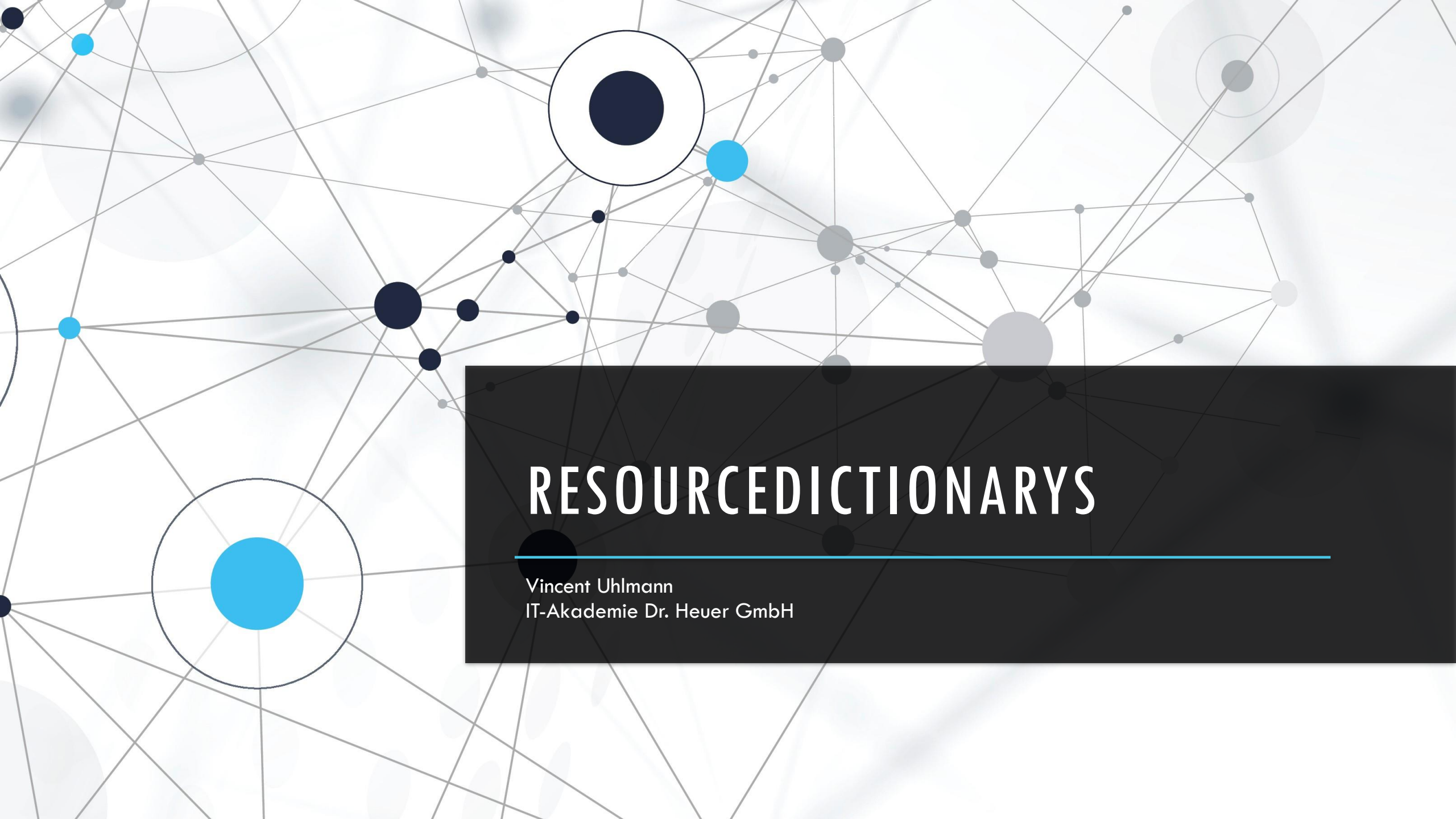

RESOURCEDICTIONARYS

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

RESOURCEDICTIONARIES

- Ein ResourceDictionary ist ein Schlüssel-Wert-Speicher für Ressourcen in XAML
- Sammlungen von wiederverwendbaren Ressourcen wie Farben, Stile usw.
- Ermöglicht zentrale Änderungen, die sich durch die ganze App ziehen

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xaml-comp compile="true" ?>
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml">
</ResourceDictionary>
```

ARTEN VON RESSOURCEN

- **Farben:** Definiert als Hex-Code oder RGB-Werte
- **Stile:** Sammlungen von Eigenschaften, die auf bestimmte Steuerelemente angewendet werden
- **Daten:** Werte wie Strings oder Zahlen
- **Templates:** XAML-Vorlagen für komplexes UI-Design
- ...

