WPF

Übersicht

- Windows Presentation Foundation
- Werkzeug zur Entwicklung grafischer Benutzeroberflächen
- deklarative Definition erfolgt mit der Beschreibungs-Sprache: XAML
- XAML
  - Extensible Application Markup Language
  - XML (Dialekt) Beschreibungsprache für Oberflächen
  - Oberflächendesigner arbeitet mit XAML
  - Ziel: Trennung Design und Implementierung
    Getrenntes Arbeiten von Designer und Entwickler
  - spezielle Designtools (MS Expression Blend)
- Compiler erzeugt aus XAML-Code automatisch Instanzen von Klassen (Bsp.: Klasse Button)

WPF

Design: XAML-Datei

```
<Window x:Class="WPF1.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="WPF1" Height="300" Width="300">
 <Grid>
  <TextBox Height="34" Margin="65,34,83,0" Name="textBox1"
   VerticalAlignment="Top" ></TextBox>
  <Button Margin="65,122,81,113" Name="button1"
          Click="button1_Click">Rechne</Button>
 </Grid>
```

Implementierung: Csharp-Datei (Funktionalität)

```
public partial class Window1 : System.Windows.Window
{public Window1(){ InitializeComponent();
    button1.Click += new RoutedEventHandler(button1_Click);}
 public void button1_Click(object sender, RoutedEventArgs e)
   {textBox1.Text = "Click";}}
```

WPF

Merkmale

- Benutzeroberfläche wird deklarativ definiert
  Code separat (ähnelt: ASP Code-Behind-Modell)

- Oberfläche verwendbar in
  - normaler Windowsanwendung (Fenster)
  - Browser
- Hierarchische Oberflächenbeschreibung
- unterstützt 2D- und 3D-Grafiken

- Ausgabe ist vektorbasiert (=> skalierbar)
- vielfältige grafische Unterstützung (Video, Bilder, Audio) und sehr gute Datenanbindungsmöglichkeiten
- schnelle Grafikausgabe

WPF

WinForm oder WPF ?

- WPF extrem leistungsfähig
- Microsoft entwickelt WinForm nicht mehr weiter

  ⇨ WPF

Entwicklungstools

- Visual Studio
- Designertool: Expression Blend

WPF

Anwendungstypen

- **WPF-Anwendung**
  entsprechen Windows-Anwendungen,
  stellen eigenes Fenster bereit

- **WPF-Browseranwendung**
  stellen kein eigenes Fenster bereit
  laufen im Browser

- **WPF-Benutzersteuerelementebibliothek**
  neues Steuerelement aus bestehenden zusammensetzen

- **Benutzerdefinierte Steuerelementebiliothek**
  eigene Steuerelemente entwicklen

