Inhoud

[RadioButton 3](#_Toc31203230)

[Groupbox 3](#_Toc31203231)

[Stackpanel 4](#_Toc31203232)

[Text inputs 4](#_Toc31203233)

[Spell check 4](#_Toc31203234)

[Event routing 5](#_Toc31203235)

[Style 5](#_Toc31203236)

[Templates 6](#_Toc31203237)

[Control Templates 6](#_Toc31203238)

[Data Templates 6](#_Toc31203239)

[Animations 7](#_Toc31203240)

[Triggers 7](#_Toc31203241)

[GirdSplitter 7](#_Toc31203242)

[Panels 8](#_Toc31203243)

[Stackpanel 8](#_Toc31203244)

[Dockpanel 8](#_Toc31203245)

[WrapPanel 9](#_Toc31203246)

[Scrollviewer 9](#_Toc31203247)

[ViewBox 9](#_Toc31203248)

[Databinding 9](#_Toc31203249)

[Adding a new object: 10](#_Toc31203250)

[INotifyPropertyChanged 11](#_Toc31203251)

[ObservableCollection<T> 12](#_Toc31203252)

[Important properties on ItemsControl 12](#_Toc31203253)

[Databinding EXAMPLE 13](#_Toc31203254)

[Binding to an element 13](#_Toc31203255)

[Converter 14](#_Toc31203256)

[EXAMPLE 14](#_Toc31203257)

# RadioButton

<StackPanel Orientation="Vertical">

<RadioButton Content="A" GroupName="A"></RadioButton>

<RadioButton Content="B" GroupName="B"></RadioButton>

<RadioButton Content="C" GroupName="C"></RadioButton>

</StackPanel>

<StackPanel Margin="50, 0" Orientation="Vertical">

<RadioButton Content="A" GroupName="A"></RadioButton>

<RadioButton Content="B" GroupName="B"></RadioButton>

<RadioButton Content="C" GroupName="C"></RadioButton>

</StackPanel>

Verdeel groepen met GroupName=””

Result:



# Groupbox

<GroupBox Width="175" Height="100">

<GroupBox.Header>

<StackPanel Orientation="Horizontal">

<TextBlock Text="titel met een " />

<Ellipse Fill="Red" Width="15" Height="15"/>

<TextBlock Text=" rode bol"></TextBlock>

</StackPanel>

</GroupBox.Header>

<StackPanel Orientation="Vertical">

<RadioButton Content="A" GroupName="A"></RadioButton>

<RadioButton Content="B" GroupName="B"></RadioButton>

<RadioButton Content="C" GroupName="C"></RadioButton>

</StackPanel>

</GroupBox>

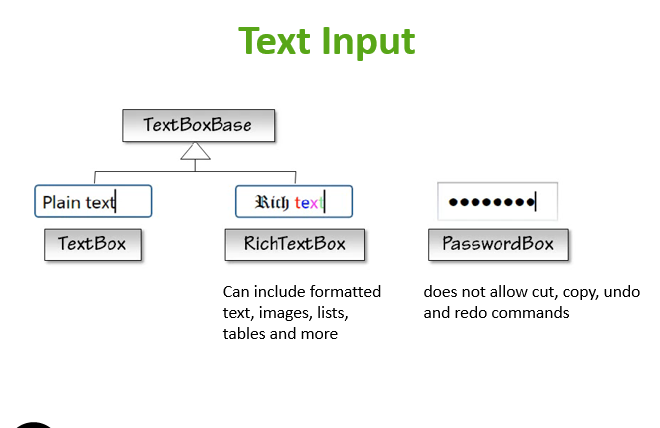
Result:



# Stackpanel

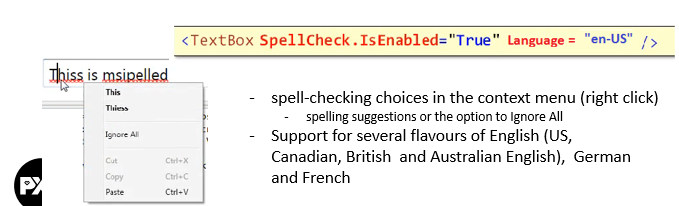
Als ge meerdere dingen bij elkaar wilt groeperen, dan moet ge gebruik maken van een stackpanel. Kijk voorgaande voorbeelden.

