

# WPF & Silverlight

Daniel Beck

# WPF

Windows Presentation Foundation

# Entstehung

- 1985 – Erste Windows-GUIs mit user und gdi
- 1991 – Ruby für VB
- 1995 – MSHTML (“Trident”)
- 2002 – Windows Forms
- 2006 – Windows Presentation Foundation

```

#include <windows.h>
LRESULT CALLBACK WndProc(HWND hwnd, UNIT msg, WPARAM wparam LPARAM lparam);
INT WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR cmdline, int cmdshow) {

    MSG msg;
    HWND hwnd;
    WNDCLASSEX wndclass = { 0 };
    wndclass.cbSize = sizeof(WNDCLASS);
    wndclass.style = CS_HREDRAW | CS_V REDRAW;
    wndclass.lpfnWndProc = WndProc;
    wndclass.hIcon = loadIcon(NULL, IDI_APPLICATION);
    wndclass.hCursor = LoadCursor (NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wndclass.lpszClassName = TEXT("Window1");
    wndclass.hInstance = hInstance;
    wndclass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
    RegisterClassEx(&wndclass);
    hwnd = CreateWindow(TEXT("Window1"), TEXT("Hello World"),
                       WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, 0, CW_USEDEFAULT,
                       0, NULL, NULL, hInstance, NULL);

    if ( !hwnd )
        return 0;
    ShowWindow(hwnd, SW_SHOWNORMAL);
    UpdateWindow(hwnd);
    while (GetMessage(&msg, NULL, 0, 0) ) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wparam, LPARAM lparam) {
    switch(msg) {
        case WM_DESTROY:
            PostQuitMessage(WM_QUIT);
            break;
        default:
            return DefWindowProc(hwnd, msg, wparam, lparam);
    }
    return 0;
}

```

```
using System.Windows.Forms;
using System;

class Program {
    [STAThread]
    static void Main() {
        Form f = new Form();
        f.Text = "Hello World";
        Application.Run(f);
    }
}
```

```
using System.Windows;  
using System;  
  
class Program {  
    [STAThread]  
    static void Main() {  
        Window f = new Window();  
        f.Title = "Hello World";  
        new Application().Run(f);  
    }  
}
```

# Merkmale

- Auflösungsunabhängigkeit
- Deklarative Programmierung
- Trennung von Darstellung und Verhalten
- Datenbindung
- Event-Handling
- Themes und Styles
- Dokumente, GUI und Medien
- Darstellung dreidimensionaler Objekte
- Hardware-Beschleunigung
- ...

# Merkmale

- Auflösungsunabhängigkeit
- Deklarative Programmierung
- Trennung von Darstellung und Verhalten
- Datenbindung
- Event-Handling
- Themes und Styles
- Dokumente, GUI und Medien
- Darstellung dreidimensionaler Objekte
- Hardware-Beschleunigung
- ...

# Vektor-Orientierung

Drück mich!

# Rastergrafik

Drück mich!

