Triggers in WPF

Hiren Khirsaria, 21 Feb 2013 CPOL

★★★★★ 4.88 (40 votes)

Rate this: vote 1vote 2vote 3vote 4vote 5

How to use Property Trigger, Data Trigger, Event Trigger in WPF

- **Download WPF_Triggers-noexe.zip - 168.2 KB**
- **Download WPF_Triggers.zip - 324.1 KB**

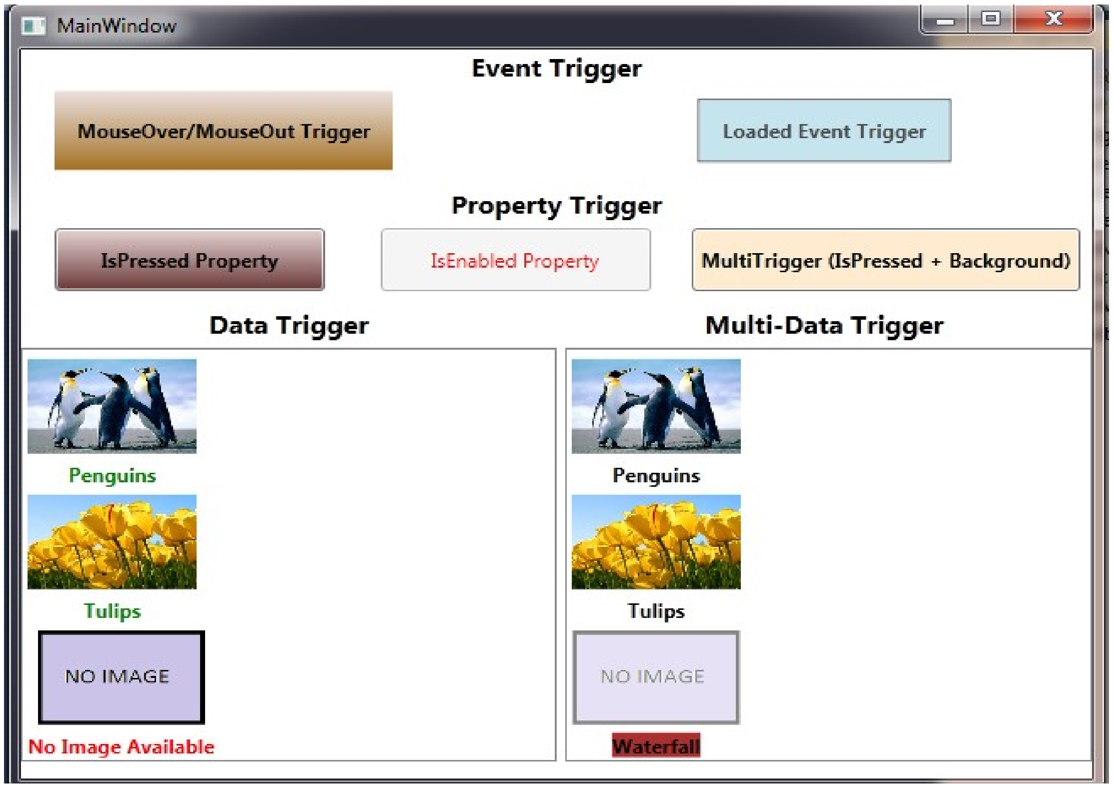
Introduction

Triggers is used in style to perform action on change any property value or event fires. Triggers create visual effects on controls. By using Triggers we can change the appearance of Framework Elements.

Background

There are three types of Triggers available.

1. Event Trigger
2. Property Trigger
3. Data Trigger



Using the code

1. Property Trigger

Property Trigger Executes Collections of Setters, when UIElements property value changes.

To create a trigger on any controls, you have to set trigger in style of the control.

Hide   Copy Code

```
<Style x:Key="ButtonStyle" TargetType="{x:Type Button}">
   <Style.Triggers>
      <Trigger Property="IsPressed" Value="True">
         <Setter Property="Opacity" Value="0.5" />
      </Trigger>
      <Trigger Property="IsEnabled" Value="False">
         <Setter Property="Foreground" Value="Red" />
      </Trigger>
   </Style.Triggers>
</Style>
```

In the above code, Trigger created on Button. It will set Opacity to 0.5 when Buttons IsPressed property change. you can set trigger on any Dependency Property of the control. Now you can apply style to Button.

Hide   Copy Code

```
<Button x:Name="PropertyTriggerButton"
    Width="160"
    Height="40"
    Margin="20,0,0,0"
    HorizontalAlignment="Left"
    Content="IsPressed Property"
    Cursor="Hand"
    FontWeight="Bold"
    Style="{StaticResource ButtonStyle}"
    ToolTip="Press To Raise Property Trigger">
```

When you press the button, the trigger executes.

MultiTrigger

MultiTrigger is used to set action on Multiple Property change. It will execute when all condition are satisfy within MulitTrigger.Condition.