# Text inputs

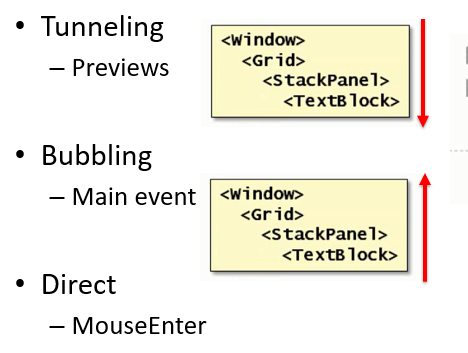


## Spell check

<TextBox SpellCheck.IsEnabled="True" Language="en-US"></TextBox>



# Event routing



There are a couple of routine patterns, tunneling and bubbling:

A tunneling event starts from the very root of the visual tree and works its way through the tree to the element of which the event is already targeted.   
For example, a tunneling event representing a mouse button press would first be offered to the window and then to the child of the window and so on until it reaches the target element.

Conversely a bubbling event starts from the target and then bubbles up the tree until it finds an element that wants to handle the event.

# Style

<Button Content="Hallo" Width="150" >

<Button.Style>

<Style x:Uid="buttonStyle" TargetType="Button">

<Setter Property="Opacity" Value="0.5"/>

<Setter Property="Height" Value="50"/>

<Setter Property="Background" Value="AliceBlue"/>

</Style>

</Button.Style>

</Button>

OF als ge de stijl wilt toepassen op meerdere elementen

<Window.Resources>

<Style x:Key="myButtonStyle" TargetType="Button">

<Setter Property="Height" Value="50"/>

<Setter Property="Width" Value="250"/>

<Setter Property="Background" Value="AliceBlue"/>

</Style>

</Window.Resources>

<Grid>

<Button Style="{StaticResource myButtonStyle}" Content="Hallo"/>

<Button Style="{StaticResource myButtonStyle}" Content="Hallo2"/>

</Grid>

# Templates

## Control Templates

Content template genoemd in de slides? Typo?

<Window.Resources>

<ControlTemplate x:Key="myButtonTemplate" TargetType="Button">

<Border>

<Border.Background>

<SolidColorBrush Color="Black"></SolidColorBrush>

</Border.Background>

<Grid Background="LightBlue" Margin="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" Margin="4,5,4,4"/>

</Grid>

</Border>

</ControlTemplate>

</Window.Resources>

<Grid>

<Button Width="100" Height="50" Template="{StaticResource myButtonTemplate}">Click me</Button>

</Grid>

Result:



## Data Templates

# Animations

## Triggers

<Grid>

<Rectangle x:Name="myRectangle" Width="100" Height="100">

<Rectangle.Triggers>

<EventTrigger RoutedEvent="Rectangle.Loaded">

<BeginStoryboard>

<Storyboard>

<ColorAnimation

Storyboard.TargetName="rectangleBrush"

Storyboard.TargetProperty="Color"

From="Blue" To="Red" Duration="0:0:2"

AutoReverse="True" RepeatBehavior="Forever"/>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

</Rectangle.Triggers>

<Rectangle.Fill>

<SolidColorBrush x:Name="rectangleBrush" Color="Red"></SolidColorBrush>

</Rectangle.Fill>

</Rectangle>

</Grid>

Result:

Knop die om de 2 seconden van blauw naar rood verandert in overgangen.

# GridSplitter

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<GridSplitter Grid.Column="1"

Grid.RowSpan="3"

HorizontalAlignment="Left"

Background="Black"

Width="5"/>

<Button Content="OK" Grid.Column="0" Grid.Row="0"/>

<Button Content="OK" Grid.Column="0" Grid.Row="1" Height="20" Margin="30, 30"/>

<Button Content="OK" Grid.Column="0" Grid.Row="2"/>

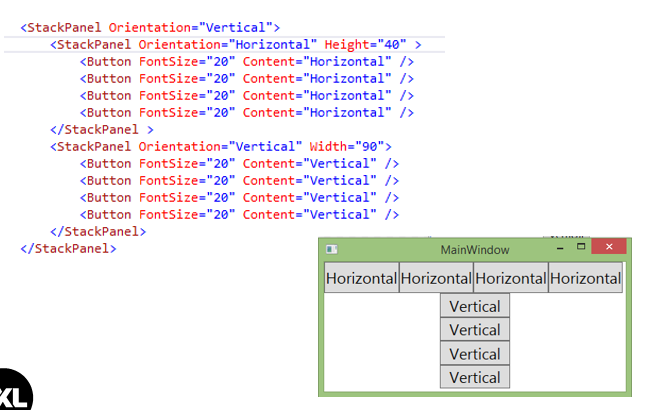
</Grid>

# Panels

## Stackpanel

Stacks child elements below or beside each other

Property: Orientation (Vertical or Horizontal)



