# «Триггеры в WPF-приложениях»

В предыдущей лабораторной работе было указано, что в стиле могут быть определены триггеры, предназначенные для изменения значений атрибутов элементов управления при возникновении какихлибо событий или при изменении каких-либо свойств. Триггеры определяются в коллекции Triggers стиля:

<Style ...>

<Style.Triggers>

<!-- Триггеры -->

</Style.Triggers>

</Style>

Существуют следующие виды триггеров:

1. **Простой триггер** (Trigger) срабатывает в случае, когда заданное свойство текущего элемента управления принимает заданное значение. При срабатывании изменяет значения свойств элемента управления или применяет анимацию с использований свойств текущего элемента управления.
2. **Триггер привязки** (DataTrigger) отличается от простого триггера тем, что проверяет связанное свойство другого элемента управления.
3. **Триггер события** (EventTrigger) срабатывает при возникновении в текущем элементе управления заданного события. При срабатывании применяет анимацию с использований свойств текущего элемента управления.
4. **Множественный триггер** (MultiTrigger) отличается наличием нескольких условий. Триггер срабатывает только в том случае, когда все условия выполняются. При срабатывании изменяет значения свойств элемента управления или применяет анимацию с использований свойств текущего элемента управления.
5. **Множественный триггер привязки** (MultiDataTrigger) отличается от множественного триггера тем,

что позволяет проверять связанные свойства других элементов управления.

## Простой триггер (Trigger)

Формат простого триггера:

<Trigger Property="СВОЙСТВО" Value="ЗНАЧЕНИЕ">

<Trigger.Setters>

<!-- Коллекция элементов Setter -->

</Trigger.Setters>

</Trigger>

Свойство Setters можно не указывать:

<Trigger Property="СВОЙСТВО" Value="ЗНАЧЕНИЕ">

<!-- Коллекция элементов Setter -->

</Trigger>

Триггер срабатывает, когда свойство, указанное в атрибуте Property, принимает значение, указанное в атрибуте Value. В триггере нельзя задать сложные зависимости (больше, меньше, вхождения в диапазон и

т.п.). При срабатывании к текущему элементу управления применяются элементы Setter, изменяющие его свойства. Как только действие триггера прекратится (свойство, указанное в атрибуте Property, принимает значение отличное от указанного в атрибуте Value), измененные свойства возвращаются к своим первоначальным значениям.

Примеры часто используемых свойств и их значений:

|  |  |  |
| --- | --- | --- |
| Свойство | Значения | Описание |
| IsVisible | True, False | Является ли элемент управления видимым |
| IsEnabled | True, False | Является ли элемент управления доступным |
| IsFocused | True, False | Имеет ли элемент логический фокус |
| IsMouseOver | True, False | Находится ли курсор над данным элементом |
| IsPressed | True, False | Активизирован ли данный элемент управления |

Пример WPF-приложения, в котором при наведении курсора на кнопку размер шрифта этой кнопки увеличивается.

|  |
| --- |
| **Пример 1 Код XAML** |
| <Window.Resources>  <Style TargetType="Button">  <Style.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Trigger.Setters>  <Setter Property="FontSize" Value="20" />  </Trigger.Setters>  </Trigger>  </Style.Triggers>  </Style>  </Window.Resources>  <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button>Открыть</Button>  <Button>Обработать</Button>  <Button>Сохранить</Button>  <Button>Закрыть</Button>  </StackPanel> |
| **Результат** |
|  |

## Задание 1

Рассмотрите случай, когда для одного и того же элемента управления срабатывают сразу несколько триггеров, устанавливающих для одного и того же свойства различные значения, и определите правило, по которому определяется приоритет применения элементов Setter этих триггеров.

## Триггер привязки (DataTrigger)

Формат простого триггера:

<DataTrigger Binding="{Binding ElementName=ИМЯ\_СВЯЗАННОГО\_ОБЪЕКТА, Path=СВОЙСТВО}" Value="ЗНАЧЕНИЕ">

<DataTrigger.Setters>

<!-- Коллекция элементов Setter -->

</DataTrigger.Setters>

</DataTrigger>

Свойство Setters можно не указывать:

<DataTrigger Binding="{Binding ElementName=ИМЯ\_СВЯЗАННОГО\_ОБЪЕКТА, Path=СВОЙСТВО}" Value="ЗНАЧЕНИЕ">

<!-- Коллекция элементов Setter -->

</DataTrigger>

Триггер срабатывает, когда свойство связанного объекта, указанное в атрибуте Binding, принимает значение, указанное в атрибуте Value. Как только действие триггера прекратится, измененные свойства возвращаются к своим первоначальным значениям.

Пример WPF-приложения, в котором сдвиге ползунка до минимального значения кнопки становятся неактивными:

|  |
| --- |
| **Пример 2 Код XAML** |
| <Window.Resources>  <Style TargetType="Button">  <Style.Triggers>  <DataTrigger Binding="{Binding ElementName=slider1, Path=Value}" Value="0">  <Setter Property="IsEnabled" Value="False"></Setter>  </DataTrigger>  </Style.Triggers>  </Style>  </Window.Resources>  <StackPanel Orientation="Horizontal" VerticalAlignment="Top">  <Button>Открыть</Button>  <Button>Обработать</Button>  <Slider Height="23" x:Name="slider1" Width="100" />  </StackPanel> |
| **Результат** |
|  |

## Множественный триггер (MultiTrigger)

Формат множественного триггера:

<MultiTrigger>

<MultiTrigger.Conditions> <Condition .../>

...

<Condition .../>

</MultiTrigger.Conditions>

<MultiTrigger.Setters>

<!-- Коллекция элементов Setter -->

</MultiTrigger.Setters>

</MultiTrigger>

Свойство Setters можно не указывать:

<MultiTrigger>

<MultiTrigger.Conditions> <Condition .../>

...

<Condition .../>

</MultiTrigger.Conditions>

<!-- Коллекция элементов Setter -->

</MultiTrigger>

Триггер срабатывает, когда все свойства, указанные в элементах Condition, принимают соответствующие значения. Как только действие триггера прекратится, измененные свойства возвращаются к своим первоначальным значениям.

Элемент условия Condition может проверять значение свойства текущего элемента управления:

<Condition Property="СВОЙСТВО" Value="ЗНАЧЕНИЕ" />

## Множественный триггер привязки (MultiDataTrigger)

Множественный триггер привязки отличается от триггера привязки тем, что элемент условия Condition может проверять как значение свойства текущего элемента управления:

<Condition Property="СВОЙСТВО" Value="ЗНАЧЕНИЕ" /> так и значение связанного свойства другого элемента управления:

<Condition Binding="{Binding ElementName=ИМЯ\_СВЯЗАННОГО\_ОБЪЕКТА, Path=ИМЯ\_СВОЙСТВА}"

Value="ЗНАЧЕНИЕ"/>

## Задание 2

Разработайте WPF-приложение с двумя многострочными текстовыми полями, кнопками «Открыть», «Очистить», «Закрыть» и выпадающим списком для задания внешнего вида текстовых полей. Задайте для текстовых полей одинаковый градиентный фон. Кнопка «Закрыть» должна быть доступна только в том случае, если в обоих текстовых полях нет текста. Задайте для кнопок различный внешний вид при наведении курсора и при нажатии на них. Внешний вид текстовых полей (тип шрифта, размер шрифта, цвет шрифта) должен меняться в зависимости от значения, выбранного в выпадающем списке.