Hide Copy Code

```
<Style x:Key="MulitTriggerButtonStyle" TargetType="Button">
    <Style.Triggers>
        <MultiTrigger>
            <MultiTrigger.Conditions>
                <Condition Property="IsPressed" Value="True" />
                <Condition Property="Background" Value="BlanchedAlmond" />
            </MultiTrigger.Conditions>
            <MultiTrigger.Setters>
                <Setter Property="Foreground" Value="Blue" />
                <Setter Property="BorderThickness" Value="5" />
                <Setter Property="BorderBrush" Value="Blue" />
            </MultiTrigger.Setters>
        </MultiTrigger>
    </Style.Triggers>
</Style>
```

2. Event Trigger

Event Trigger used perform action when RoutedEvent of FrameworkElement raise.

Event Trigger generally used to perform some animation on control (like : colorAnimation, doubleAnumation using KeyFrame etc.)

Let's first understand Storyboard and Animation.

Animations:

An animation can provide user interface more attractive with look and feel. We can also create visual effects on the control, Animation can be any type like:

- change background color of the control
- rotate screen in 90 degree angle
- move object from one location to another

- Change Opacity (FadeIn/FadeOut) of the circle.

Animation is used with Property of the UIElement. WPF provides different types of animation used with properties, like:

ColorAnimation : used to animate/change the color property (SolidColorBrush, LinearGradientBrush) of the UIElement over specific **Duration.** It has two properties **: From(source)** and **To(target)**

Hide Copy Code

```
<Border Name="border1" Width="100" Height="30"
   BorderBrush="Black" BorderThickness="1">
   <Border.Background>
      <SolidColorBrush x:Name="MyBorder" Color="LightBlue" />
   </Border.Background>
   <Border.Triggers>
      <EventTrigger RoutedEvent="Mouse.MouseEnter">
         <BeginStoryboard>
            <Storyboard>
             <ColorAnimation Duration="0:0:1"
                Storyboard.TargetName="MyBorder"
                Storyboard.TargetProperty="Color"
                To="Gray" />
            </Storyboard>
         </BeginStoryboard>
      </EventTrigger>
   </Border.Triggers>
 </Border>
```

ColorAnimationUsingKeyFrames: also works same as colorAnimation in Addition to animate object with KeyFrames.

Hide Copy Code

```
<Border Name="EventTriggerBorder"
   Width="100"
   Height="30"
   BorderBrush="Black"
   BorderThickness="1">
   <Border.Background>
      <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
         <GradientStop Offset="0" Color="#FFE9DDDA" />
         <GradientStop Offset="1" Color="#FF9F6A1D" />
      </LinearGradientBrush>
   </Border.Background>
   <Border.Triggers>
      <EventTrigger RoutedEvent="Mouse.MouseEnter">
         <BeginStoryboard>
            <Storyboard>
             <ColorAnimationUsingKeyFrames Storyboard.TargetName="EventTriggerBorder"
                Storyboard.TargetProperty="Background.GradientStops[0].Color">
                 <LinearColorKeyFrame KeyTime="0:0:1" Value="#FF00FF" />
                        </ColorAnimationUsingKeyFrames>
            </Storyboard>
         </BeginStoryboard>
      </EventTrigger>
   </Border.Triggers>
```

```
</Border>
```

This will animate first **GradientStop** color of Border Element from #FFE9DDDA to #FF00FF in 1 seconds with smooth interval. Keytime describes HH:MM:SS values.

DoubleAnimation used to change values of width/height of the UIElement on specific Duration.

Hide   Copy Code

```
<Border Name="EventTriggerBorder"
    Width="100"
    Height="30"
    Background="Olive"
    BorderBrush="Black"
    BorderThickness="1">
    <Border.Triggers>
        <EventTrigger RoutedEvent="Mouse.MouseEnter">
            <BeginStoryboard>
                <Storyboard>
                <DoubleAnimation AutoReverse="True"
                    Duration="0:0:2"
                    From="1"
                    Storyboard.TargetName="EventTriggerBorder"
                    Storyboard.TargetProperty="(Border.Opacity)"
                    To="0.5" />
                <DoubleAnimation AutoReverse="True"
                    Duration="0:0:5"
                    From="100"
                    Storyboard.TargetName="EventTriggerBorder"
                    Storyboard.TargetProperty="(Border.Width)"
                    To="20" />
               </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Border.Triggers>
</Border>
```

In the above snippet, Border width animated from 100 to 20 over duration of 5 seconds. and immediately animated from 20 to 100, because of **AutoReverse** property. The AutoReverse property will animate object **From**->**To**->**From**. The same way Border Opacity property changing from 1 to 0.5 and vise a versa.

DoubleAnimationUsingKeyFrames: Functions same as DoubleAnimation with addition to KeyFrames Property.