## Dockpanel

<Grid>

<DockPanel>

<Button Content="Top" DockPanel.Dock="Top"/>

<Button Content="Bottom" DockPanel.Dock="Bottom"/>

<Button Content="Left" DockPanel.Dock="Left"/>

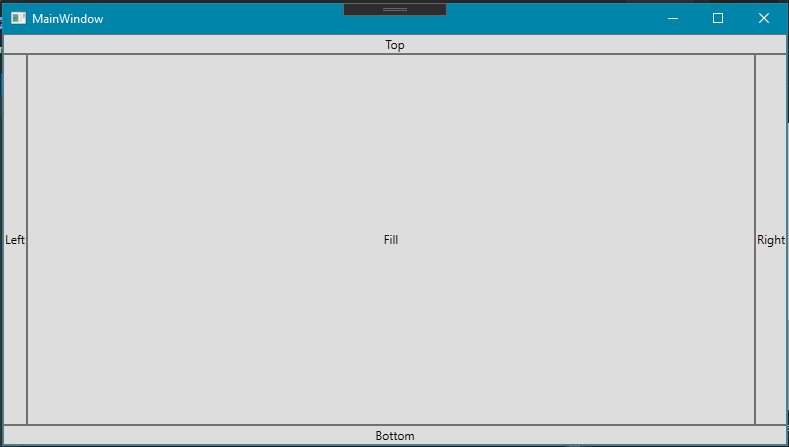
<Button Content="Right" DockPanel.Dock="Right"/>

<Button Content="Fill"/>

</DockPanel>

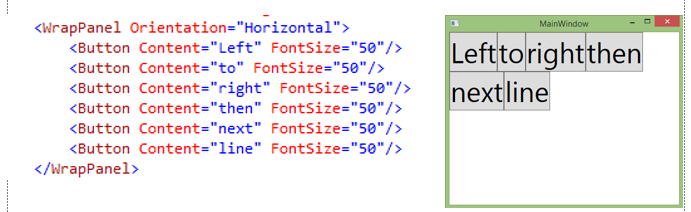
</Grid>

Result:



## WrapPanel

Wraps the elements to new lines (horizontal orientation) / new columns (vertical orientation) if no space is left in the width / height of the container



# Scrollviewer

Scrollviewer zet scrollbars

<Grid>

<ScrollViewer HorizontalScrollBarVisibility="Visible" >

<Canvas Width="8000" Height="4000" Background="Gray"></Canvas>

</ScrollViewer>

</Grid>

# ViewBox

Viewbox scales the content with the available space.

<Grid>

<Viewbox>

<Canvas Width="8000" Height="4000" Background="Gray"></Canvas>

</Viewbox>

</Grid>

# Databinding

Code behind:

public MainWindow()

{

InitializeComponent();

Person = new Person();

person.Age = 15;

person.Balance = 500.25;

person.Birthplace = "Sint-Truiden";

person.Name = "Jef";

DataContext = person;

}

XAML:

<Grid>

<TextBlock Text="{Binding Name}" VerticalAlignment="Top"/>

<TextBlock Text="{Binding Age}" VerticalAlignment="Top"/>

<TextBlock Text="{Binding Balance}" VerticalAlignment="Top"/>

<TextBlock Text="{Binding Birthplace}" VerticalAlignment="Top"/>

</Grid>

## Adding a new object:

Game newGame = NewGameGroupBox.DataContext as Game;

**One Way Binding:** source -> target

**INofifyPropertyChanged interface:** underlying data changes -> view is updated

**Two Way Binding:** user can modify data in UI -> data updated in the source

**Element Binding:** bind to another element on the page

**DataContext:** datasource itself, object whose properties you are binding from

**List Binding:** Bind collections of data to list boxes, combo boxes,…

**Data Templates:** tell the list what part of the data to display by using data templates

**Data Conversion:** converts the data at the source to the type expected at the target

**Data Validation:** Ensuring that the data is valid

## INotifyPropertyChanged

public class Person : INotifyPropertyChanged

{

private string \_name;

private int \_age;

private string \_birthplace;

private double \_balance;

public string Name

{

get => \_name;

