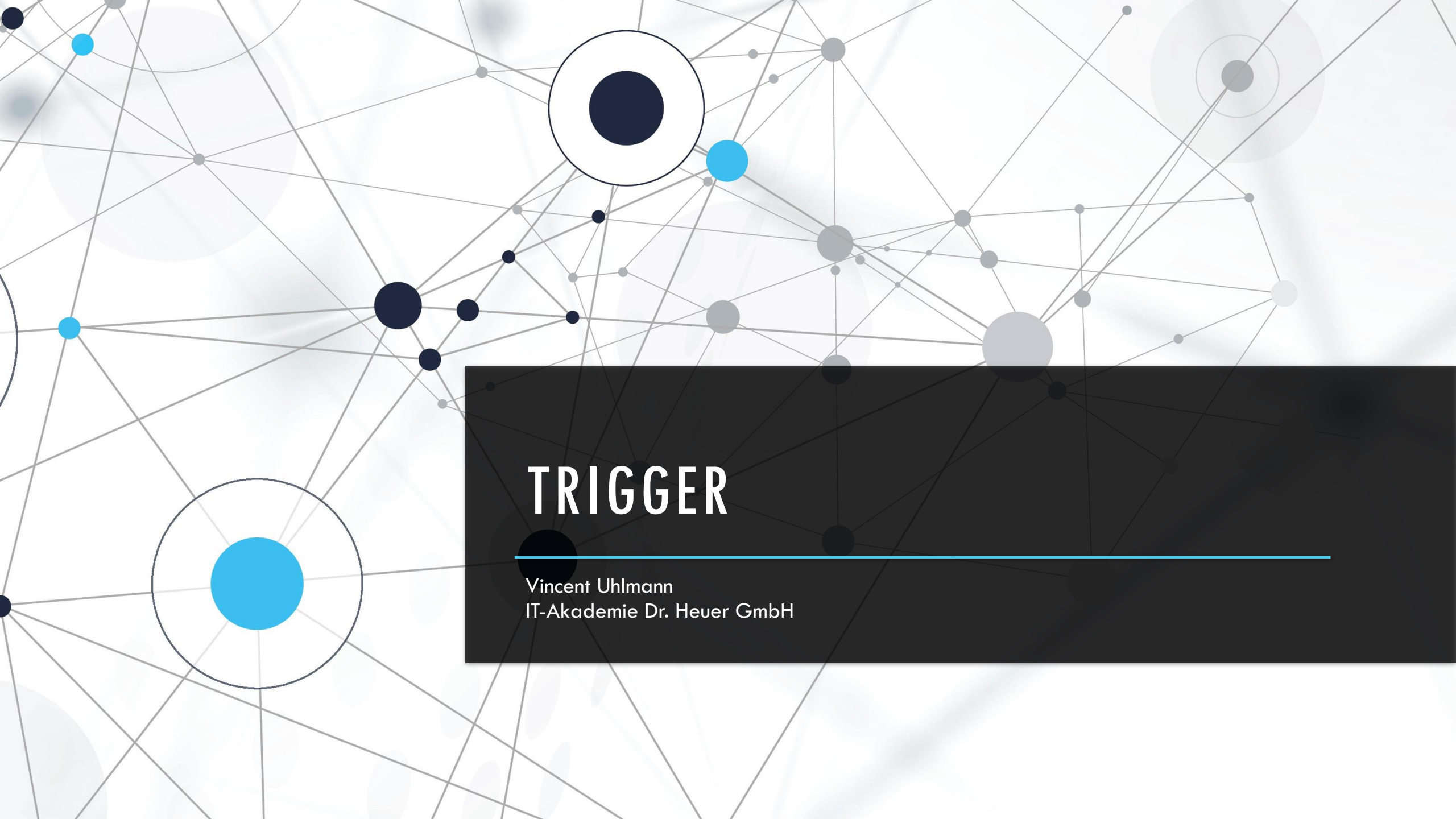

TRIGGER

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

TRIGGER

- Trigger sind eine Methode in XAML, um UI-Änderungen basierend auf Ereignissen oder Zustandsänderungen zu steuern
- Es gibt verschiedene Trigger für unterschiedliche Anwendungsfälle

- Property Trigger
- Data Trigger
- Event Trigger
- Multi Trigger
- …

PROPERTY TRIGGER

- Reagiert auf Änderungen von Eigenschaftswerten von Elementen

```
<Entry Placeholder="Enter name">
    <Entry.Triggers>
        <Trigger TargetType="Entry" Property="IsFocused" Value="True">
            <Setter Property="BackgroundColor" Value="Yellow" />
        </Trigger>
    </Entry.Triggers>
</Entry>
```

