

Windows Presentation Foundation

Artikel • 02.03.2024

Windows Presentation Foundation (WPF) bietet Entwicklern ein einheitliches Programmiermodell zum Erstellen von Desktopbranchenanwendungen unter Windows.

- [Einführung in WPF](#)
- [Erste Schritte](#)
- [Anwendungsentwicklung](#)
- [Erweitert](#)
- [Steuerelemente](#)
- [Daten](#)
- [Grafiken und Multimedia](#)
- [Security](#)
- [WPF-Beispiele](#)
- [Klassenbibliothek](#)

Zusammenarbeit auf GitHub

Die Quelle für diesen Inhalt finden Sie auf GitHub, wo Sie auch Issues und Pull Requests erstellen und überprüfen können. Weitere Informationen finden Sie in unserem [Leitfaden für Mitwirkende](#).

.NET

Feedback zu .NET Desktop feedback

.NET Desktop feedback ist ein Open Source-Projekt. Wählen Sie einen Link aus, um Feedback zu geben:

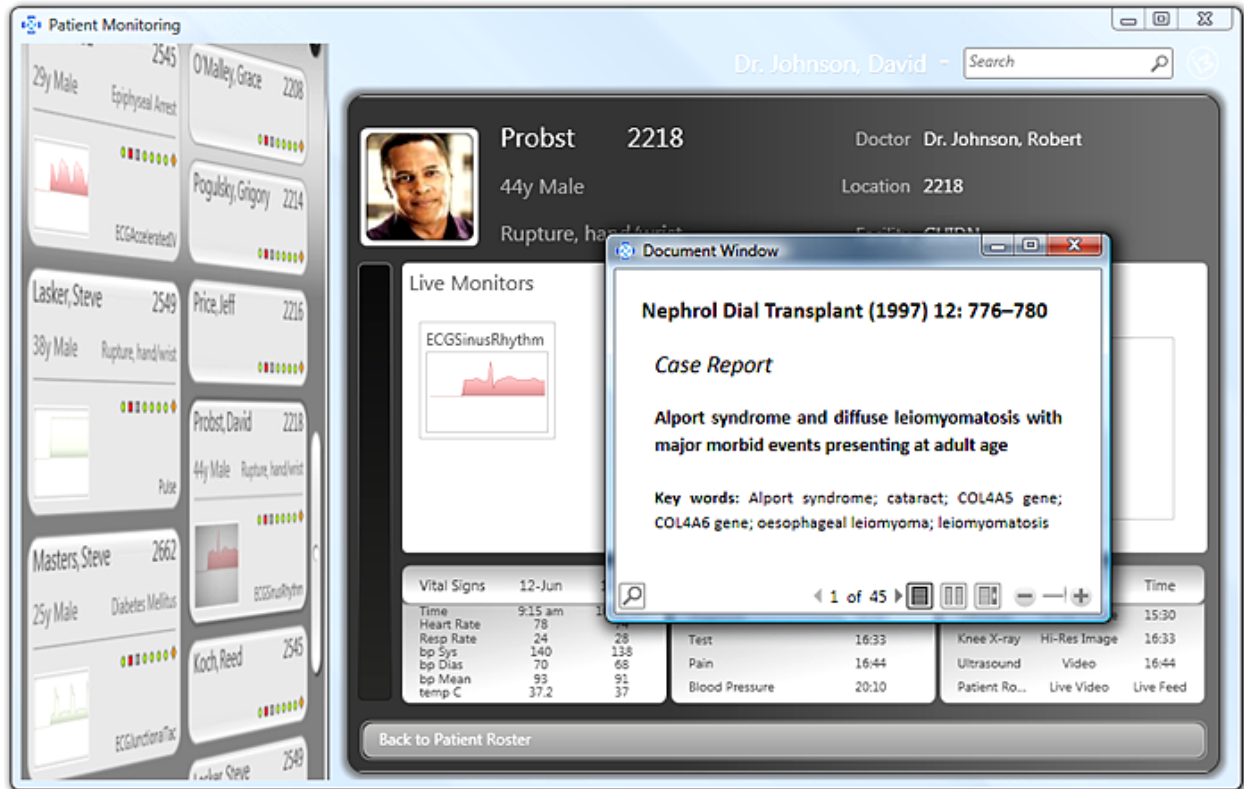
 [Problem in der Dokumentation öffnen](#)