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

public int Age { get => \_age; set => \_age = value; }

public string Birthplace { get => \_birthplace; set => \_birthplace = value; }

public double Balance

{

get => \_balance;

set

{

\_balance = value;

//nameof() gives the name of the variable

OnPropertyChanged(nameof(Balance));

}

}

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged([CallerMemberName] string caller = "")

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(caller));

}

}

}

<Grid>

<TextBlock Text="{Binding Name, Mode=TwoWay}" VerticalAlignment="Top"/>

<TextBlock Text="{Binding Age}" VerticalAlignment="Top"/>

<TextBlock Text="{Binding Balance, Mode=TwoWay}" VerticalAlignment="Top"/>

<TextBlock Text="{Binding Birthplace}" VerticalAlignment="Top"/>

</Grid>

OneWay bindings only propagate changes in the source value to the target (assuming the source implements INotifyPropertyChanged)

TwoWay binding propagates changes in both directions, ensuring that the two values are always synchronized.

## ObservableCollection<T>

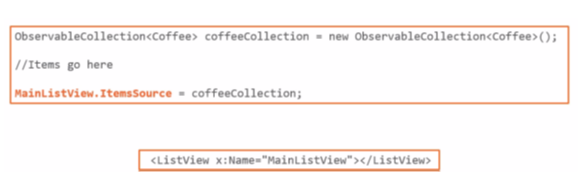
updates the collection, resulting in updates in the items control

From list to ObservableCollection:

gamesOC = new ObservableCollection<Game>(GetDummyGames());

## Important properties on ItemsControl

* ItemSource
  1. Collection of data objects to display in each item
  2. Default= the result of the *ToString* method of the data object is displayed in each item
* DisplayMemberPath
  1. Property of the data object to display in each item
* ItemTemplate
  1. Block of XAML (DataTemplate) to display in each item
  2. DataContext of the item is the data object
* ItemsPanel
  1. Layout container for all the items



## Databinding EXAMPLE

<Grid>

<StackPanel>

<ComboBox Name="personsComboBox" FontSize="16" Height="50" Width="525" SelectedIndex="0">

<ComboBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Margin="2">

<TextBlock FontWeight="Bold" Text="Name: " Margin="2"/>

<TextBlock Text="{Binding Name, Mode=TwoWay}" Margin="2"/>

<TextBlock FontWeight="Bold" Text="Age: " Margin="2"/>

<TextBlock Text="{Binding Age}" Margin="2"/>

<TextBlock FontWeight="Bold" Text="Birthplace: " Margin="2"/>

<TextBlock Text="{Binding Birthplace}" Margin="2"/>

<TextBlock FontWeight="Bold" Text="Balance: " Margin="2"/>

<TextBlock Text="{Binding Balance, Mode=TwoWay}" Margin="2"/>

</StackPanel>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

</StackPanel>

</Grid>

public MainWindow()

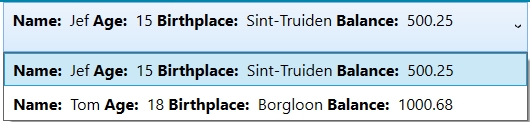
{

InitializeComponent();

PersonRepository \_personRepository = new PersonRepository();

personsComboBox.ItemsSource = \_personRepository.getAll();

}



## Binding to an element

<Grid>

<StackPanel Orientation="Horizontal">

<Slider x:Name="mySlider" Minimum="0" Maximum="100" Width="300"/>

<TextBlock Margin="5" Text="{Binding Path=Value, ElementName=mySlider}"/>

</StackPanel>

</Grid>

# Converter

Bestaat uit 2 methoden:

Convert: Source -> Target

ConvertBack: wordt alleen gebruik in TwoWay binding. Target -> Source

## EXAMPLE

Guts,Chapter4,Oef2

Converters(map) -> RatingConverter.cs(class)

<Window.Resources>

<c:RatingConverter x:Key="RatingConverter">

</c:RatingConverter>

</Window.Resources>

De property waarop de converter toegepast moet worden:

<TextBox Text="{Binding Path=Rating, Converter={StaticResource RatingConverter}, Mode=TwoWay}" x:Name="ratingTextBox"/>

public class RatingConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return (double)value \* 10;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return (int) value / 10;

}

}