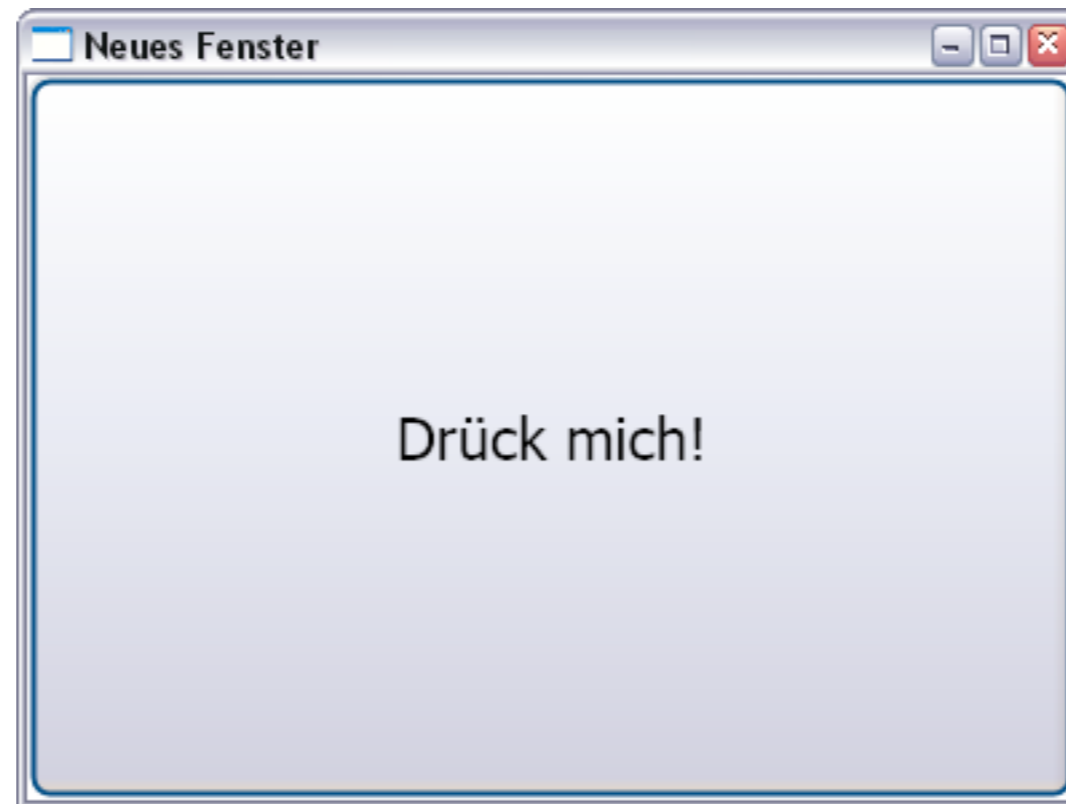


# Vektorgrafik

Drück mich!

**Drück mich!**

# Deklarative Programmierung



# Deklarative Programmierung

## Imperative Programmierung in C#

```
Window w = new Window();  
w.Title = "Neues Fenster";  
w.Width = 400;  
w.Height = 300;  
Button b = new Button();  
b.Content = "Drück mich!";  
w.Content = b;  
w.Show();
```

# Deklarative Programmierung

## Deklarative Programmierung mit XAML

```
<Window Title="Neues Fenster"  
    Width="400" Height="300">  
    <Button>  
        Drück mich!  
    </Button>  
</Window>
```



# Namensräume

```
<Window  
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
  xmlns:l="clr-namespace:MyApplication">  
  ...  
</Window>
```

# Content Properties

```
<TextBlock Text="Hallo Welt!" />
```

ist gleichbedeutend mit

```
<TextBlock>Hallo Welt!</TextBlock>
```

Der Inhalt eines Elements wird für die *Content Property* verwendet, in diesem Fall `Text`.

Auch `Collections` können *Content Property* sein.

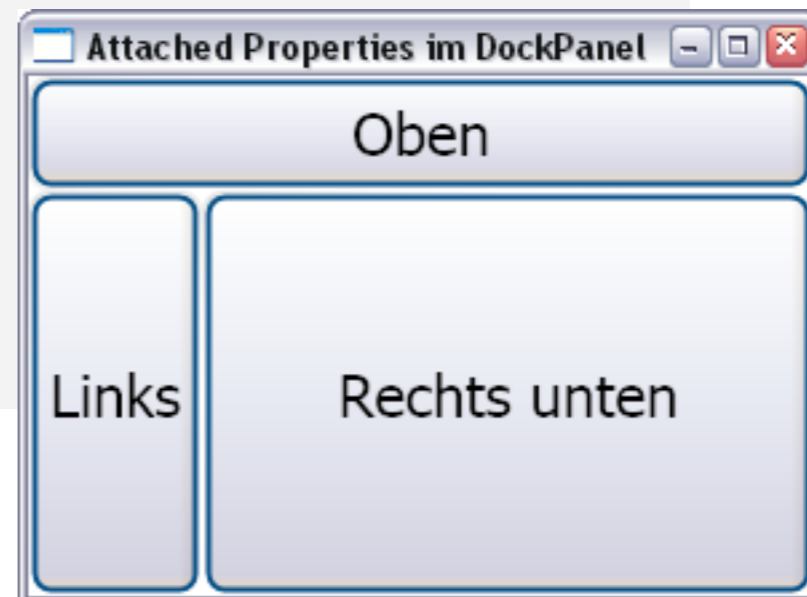
# Property Element Syntax

Alternative Deklaration von Attributen –  
insbesondere für komplexere Werte

```
<Window>  
  <Window.Title>Neues Fenster</Window.Title>  
  <Window.Width>400</Window.Width>  
  <Window.Height>300</Window.Height>  
  <Button>  
    <Button.Content>  
      Drück mich!  
    </Button.Content>  
  </Button>  
</Window>
```

# Attached Properties

```
<DockPanel>  
  <Button DockPanel.Dock="Top" ... />  
    Oben  
  </Button>  
  <Button DockPanel.Dock="Left" ... />  
    Links  
  </Button>  
  <Button ... />  
    Rechts unten  
  </Button>  
</DockPanel>
```



# Komposition



Drück mich!