 [Abgeben von Produktfeedback](#)

WPF-Übersicht

Artikel • 04.05.2023

Mit Windows Presentation Foundation (WPF) können Sie Desktop-Clientanwendungen für Windows erstellen, die visuell herausragende Benutzeroberflächen haben.



Der Kern von WPF ist eine auflösungsunabhängige und vektorbasierte Rendering-Engine, die die Leistungsfähigkeit moderner Grafikhardware nutzt. Dieser Kern wird durch WPF um mehrere Anwendungsentwicklungsfeatures erweitert, wozu Extensible Application Markup Language (XAML), Steuerelemente, Datenbindung, Layout, 2D- und 3D-Grafik, Animationen, Stile, Vorlagen, Dokumente, Medien, Text und Typographie zählen. WPF ist Teil von .NET. Sie können also Anwendungen erstellen, die andere Elemente der .NET-API beinhalten.

Diese Übersicht ist für Einsteiger gedacht und beschreibt die wichtigsten Funktionen und Konzepte von WPF.

Programmieren mit WPF

WPF ist eine Teilmenge von .NET-Typen, die sich (zum größten Teil) im [System.Windows](#)-Namespace befinden. Wenn Sie bereits Anwendungen mit .NET und verwalteten Technologien wie ASP.NET und Windows Forms erstellt haben, sind Sie mit den Grundlagen der WPF-Programmierung vertraut. Sie instanziiieren Klassen, legen

Eigenschaften fest, rufen Methoden auf und behandeln Ereignisse und verwenden dafür Ihre bevorzugte .NET Framework-Programmiersprache, etwa C# oder Visual Basic.

WPF umfasst zusätzliche Programmierkonstrukte, mit denen Eigenschaften und Ereignisse erweitert werden: [Abhängigkeitseigenschaften](#) und [Routingereignisse](#).

Markup und CodeBehind

WPF ermöglicht es Ihnen, eine Anwendung sowohl mit *Markup* als auch mit *CodeBehind* zu entwickeln, eine Vorgehensweise, mit der ASP.NET-Entwickler vertraut sein sollten. Im Grundsatz verwenden Sie XAML-Markup, um die Darstellung einer Anwendung zu implementieren, und verwaltete Programmiersprachen (CodeBehind), um das Verhalten der Anwendung zu implementieren. Diese Trennung von Darstellung und Verhalten bietet folgende Vorteile:

- Entwicklungs- und Wartungskosten werden verringert, weil darstellungsspezifisches Markup nicht unmittelbar mit verhaltensspezifischem Code gekoppelt ist.
- Die Entwicklung ist effizienter, da Designer die Darstellung einer Anwendung implementieren können, während Entwickler gleichzeitig das Verhalten der Anwendung implementieren.
- Die [Globalisierung und Lokalisierung](#) für WPF-Anwendungen wird stark vereinfacht.

Markup

XAML ist eine auf XML basierende Markupsprache, mit der die Darstellung einer Anwendung deklarativ implementiert werden kann. Sie wird üblicherweise dazu verwendet, Fenster, Dialogfelder, Seiten und Benutzersteuerelemente zu erstellen und diese mit Steuerelementen, Formen und Grafiken zu füllen.