WPF

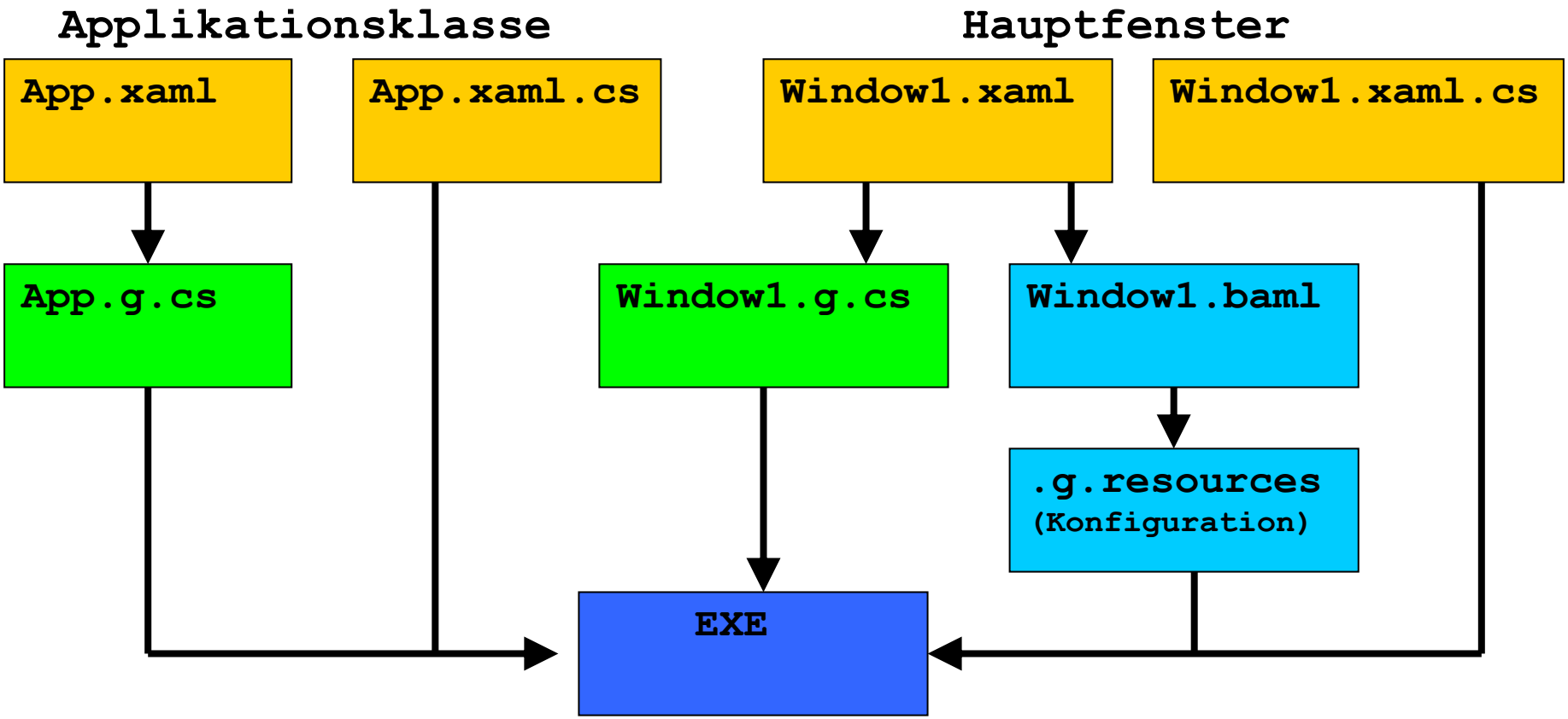
WPF-Anwendung

- entspricht Windows-Anwendung

- Visual Studio erstellt die Dateien:
  App.xaml
    Verweis auf Startfenster und Ressourcen
  App.xaml.cs
    enthält die Applikations-Klasse App (veröffentlicht
    Eigenschaften und löst Ereignisse aus)
  MainWindow.xaml
    beschreibt Oberfläche und
    stellt Bezug zu Programmcode her
  MainWindow.xaml.cs
    enthält Programmcode zur Oberfläche (Code-Behind)

WPF

Kompilation einer WPF-Anwendung



WPF

XAML

- eXtensible Application Markup Language

- deklarative Programmiersprache

- Basis: XML

- jede XAML-Datei besitzt ein Wurzelelement
  - WPF-Anwendung: Window
  - Browser-Anwendung: Page

- innerhalb des Wurzelementes erfolgt Beschreibung i. d. R. werden Container (Grid,StackPanel,...) verwendet. Elemente können verschachtelt werden.

WPF

XAML-Elemente

- entsprechen WPF-Klassen (Serialisierungsformat)

- Bsp.: Button
  - Deklaration per XAML

```
<Button Margin="65,122,81,113" Name="button1">Rechne</Button>
```

  - Codierung per Programm (nicht als XAML)

```
Button btn = new Button();
Btn.Content = "button1";
this.AddChild(btn);
```

    AddChild führt die Zuordnung zum übergeordneten Container aus

WPF

Ereignisse

- Die Verarbeitung von Benutzerinteraktionen erfolgt über Ereignisse
- Basis: Observer-Pattern, Delegaten

- Bsp.: Button-Click
  in XAML

```
<Button Margin="65,122,81,113" Name="button1"
Click="button1_Click">Rechne</Button>
```

  als Programmcode

```
button1.Click += new RoutedEventHandler(button1_Click);
```