Drück mich!

```
<Button>  
  Drück mich!  
</Button>
```

Drück mich!

(Zwischenschritt)

```
<Button>  
  <Button.Background>  
    ...  
  </Button.Background>  
  <TextBlock>  
    Drück mich!  
  </TextBlock>  
</Button>
```



Drück mich!

```
<Button>
  <Button.Background>
    <LinearGradientBrush>
      <GradientStop Color="Red" Offset="0" />
      <GradientStop Color="Green" Offset="1" />
    </LinearGradientBrush>
  </Button.Background>
  <TextBlock>
    Drück mich!
  </TextBlock>
</Button>
```

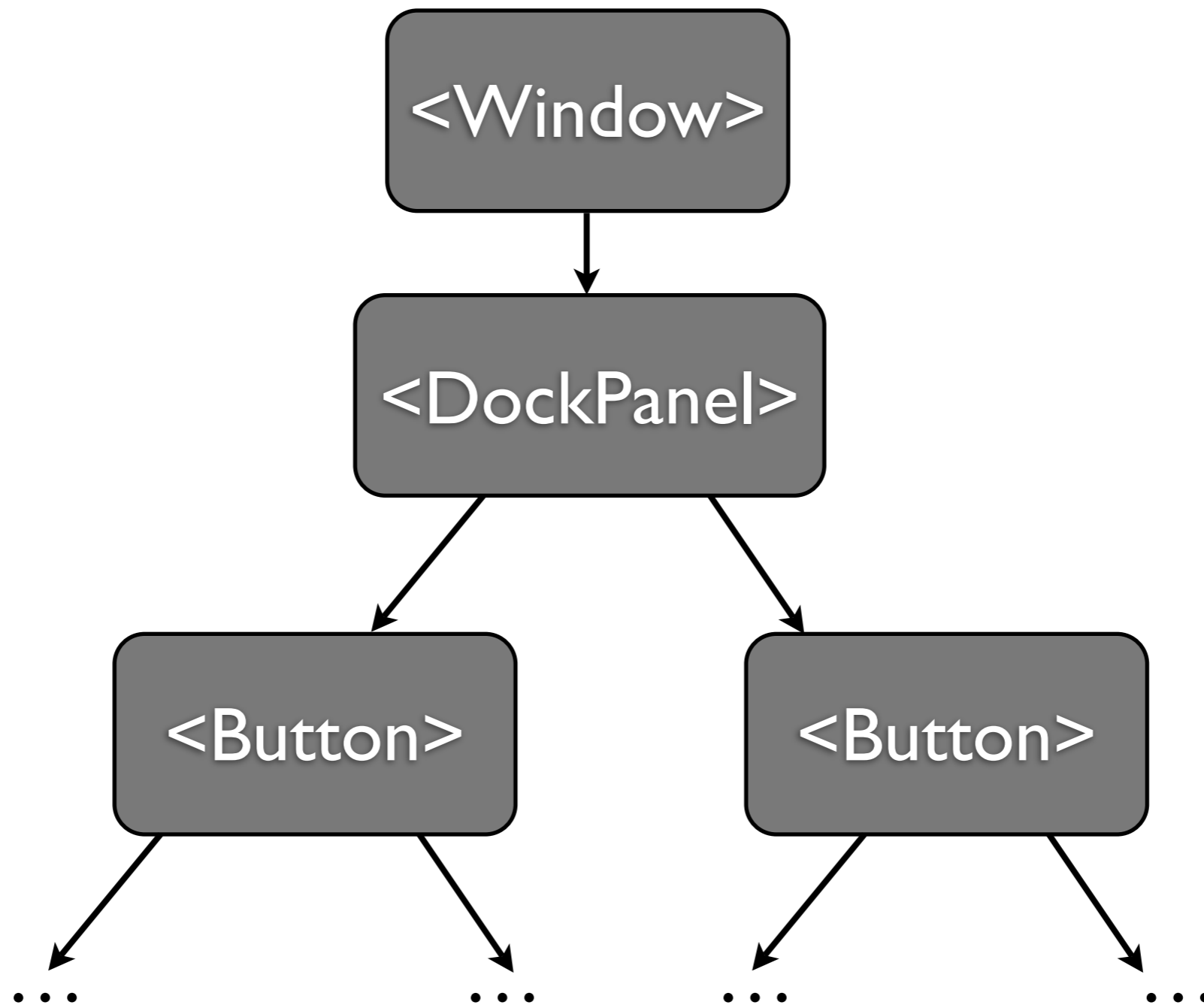


```
<Button>
  <Button.Background>
    <LinearGradientBrush>
      <GradientStop Color="Red" Offset="0" />
      <GradientStop Color="Green" Offset="1" />
    </LinearGradientBrush>
  </Button.Background>
  <TextBlock Text="Drück mich!">
    <TextBlock.Foreground>
      <RadialGradientBrush>
        <GradientStop Color="Blue" Offset="0" />
        <GradientStop Color="Yellow" Offset="1" />
      </RadialGradientBrush>
    </TextBlock.Foreground>
  </TextBlock>
</Button>
```