Hide   Shrink ▲   Copy Code

```
<Border Name="EventTriggerBorder"
    Width="100"
    Height="30"
    Background="Olive"
    BorderBrush="Black"
    BorderThickness="1">
    <Border.Triggers>
        <EventTrigger RoutedEvent="Mouse.MouseEnter">
```

```
            <BeginStoryboard>
               <Storyboard>
                <DoubleAnimationUsingKeyFrames Storyboard.TargetName="EventTriggerBorder"
                                        Storyboard.TargetProperty="(FrameworkElement.Width)">
                    <EasingDoubleKeyFrame KeyTime="0:0:0.7" Value="250" />
                </DoubleAnimationUsingKeyFrames>
               </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
        <EventTrigger RoutedEvent="Mouse.MouseLeave">
            <BeginStoryboard>
               <Storyboard>
                <DoubleAnimationUsingKeyFrames Storyboard.TargetName="EventTriggerBorder"
                                       Storyboard.TargetProperty="(FrameworkElement.Width)">
                    <EasingDoubleKeyFrame KeyTime="0:0:0.7" Value="100" />
                </DoubleAnimationUsingKeyFrames>
               </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Border.Triggers>
</Border>
```

Storyboard

Storyboard is used to provide animation to the properties of the UIElement. Storyboard has **TargetName** and **TargetProperty** attached properties for apply animation to the Control (TargetName) and Control Property (TargetProperty).

Hide   Copy Code

```
<Storyboard x:Key="LoadStoryBoard"
    AutoReverse="True"
    RepeatBehavior="Forever">
    <DoubleAnimationUsingKeyFrames Storyboard.TargetName="LoadedBorder"
                Storyboard.TargetProperty="(UIElement.Opacity)">
        <EasingDoubleKeyFrame KeyTime="0:0:0.7" Value="0.4" />
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

In the above snippet, Opacity of the border will change from 1 to 0.4 wihtin 0.7 seconds. **RepeatBehavior**Property ia used to Repeat Animation based on value of the property. To start a Storyboard, you have to call action BeginStoryboard within EventTrigger of the Control. EventTrigger describes perform action when any event executes by the control. like:

Hide   Copy Code

```
<Border.Triggers>
    <EventTrigger RoutedEvent="FrameworkElement.Loaded">
        <BeginStoryboard Storyboard="{StaticResource LoadStoryBoard}" />
    </EventTrigger>
</Border.Triggers>
```

3. DataTrigger

As the name suggest, DataTrigger applies property value to perform action on Data that Binding to the UIElement. DataTrigger allows to set property value when Binding Data matches specified condition. For example:

Hide Copy Code

```
<DataTemplate>
  <Grid Margin="0 5 0 0">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Image x:Name="viewImage"
      Grid.Row="0" Width="100"
      Height="60" HorizontalAlignment="Center"
      Source="{Binding Picture}" Stretch="Fill" />
    <TextBlock x:Name="viewText"
      Grid.Row="1" Margin="0 5 0 0"
      HorizontalAlignment="Center"
      FontWeight="Black" Foreground="Green"
      Text="{Binding Title}" />
  </Grid>
  <DataTemplate.Triggers>
   <DataTrigger Binding="{Binding Path=Picture}" Value="{x:Null}">
    <Setter TargetName="viewImage" Property="Source" Value="/Images/noImage.png" />
    <Setter TargetName="viewText" Property="Text" Value="No Image Available" />
    <Setter TargetName="viewText" Property="Foreground" Value="Red" />
   </DataTrigger>
  </DataTemplate.Triggers>
</DataTemplate>
```

In above example, DataTrigger created on Picture property of Binding data. Setters object of the DataTrigger describes property values to apply when Binding Data match the condition. Picture is **Byte[]** property of the class. If Picture is null then DataTrigger applies on Image to set Image Source to NoImage, otherwise it will set Image Source from Picture. Same as if Picture is null, applies **TextBlock** value to **"No Image Availalbe"** and **Foreground** color to **Red** otherwise sets Default value from Data.

4. MultiDataTrigger

MultiDataTrigger is same as DataTrigger in addition property value applied on multiple condition is matches.

Hide Copy Code

```
<DataTemplate.Triggers>
  <MultiDataTrigger>
    <MultiDataTrigger.Conditions>
      <Condition Binding="{Binding Path=Picture}" Value="{x:Null}" />
      <Condition Binding="{Binding Path=Title}" Value="Waterfall" />
    </MultiDataTrigger.Conditions>
    <MultiDataTrigger.Setters>
     <Setter TargetName="viewImage" Property="Source" Value="/Images/noImage.png"/>
     <Setter TargetName="viewImage" Property="Opacity" Value="0.5" />
     <Setter TargetName="viewText" Property="Background" Value="Brown" />
    </MultiDataTrigger.Setters>
  </MultiDataTrigger>
```

```
</DataTemplate.Triggers>
```

In above code, all collections of setter applies property value when below two conditions are satisfied:

1. Picture = null
2. Title = 'Waterfall'

Otherwise the default property values are applied.

https://www.codeproject.com/Tips/522041/Triggers-in-WPF