```
<ResourceDictionary>
    <Color x:Key="MainColor">#562234</Color>
</ResourceDictionary>
```

ERSTELLEN VON STYLES

- Styles sind XAML-Objekte, die Eigenschaften von UI-Elementen definieren
- Beispiele für Style-Eigenschaften: BackgroundColor, FontSize, Margin
- Ein Style kann einem Element direkt oder über einen Schlüssel zugewiesen werden
- Beispiel eines Styles für einen Button in XAML

```
<Style x:Key="PrimaryButton" TargetType="Button">
    <Setter Property="TextColor" Value="White" />
    <Setter Property="BackgroundColor" Value="#007bff" />
    <Setter Property="FontAttributes" Value="Bold" />
</Style>
```

VERWENDUNG VON STYLES

- Styles werden mit dem StaticResource oder DynamicResource Markup verwendet
- Beispiel für die Anwendung eines Styles auf einen Button

```
<Button Text="Klicken Sie hier" Style="{StaticResource PrimaryButton}" />
```

RESSOURCEN: SCOPE UND ZUGRIFF

- Jedes von VisualElement abgeleitete Objekt hat eine Resources-Eigenschaft, die ein ResourceDictionary enthält
- Die Application-Klasse verfügt ebenfalls über eine Resources-Eigenschaft für globale Ressourcen
- Diese verschiedenen Ressourceneigenschaften ermöglichen es uns, den Zugriff/Umfang der einzelnen Stile zu definieren

ANSICHTSEBENE (VIEW LEVEL)

- Ressourcen, die einem spezifischen Steuerelement wie Button oder Label zugewiesen sind, sind nur dort anwendbar

```
<Button Text="Klicke mich">
    <Button.Resources>
        <ResourceDictionary>
            <Style x:Key="ButtonStyle" TargetType="Button">
                <Setter Property="BackgroundColor" Value="Pink" />
                <Setter Property="TextColor" Value="White" />
            </Style>
        </ResourceDictionary>
    </Button.Resources>
    <Button.Style>
        <StaticResource Key="ButtonStyle" />
    </Button.Style>
</Button>
```

LAYOUTEBENE (LAYOUT LEVEL)

- Ressourcen, die einem Layout wie StackLayout oder Grid zugewiesen sind, sind auf das Layout und dessen Kinder anwendbar

```
<StackLayout>
    <StackLayout.Resources>
        <ResourceDictionary>
            <Color x:Key="LayoutBackgroundColor">#eeeeee</Color>
        </ResourceDictionary>
    </StackLayout.Resources>
    <Button BackgroundColor="{StaticResource LayoutBackgroundColor}" />
</StackLayout>
```

SEITENEBENE (PAGE LEVEL)

- Ressourcen auf der Seitenebene sind auf die Seite und alle ihre Kinder anwendbar

```
<ContentPage
    x:Class="Test.MainPage"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml">
    <ContentPage.Resources>
        <ResourceDictionary>
            <Style x:Key="PageLabelsStyle" TargetType="Label">
                <Setter Property="TextColor" Value="Blue" />
            </Style>
        </ResourceDictionary>
    </ContentPage.Resources>
    <Label Style="{StaticResource PageLabelsStyle}" Text="Willkommen" />
</ContentPage>
```

ANWENDUNGSEBENE (APPLICATION LEVEL)

- Globale Ressourcen in der App.xaml sind in der gesamten App verwendbar

```
<Application.Resources>
    <ResourceDictionary>
        <Color x:Key="AppPrimaryColor">#ff6600</Color>
        <Style x:Key="GlobalButtonStyle" TargetType="Button">
            <Setter Property="TextColor" Value="{StaticResource AppPrimaryColor}" />
        </Style>
    </ResourceDictionary>
</Application.Resources>
```

MERGEDDICTIONARIES

- Erlaubt das Bündeln und Teilen von Ressourcen über mehrere XAML-Dateien hinweg
- Ermöglicht das Laden von Ressourcen aus verschiedenen Quellen zur Laufzeit
- Modularisiert die XAML-Ressourcen-Dateien

```
<ResourceDictionary>
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="Resources/Styles/Colors.xaml" />
        <ResourceDictionary Source="Resources/Styles/Styles.xaml" />
    </ResourceDictionary.MergedDictionaries>
</ResourceDictionary>
```

IMPLIZITE STILE

- Implizite Stile benötigen keinen x:Key und wenden sich an alle Steuerelemente des angegebenen Typs im Scope

```
<StackLayout>
    <StackLayout.Resources>
        <ResourceDictionary>
            <Style TargetType="Button">
                <Setter Property="BackgroundColor" Value="Green" />
            </Style>
        </ResourceDictionary>
    </StackLayout.Resources>
    <Button />
    <Button />
    <Button />
    <Button />
</StackLayout>
```

STILVERERBUNG

- Ermöglicht, dass ein Stil von einem anderen Stil Eigenschaften erbt
- Verhindert Redundanz und erleichtert Änderungen über mehrere Stile hinweg
- Der abgeleitete Stil erbt alle Setter und kann zusätzliche Setter definieren oder bestehende überschreiben

```
<Style x:Key="BaseButtonStyle" TargetType="Button">
    <Setter Property="BackgroundColor" Value="LightGray"/>
    <Setter Property="TextColor" Value="Black"/>
    <Setter Property="Padding" Value="10"/>
</Style>

<Style x:Key="ConfirmButtonStyle" TargetType="Button" BasedOn="{StaticResource BaseButtonStyle}">
    <Setter Property="BackgroundColor" Value="Green"/>
    <Setter Property="TextColor" Value="White"/>
</Style>
```