# Ressourcen

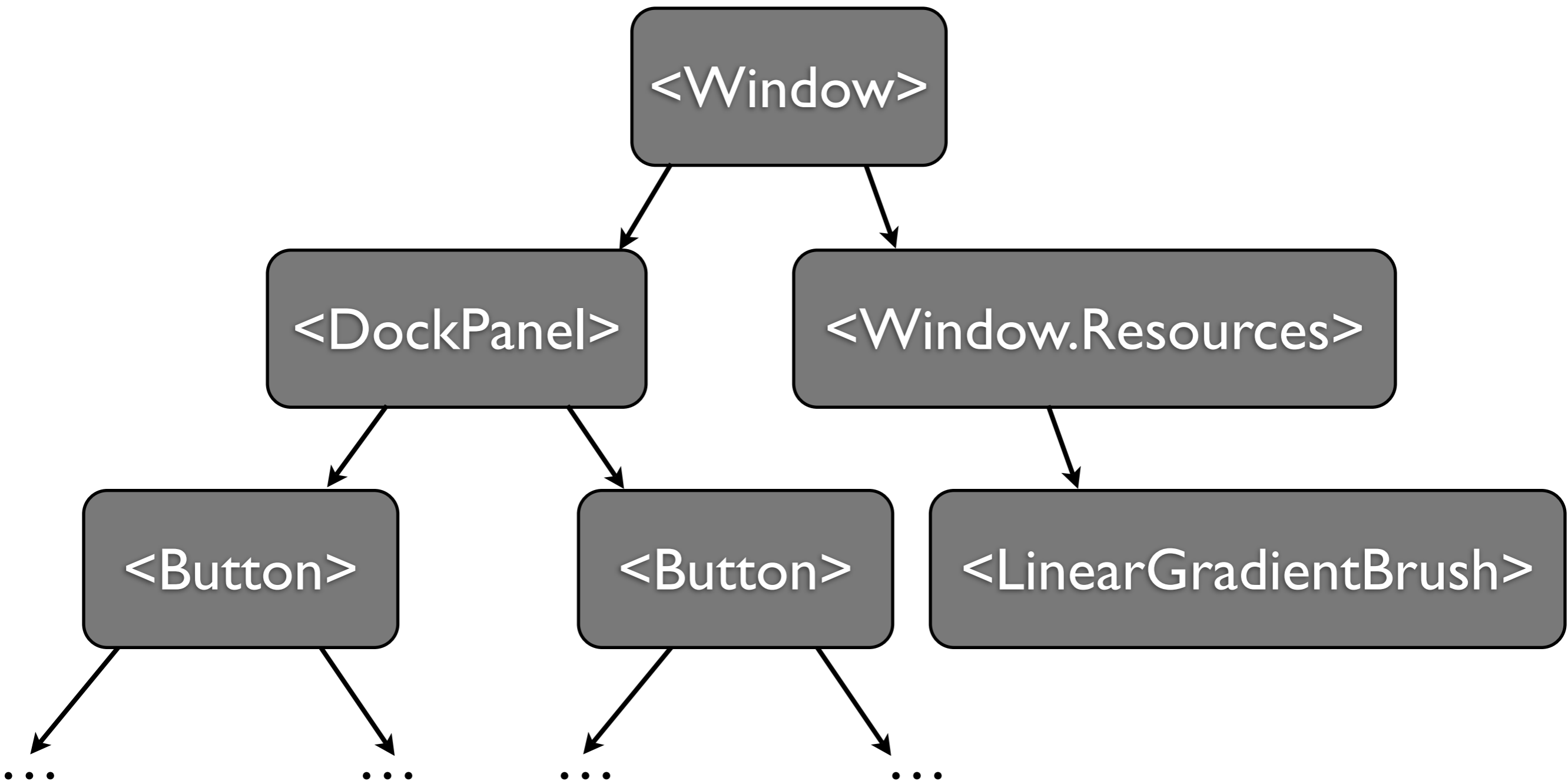
```
<Window ...>  
  <Window.Resources>  
    <LinearGradientBrush x:Key="bgBrush">  
      <GradientStop Color="Red" Offset="0" />  
      <GradientStop Color="Green" Offset="1" />  
    </LinearGradientBrush>  
    <RadialGradientBrush x:Key="fgBrush">  
      <GradientStop Color="Blue" Offset="0" />  
      <GradientStop Color="Yellow" Offset="1" />  
    </RadialGradientBrush>  
  </Window.Resources>  
  ...  
</Window>
```