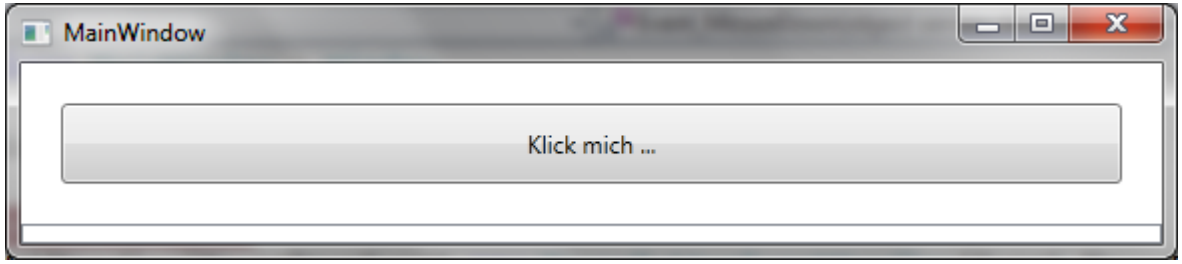
- WPF-Elemente können verschachtelt werden

WPF

Verschachtelte Elemente

Wie erfolgt die Eventverarbeitung bei verschachtelten Elementen?

Bsp.:



Klick auf Label „Klick mich...“

Label in Button in Grid in Window

Ereignistypen

- **Direkte Events**
  Werden nur von den Elementen verarbeitet bei denen sie aufgetreten sind.

- **Tunneling-Events**
  weiterreichen von der Wurzel an untergeordnete Elemente

- **Bubbling-Events**
  weiterreichen an übergeordnete Elemente
  entspricht der Bearbeitung von Standardevents

Beispiel: Mouse-Events

- GotMouseCapture (Bubble)
- MouseEnter(Direct)
- MouseLeave (Direct)
- PreviewMouseLeftButtonDown(Tunnel) (Prefix: Preview)
  MouseLeftButtonDown(Bubble)
- [Preview]MouseMove(Tunnel,Bubble)
- . . .

Verarbeitung

1. Verarbeitung der Tunneling-Events
2. Verarbeitung der Bubbling-Events

Ereignisse abbrechen: `e.Handled = true;`

WPF

WPF Container

- Window
  - entspricht herkömmlicher WinForm-Technologie
  - Anwendung durch mehrere Fenster gekennzeichnet
  - Navigation: öffnen und schliessen von Fenstern
  - Klasse Window: Container für Steuerelemente
- NavigationWindow
  - entspricht Konzept von Internet-Browsern
  - Inhalte werden auf mehrere Seiten verteilt
  - Navigation: Vor und Zurück
  - Klasse NavigationWindow
    - von Window abgeleitet
    - Inhalte durch Page-Objekte beschrieben
    - System.Windows.Controls.Page kapselt Inhaltsseite, zu der navigiert werden kann

WPF

Layout-Container

Anordnung und Darstellung der Elemente wird durch Layout-Container geregelt

- i. d. R. relative Positionen
- nur Canvas arbeitet mit absoluten Positionen
- Layout-Container können verschachtelt werden

