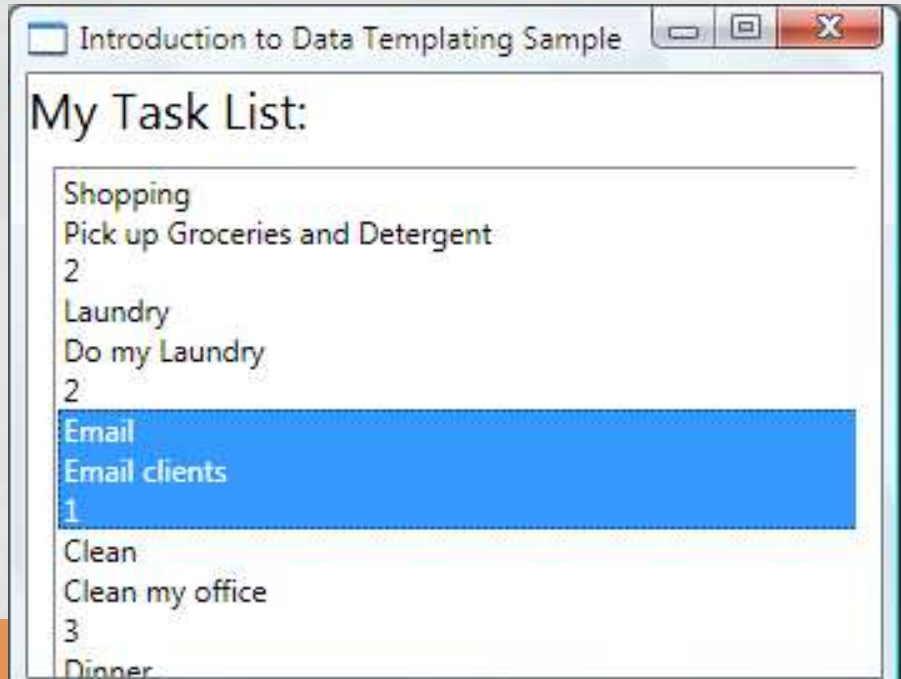
With a DataTemplate

- ListBox.DataTemplate property 에 DataTemplate object 를 넣어 원하는 형태로 출력해 보자.

```
<ListBox Width="400" Margin="10"
         ItemsSource="{Binding Source={StaticResource myTodoList}}">
   <ListBox.ItemTemplate>
     <DataTemplate>
       <StackPanel>
         <TextBlock Text="{Binding Path=TaskName}" />
         <TextBlock Text="{Binding Path=Description}"/>
         <TextBlock Text="{Binding Path=Priority}"/>
       </StackPanel>
     </DataTemplate>
   </ListBox.ItemTemplate>
 </ListBox>
```

With a DataTemplate(cont)

- Text="{Binding Path=TaskName}" 과 binding code 에서 느낄 수 있는 것은 무엇인가? DataTemplate.DataContext 에 collection 의 item 인 Task object 가 설정되었음을 알 수 있다.
- Binding.Source 를 지정하지 않는 경우는 DataContext 가 할당된 경우이다.



DataTemplate As a Resource

- Data template 을 공유하는 방법은 없을까? 우리가 resource 는 공유가능한 object 나 value 라고 했다. 실시간에 template 이 변할 이유는 없으므로 static resource 로 추가할 수 있다.

DataTemplate as a Resource(cont)

- 다음과 같이 정의할 수 있다.

```
<Window.Resources>
   <DataTemplate x:Key="myTaskTemplate">
     <StackPanel>
       <TextBlock Text="{Binding Path=TaskName}" />
       <TextBlock Text="{Binding Path=Description}"/>
       <TextBlock Text="{Binding Path=Priority}"/>
     </StackPanel>
   </DataTemplate>
</Window.Resources>
```

- 다음과 같이 사용할 수 있다.

```
<ListBox Width="400" Margin="10"
         ItemsSource="{Binding Source={StaticResource myTodoList}}"
         ItemTemplate="{StaticResource myTaskTemplate}"/>
```

DataTemplate : DataType

- 어떤 data template 을 특정 type 에 대해서 자동적으로 사용하도록 지정할 수 있다.

```
<DataTemplate DataType="{x:Type local:Task}">
  <StackPanel>
    <TextBlock Text="{Binding Path=TaskName}" />
    <TextBlock Text="{Binding Path=Description}"/>
    <TextBlock Text="{Binding Path=Priority}"/>
  </StackPanel>
</DataTemplate>
```

- Data template 을 static resource 로 만들고 DataType 을 지정한 후 ListBox 가 자동으로 해당 template 을 사용하는지 확인하라.

DataTemplate : DataType(cont)

- Data template resource 에 x:key 를 설정하게 되면 자동으로 해당 type 에 template 을 적용하는 mechanism 이 동작하지 않는다는 것에 주의 하라.
- Content control 에 들어가는 type 에도 자동으로 적용되지 않는다(data template 자체가 collection 을 위한 기능이다).

ContentTemplate

- 여기에서 잠시 content template 에 대해서 언급하고 넘어가겠다.
- ContentControl 의 내용은 Content property 에 들어가고, ItemsControl 의 내용은 ItemsSource property 에 들어 간다.
- ItemsControl 에는 DataTemplate 이 적용된다. 그러면 content control 도 이에 대응하는 것이 있지 않을까? ContentTemplate 이 바로 그것이다. 작성하는 법은 DataTemplate 과 같다.

ContentTemplate(cont)

- ContentControl 의 Content 에다가 collection 을 binding 하면 선택된 item 이 binding 된다.
- 앞의 예제에서 작성한 예제에서 ContentConrol object 를 하나 추가하고, 거기에다가 ContentTemplate 을 적용해 보라.

Excercise

- 여러 가지 layout 을 사용해서 앞의 예제들을 멋지게 꾸며 보자.