# Markup Extensions



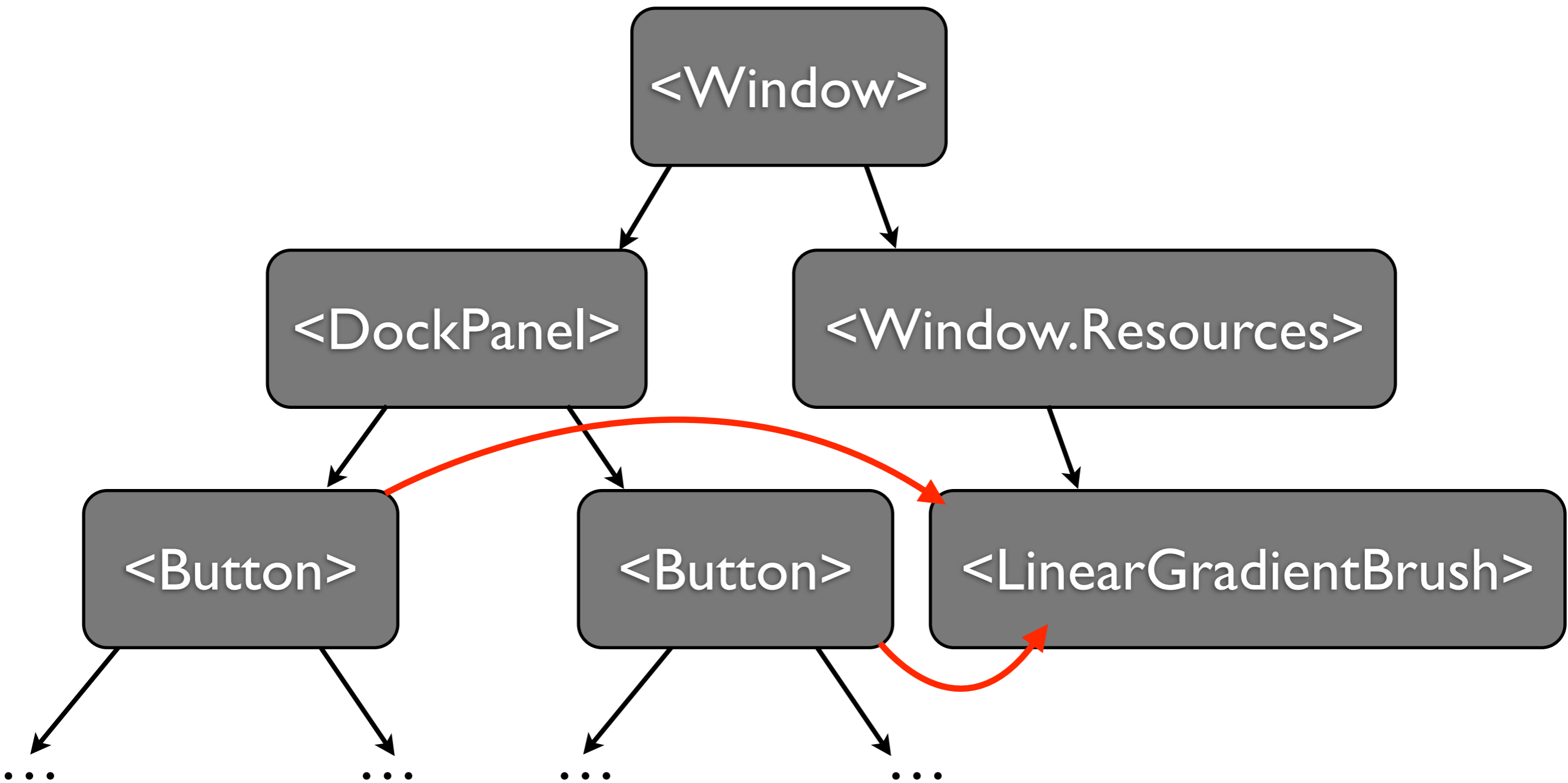
XML-Dokumente sind hierarchisch aufgebaut