Layout-Container-Typen

- Canvas
- StackPanel
- WrapPanel
- DockPanel
- Uniform Grid
- Grid

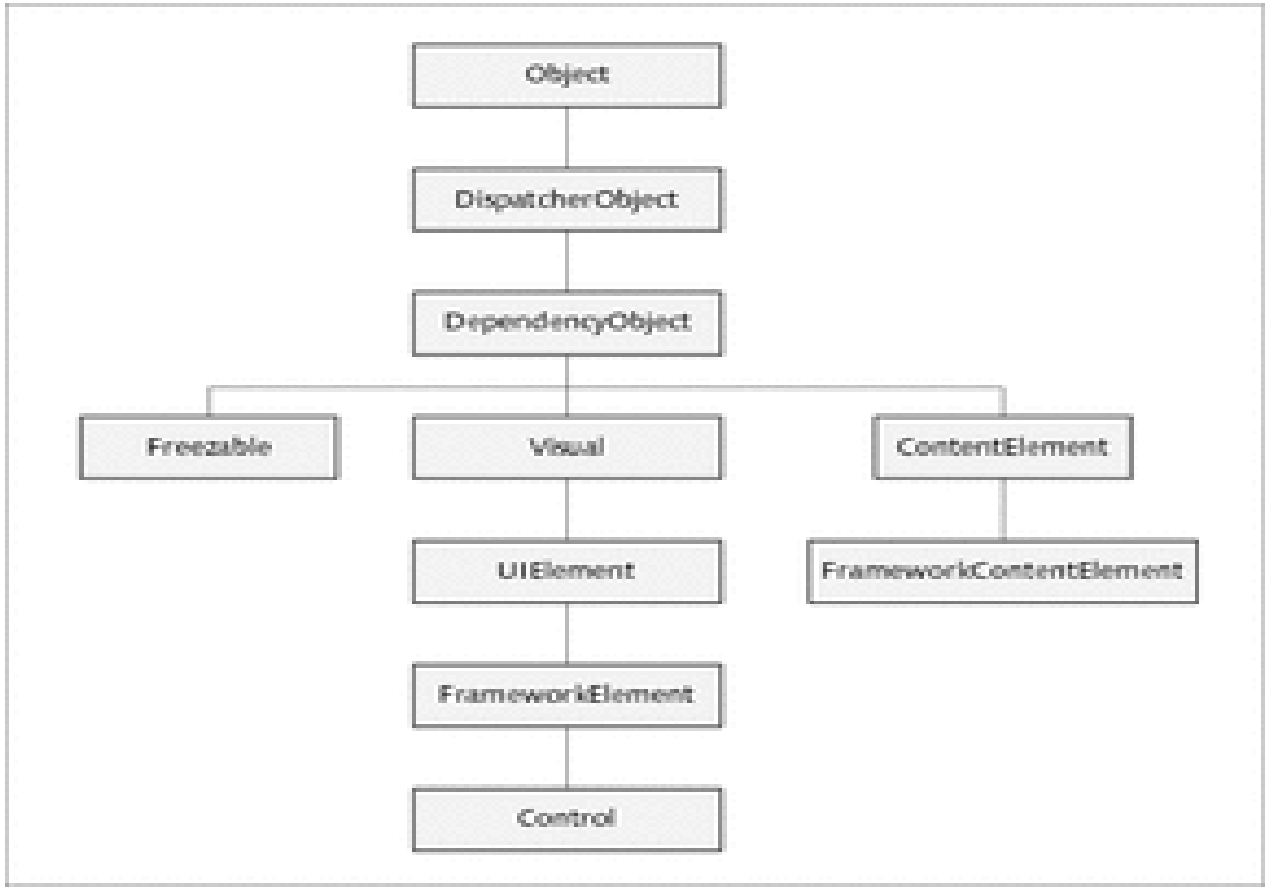
WPF-Steuerelemente

- besitzen tiefe Vererbungshierarchie (siehe unten)

- Content Eigenschaft
  - in Basisklasse ContentControl
  - Kann genau ein Element vom Typ Object enthalten:
    - Text oder
    - beliebiges anderes Steuerelement (Verschachtelung)
      => vielfältige Gestaltungsmöglichkeiten

WPF

Steuerelemente-Vererbungs-Hierarchie



WPF

Datenbindung

- Synchronisation zwischen DatenQuelle und Steuerelement
- alle abhängigen Eigenschaften sind datenbindungsfähig

- Datenquellen
  - Eigenschaften anderer Komponenten
  - XML-Datei
  - Collection
  - Datenbank

- Binding
  Beschreibt Datenbindung zwischen Quelle und Komponente
- DataContext
  stellt Daten bereit (Bindeglied: Quelle – Komponente)
  Property, die Defaultsource des Bindings enthält

WPF

Beispiel: Synchronisation Textbox

```
<TextBox Height="34" Margin="65,34,83,0" Name="textBox1"
         VerticalAlignment="Top">
</TextBox>
<TextBox Height="23" HorizontalAlignment="Left"
         Margin="65,82,0,0" Name="textBox2"
         VerticalAlignment="Top" Width="132">

         <Binding ElementName="textBox1" Path="Text" />

</TextBox>
```

- **ElementName gibt Namen der Datenquelle an**