DATA TRIGGER

- Ermöglicht das Setzen von Eigenschaften basierend auf Datenbedingungen

```
<Entry
    x:Name="entry"
    Placeholder="Enter text"
    Text="" />
<Button Text="Save">
    <Button.Triggers>
        <DataTrigger
            Binding="{Binding Source={x:Reference entry}, Path=Text.Length}"
            TargetType="Button"
            Value="0">
            <Setter Property="BackgroundColor" Value="Green" />
        </DataTrigger>
    </Button.Triggers>
</Button>
```

MULTI TRIGGER

- Führt eine oder mehrere Aktionen aus, wenn eine oder mehrere Bedingungen erfüllt sind
- Funktioniert sowohl mit BindingConditions als auch mit PropertyConditions

```
<Entry x:Name="email" Text="" />
<Entry x:Name="phone" Text="" />
<Button Text="Save">
    <Button.Triggers>
        <MultiTrigger TargetType="Button">
            <MultiTrigger.Conditions>
                <BindingCondition Binding="{Binding Source={x:Reference email}, Path=Text.Length}" Value="0" />
                <BindingCondition Binding="{Binding Source={x:Reference phone}, Path=Text.Length}" Value="0" />
            </MultiTrigger.Conditions>
            <Setter Property="BackgroundColor" Value="Green" />
        </MultiTrigger>
    </Button.Triggers>
</Button>
```

EVENT TRIGGER

- Ein EventTrigger stellt einen Auslöser dar, der als Reaktion auf ein Ereignis eine Reihe von Aktionen ausführt
- Im Gegensatz zu Trigger hat EventTrigger kein Konzept der Beendigung des Zustands, so dass die Aktionen nicht rückgängig gemacht werden, sobald die Bedingung, die das Ereignis ausgelöst hat, nicht mehr erfüllt ist
- Für einen EventTrigger muss ein Event festgelegt werden
- In diesem Beispiel gibt es keine Setter-Elemente. Stattdessen gibt es ein OnClickedTriggerAction-Objekt

```
<Button Clicked="Button_Clicked" Text="Save">
    <Button.Triggers>
        <EventTrigger Event="Clicked">
            <local:OnClickedTriggerAction />
        </EventTrigger>
    </Button.Triggers>
</Button>
```

EVENT TRIGGER

- Die Implementierung einer Triggeraktion muss:
- Die generische Klasse TriggerAction<T> implementieren, wobei der generische Parameter dem Typ des Steuerelements entspricht, auf das der Trigger angewendet werden soll
- Die Invoke-Methode überschreiben, die aufgerufen wird, wenn das Triggerereignis eintritt
- Stellen Sie optional Eigenschaften zur Verfügung, die in XAML festgelegt werden können, wenn der Trigger deklariert wird

```
public class OnClickedTriggerAction : TriggerAction<Button>
{
    protected override void Invoke(Button sender)
    {
        if (sender.BackgroundColor == Colors.Green)
            sender.BackgroundColor = Colors.Red;
        else
            sender.BackgroundColor = Colors.Green;
    }
}
```

VISUAL STATE TRIGGER

- Visual State Manager verwaltet Zustände von UI-Komponenten und deren Übergänge
- Ermöglicht das Anpassen der UI basierend auf verschiedenen Zuständen wie Focused, Disabled, Normal

- **Visual States**: Definieren spezifische Looks oder Verhaltensweisen für Steuerelemente unter bestimmten Bedingungen
- **Visual State Groups**: Gruppieren ähnliche Visual States. Jedes Element kann mehrere Visual State Groups haben

VISUAL STATE TRIGGER

```
<Entry FontSize="18">
    <VisualStateManager.VisualStateGroups>
        <VisualStateGroupList>
            <VisualStateGroup x:Name="CommonStates">
                <VisualState x:Name="Normal">
                    <VisualState.Setters>
                        <Setter Property="BackgroundColor" Value="Lime" />
                    </VisualState.Setters>
                </VisualState>

                <VisualState x:Name="Focused">
                    <VisualState.Setters>
                        <Setter Property="FontSize" Value="36" />
                    </VisualState.Setters>
                </VisualState>
            </VisualStateGroup>
        </VisualStateGroupList>
    </VisualStateManager.VisualStateGroups>
</Entry>
```