# Markup Extensions



XML-Dokumente sind hierarchisch aufgebaut

# Markup Extensions

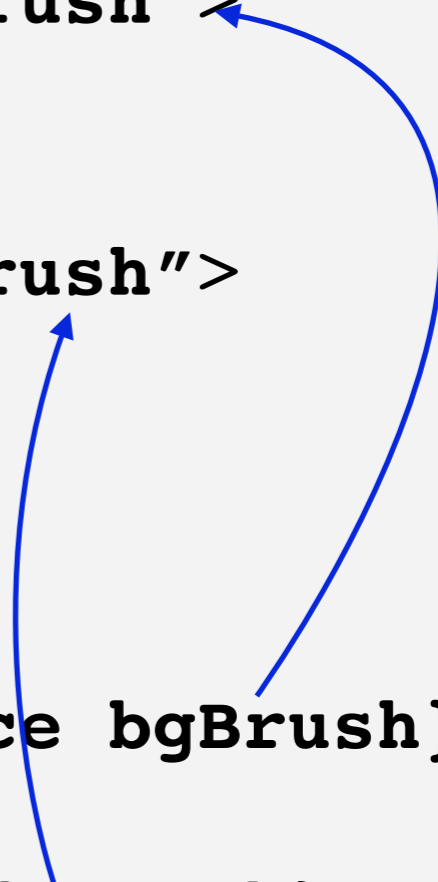


XML-Dokumente sind hierarchisch aufgebaut

# Markup Extensions


Markup Extensions ermöglichen Querverweise

```
<Window.Resources>
  <LinearGradientBrush x:Key="bgBrush">
    ...
  </LinearGradientBrush>
  <RadialGradientBrush x:Key="fgBrush">
    ...
  </RadialGradientBrush>
</Window.Resources>
...
<Button Background="{StaticResource bgBrush}">
  <TextBlock Text="Drück mich!"
    Foreground="{StaticResource fgBrush}">
</Button>
```



# Control Templates

```
<ControlTemplate x:Key="MyButton"
  TargetType="{x:Type Button}" >
  <Border CornerRadius="2"
    BorderThickness="1" BorderBrush="...">
    <TextBlock
      Background="{StaticResource bgBrush}"
      Foreground="{StaticResource fgBrush}">
      <ContentPresenter />
    </TextBlock>
  </Border>
</ControlTemplate>
...
<Button Content="Foo"
  Template="{StaticResource MyButton}" />
```




# “Code-behind”

Geteilte Definition einer Klasse

W.xaml

```
<Window x:class="MyApp.W"  
  Title="Neuer Titel">  
  ...  
</Window>
```



W.xaml.cs

```
namespace MyApp {  
  public partial class W : Window {  
    public W() {  
      InitializeComponent();  
      Title = "Neuerer Titel";  
    }  
  }  
}
```


# Darstellung...

## W.xaml

```
<Button ↵  
    Height="50" ↵  
    Width="400" ↵  
    Background="Red" ↵  
    Content="Drück mich!" />
```

# ...und Verhalten

W.xaml.cs

```
private void Button_Click ( 
    object sender, RoutedEventArgs e) {
    theButton.Content = "Gedrückt!";
}
```

# Die Verbindung

W.xaml

```
<Button ... ↵  
    Click="Button_Click" ↵  
    x:Name="theButton" />
```

# Datenbindung

Applikation soll aktuelle Daten anzeigen

Aktualisierung der Anzeige ist  
umständlich und fehlerträchtig


Format der Daten soll unabhängig  
von der Präsentation sein

# Datenbindung Modell

```
class Data : INotifyPropertyChanged {  
    ...  
    private bool _status;  
    public bool Status {  
        get { return _status; }  
        set {  
            _status = value;  
            PropertyChanged(this, new  
                PropertyChangedEventArgs("Status"));  
        }  
    }  
}
```

# Datenbindung

## Code-behind

```
class W {  
    Data theData = new Data();  
  
    public W() {  
        DataContext = theData;  
        InitializeComponent();  
    }  
  
    private void Toggle(...) {  
        theData.Status =  
            !theData.Status;   
    }  
}
```

