WPF Intro and XAML

Layout definition in Code

- Coming from Windows Forms: definition of layout in code (generated by designer)
- WPF: can be used like Windows Forms - although that's not the preferred way

Sample: Currency Calculator, defined in Code

- New project (CurrencyCalculator.Wpf.Code, Solution: CurrencyCalculator)
- Delete App.xaml und MainWindow.xaml
- Add CurrencyCalculatorWindow.cs

```
using System;
using System.Windows;

namespace WpfCurrencyCalculator_Code
{
    public class CurrencyCalculatorWindow : Window
    {
        [STAThread]
        public static void Main(string[] args)
        {
            Window window = new CurrencyCalculatorWindow();
            window.Show();
        }
    }
}
```

- Window closes immediately -> Application necessary

```
Application app = new Application();
Window window = new CurrencyCalculatorWindow();
window.Show();
app.Run();
```

- Experiment with some properties

```
Window window2 = new CurrencyCalculatorWindow();
window2.Title += "2";
window2.Show();

app.ShutdownMode = ShutdownMode.OnMainWindowClose;
// app.ShutdownMode = ShutdownMode.OnLastWindowClose;
```

- Show Content-Property of Window

  ```
  public CurrencyCalculatorWindow()
  {
      Content = "Hello World!";
  }
  ```

- Create controls

  ```
  private TextBox txtLeftValue;
  private TextBox txtRightValue;
  private ComboBox cmbLeftCurrency;
  private ComboBox cmbRightCurrency;

  public CurrencyCalculatorWindow()
  {
      txtLeftValue = new TextBox();
      txtRightValue = new TextBox();
      cmbLeftCurrency = new ComboBox();
      cmbRightCurrency = new ComboBox();
  }
  ```

- Use Layoutmanager to align controls

  ```
  StackPanel panel = new StackPanel();

  panel.Children.Add(txtLeftValue);
  panel.Children.Add(cmbLeftCurrency);
  panel.Children.Add(txtRightValue);
  panel.Children.Add(cmbRightCurrency);

  this.Content = panel;
  this.Title = "WPF Currency Converter (Code)";
  ```

- Change Orientation to Horizontal

- Discuss "Problem" of fluid and vector based layout

  - Set Width of TextBoxes (e.g. 80)
  - Add margins to StackPanel and controls

  ```
  this.SizeToContent = SizeToContent.WidthAndHeight;
  this.ResizeMode = ResizeMode.NoResize;
  ```

- Add CurrencyCalculator.BL to solution

  ```
  private ICurrencyCalculator calculator;

  this.calculator = CurrencyCalculatorFactory.GetCalculator();
  ```

```
foreach (CurrencyData currency in calculator.GetCurrencyData())
{
    cmbLeftCurrency.Items.Add(currency);
    cmbRightCurrency.Items.Add(currency);
}

cmbLeftCurrency.SelectedIndex = cmbRightCurrency.SelectedIndex = 0;
```

- Register for **SelectionChanged** events

```
enum Conversion { LeftToRight, RightToLeft }

private void OnSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (sender == cmbLeftCurrency)
        Convert(Conversion.LeftToRight);
    else
        Convert(Conversion.RightToLeft);
}

private void Convert(Conversion conversion)
{
    if (cmbLeftCurrency.SelectedItem == null || cmbRightCurrency.
SelectedItem == null)
        return;

    string leftCurrency = ((CurrencyData)cmbLeftCurrency.SelectedItem).
Symbol;
    string rightCurrency = ((CurrencyData)cmbRightCurrency.SelectedItem).
Symbol;

    double input;
    if (conversion == Conversion.LeftToRight)
    {
        if (double.TryParse(txtLeftValue.Text, out input))
            txtRightValue.Text =
calculator.Convert(input, leftCurrency, rightCurrency).ToString("F2");
    }
    else
    {
        if (double.TryParse(txtRightValue.Text, out input))
            txtLeftValue.Text =
calculator.Convert(input, rightCurrency, leftCurrency).ToString("F2");
    }
}
```