# Datenbindung

# Anzeige

```
<Button ... ↵  
    Click="Toggle" ↵  
    Content="{Binding Path="Status}" />
```

# Data Templates

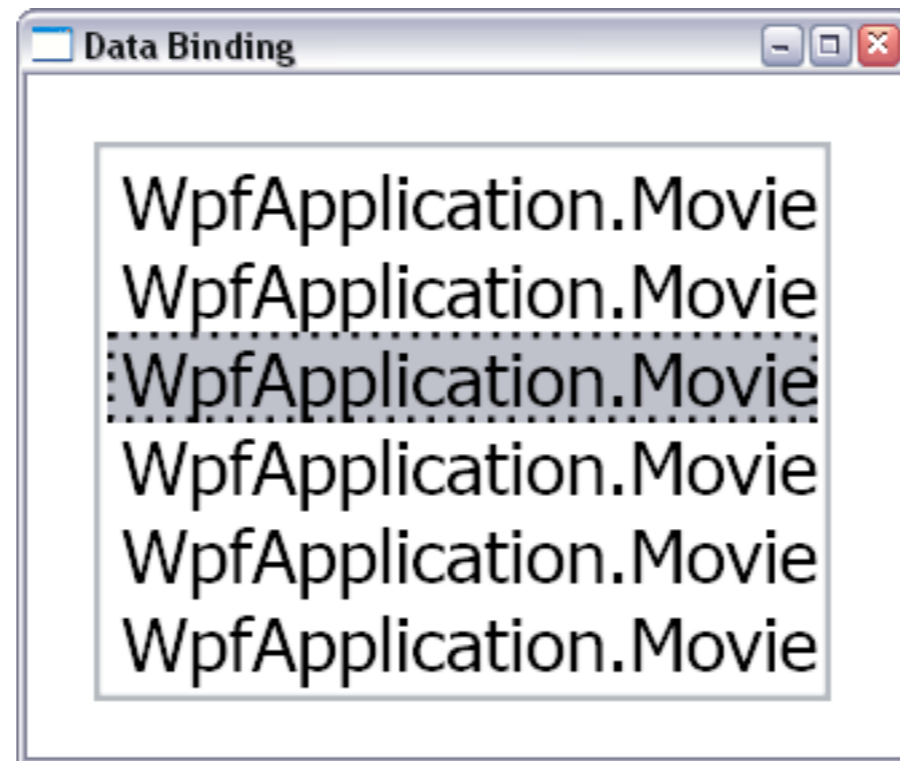
```
ObservableCollection<Movie> movies =  
    new ObservableCollection<Movie>();  
  
public W() {  
    movies.Add(new Movie("Finding Nemo", 2003));  
    movies.Add(new Movie("Batman Begins", 2005));  
    movies.Add(new Movie("Pulp Fiction", 1994));  
    movies.Add(new Movie("The Return of the King", 2003));  
    movies.Add(new Movie("The Fellowship of the Ring", 2001));  
    movies.Add(new Movie("The Two Towers", 2002));  
    DataContext = movies;  
    InitializeComponent();  
}
```

# Data Templates

```
<ListBox ItemsSource="{Binding Path=."} />
```

# Data Templates

```
<ListBox ItemsSource="{Binding Path=."}"/>
```

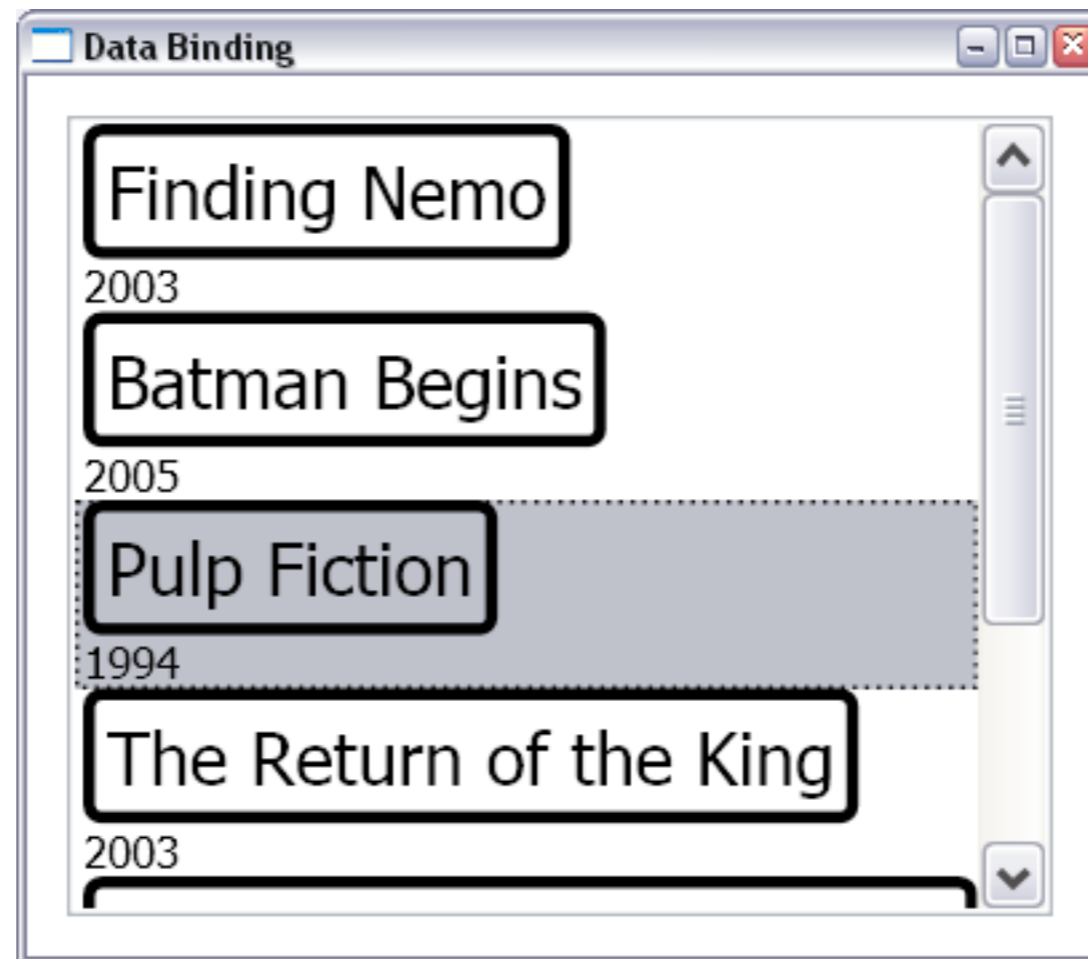


# Data Templates

```
<DataTemplate
  x:Key="MovieTemplate" ↵
  DataType="{x:Type l:Movie}">
<StackPanel ...>
  <Border ...>
    <TextBlock FontSize="24" ↵
      Text="{Binding Title}" />
  </Border>
  <TextBlock ↵
    Text="{Binding Year}" />
</StackPanel>
</DataTemplate>
```

# Data Templates

```
<ListBox ItemsSource="{Binding Path=."} " ↵  
  ItemsTemplate="{StaticResource MovieTemplate}" />
```



# XAML Browser Application

Im Browser ausführbar mit seitenbasierter  
Browser-Navigation (Vor, Zurück, History)

page1.xaml

```
<Page ... WindowTitle="Browser Application">
  <StackPanel>
    <TextBlock FontSize="24">
      Hallo Welt!
    </TextBlock>
    <TextBlock>
      <Hyperlink NavigateUri="page2.xaml">
        Und weiter gehts...
      </Hyperlink>
    </TextBlock>
  </StackPanel>
</Page>
```



# Silverlight

WPF/Everywhere

# Was ist Silverlight?

Laufzeitumgebung für *Rich  
Internet Applications*

# Version 1.0

- Einbettung in HTML
- Canvas für Grafikprimitive
- Steuerung über DOM per JavaScript
- Animation und Event-Handling
- MP3, WMA, WMV

# Version 1.1

“1.1 Alpha Refresh September Preview”

# Version 2.0

- Code-behind in C# und VB.NET
- Controls und Layouts
- Base Class Library (Collections, LINQ, ...)
- Beta-Release Anfang März 2008

# Einbettung in HTML

```
<html>
  <head>
    <script type="text/javascript" src="Silverlight.js"></script>
    <script type="text/javascript" src="createSilverlight.js"></script>
  </head>
  <body>
    <h1>Hallo Welt!</h1>
    <div id="slContainer"></div>
    <script type="text/javascript">
      var parentElement =
        document.getElementById("slContainer");
      createMySilverlightPlugin();
    </script>
  </body>
</html>
```

# Vergleich

Windows Presentation Foundation und Silverlight

# Stand-Alone (WPF)

- Windows
- Flexibilität in der Entwicklung
- Benötigt Installation

# XBAP (WPF)

- Windows mit MSIE oder Firefox
- Keine Installation
- Verwendung wie Website
- Eingeschränkte Rechte

# Silverlight 1.0

- Windows und Mac
- Einbettung in HTML
- Entwicklung nur mit JS

# Silverlight 2.0

- Windows und Mac
- Entwicklung mit C# und .NET
- Noch nicht verfügbar