- Add EventHandler for Textboxes:

```
txtLeftValue.TextChanged += OnTextChanged;
txtRightValue.TextChanged += OnTextChanged;
```

```
private void OnTextChanged(object sender, TextChangedEventArgs e)
{
    if (sender == txtLeftValue && txtLeftValue.IsFocused)
        Convert(Conversion.LeftToRight);
    else if (sender == txtRightValue && txtRightValue.IsFocused)
        Convert(Conversion.RightToLeft);
}
```

Layout definition in XAML

- eXtensible Markup Language
- Discuss the approach and advantages of XAML

Sample: Currency Calculator, defined in XAML

- New WPF Application: CurrencyCalculator.Wpf.Xaml
- Rename MainWindow to CurrencyCalculatorWindow
- Discuss Code-behind file

```
<Window x:Class="WpfCurrencyCalculator_Xaml.CurrencyCalculatorWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="WPF Currency Converter (XAML)" SizeToContent="WidthAndHeight"
        ResizeMode="NoResize">

    <StackPanel Orientation="Horizontal" Margin="10">
        <TextBox Name="txtLeftValue" Width="80"/>
        <ComboBox Name="cmbLeftCurrency" Margin="5,0"/>
        <TextBox Name="txtRightValue" Width="80"/>
        <ComboBox Name="cmbRightCurrency" Margin="5,0,0,0"/>
    </StackPanel>
</Window>
```

- Adapt StartupUri and startup project in Visual Studio
- Copy source code from Code solution (event handlers, etc.)
- Show different approach for adding Items

```
//Version 2:
IEnumerable<CurrencyData> currencies = calculator.GetCurrencyData();
cmbLeftCurrency.ItemsSource = cmbRightCurrency.ItemsSource = currencies;

cmbLeftCurrency.SelectedItem = currencies.First(c => c.Symbol == "USD");
cmbRightCurrency.SelectedItem = currencies.First(c => c.Symbol == "EUR");
```

- Replace StackPanel with Grid

    - Show width/height options

Visual/Logical tree, Data Templates

- Discuss differences between visual and logical tree
- Separation between functionality and design in WPF controls

Create a Data Template

- Copy images folder from Templates

```
<ComboBox.ItemTemplate>
    <DataTemplate>
        <StackPanel Orientation="Horizontal" Margin="2">
            <Image Width="25" Height="15" Source="Images/USD.gif"/>
            <TextBlock Text="{Binding Path=Symbol}"
VerticalAlignment="Center" Margin="5,0,0,0" />
        </StackPanel>
    </DataTemplate>
</ComboBox.ItemTemplate>
```

- Explain DataContext, basic Databinding features

Converters

- Create a converter to build the image url dynamically
- Explain XAML-namespaces
- Explain Resources
- Use newly created converter in DataTemplate

Resources

- Reuse style, template definitions by making them to a named resource on page or app level
- Move DataTemplate to Resources
- Use it in both ComboBoxes

Graphics in WPF

- Show some controls (Rectangle, Elipse, etc.)
- Optional: create a history chart by using `MonthlyRatesOfExchange`method, data binding and rectangles

```
<ItemsControl Name="rates" >
    <ItemsControl.LayoutTransform>
        <RotateTransform Angle="270" />
    </ItemsControl.LayoutTransform>
    <ItemsControl.ItemTemplate>
        <DataTemplate>

            <StackPanel Orientation="Horizontal">
                <Rectangle Width="{Binding AverageRate, Converter={StaticResource ChartWidth
                            Height="20" Fill="Green"
                            HorizontalAlignment="Left"/>

                <TextBlock Text="{Binding AverageRate, StringFormat=N2}" Margin="10,0,0,0" /
            </StackPanel>

        </DataTemplate>
    </ItemsControl.ItemTemplate>
</ItemsControl>
```

```
public class ChartWidthConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo cultu
    {
        return (double)value * 100;
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo c
    {
        throw new NotImplementedException();
    }
}
```

Multilingual interfaces

- Create Resources (*.resx files)
- Change custom tool of Resource from "ResXFileCodeGenerator" to "**PublicResXFileCodeGenerator**"
- Bind to strings in XAML