Im folgenden Beispiel wird XAML verwendet, um die Darstellung eines Fensters zu implementieren, das eine einzelne Schaltfläche enthält:

XAML

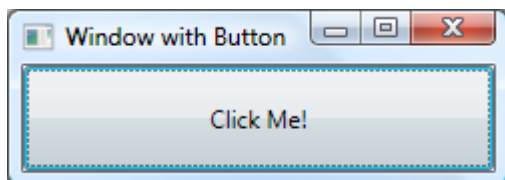
```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  Title="Window with Button"
  Width="250" Height="100">

  <!-- Add button to window -->
  <Button Name="button">Click Me!</Button>
```

```
</Window>
```

In diesem XAML-Code werden ein Fenster und eine Schaltfläche definiert, indem das `Window` - und das `Button` -Element verwendet werden. Jedes Element wird mit Attributen konfiguriert. Hier wird beispielsweise das `Window` -Attribut des `Title` -Elements verwendet, um den Text für die Titelleiste des Fensters festzulegen. Zur Laufzeit werden die im Markup definierten Elemente und Attribute von WPF in Instanzen von WPF-Klassen konvertiert. Beispielsweise wird das `Window` -Element in eine Instanz der `Window` -Klasse konvertiert, deren `Title` -Eigenschaft dem Wert des `Title` -Attributs entspricht.

In der folgenden Abbildung ist die Benutzeroberfläche dargestellt, die durch den XAML-Code im vorherigen Beispiel definiert ist:



Da XAML auf XML basiert, wird die damit erstellte Benutzeroberfläche in einer Hierarchie geschachtelter Elemente zusammengestellt, die als **Elementstruktur** bezeichnet wird. Die Elementstruktur stellt eine logische und intuitive Art und Weise zum Erstellen und Verwalten von Benutzeroberflächen bereit.

CodeBehind

Der Hauptzweck einer Anwendung besteht darin, die Funktionalität zu implementieren, mit der auf Benutzeraktionen reagiert wird, wozu auch das Behandeln von Ereignissen (z. B. Klicken auf Menüs, Symbolleisten oder Schaltflächen) sowie das Aufrufen von Geschäftslogik und als Reaktion darauf von Datenzugriffslogik zählen. In WPF wird dieses Verhalten in Code implementiert, der mit Markup verknüpft ist. Diese Art von Code wird als CodeBehind bezeichnet. Im folgenden Beispiel werden der aktualisierte Markupcode aus dem vorherigen Beispiel und das CodeBehind gezeigt:

XAML

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.AWindow"
    Title="Window with Button"
    Width="250" Height="100">
```

```
<!-- Add button to window -->
<Button Name="button" Click="button_Click">Click Me!</Button>

</Window>
```

C#

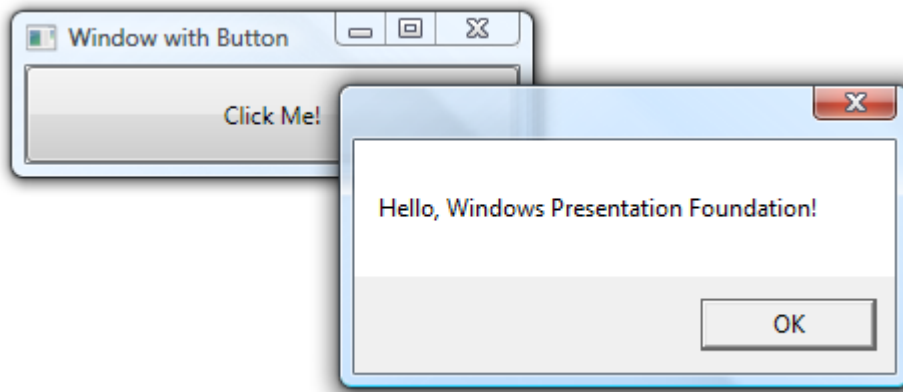
```
using System.Windows; // Window, RoutedEventArgs, MessageBox

namespace SDKSample
{
    public partial class AWindow : Window
    {
        public AWindow()
        {
            // InitializeComponent call is required to merge the UI
            // that is defined in markup with this class, including
            // setting properties and registering event handlers
            InitializeComponent();
        }

        void button_Click(object sender, RoutedEventArgs e)
        {
            // Show message box when button is clicked.
            MessageBox.Show("Hello, Windows Presentation Foundation!");
        }
    }
}
```

In diesem Beispiel wird im CodeBehind eine Klasse implementiert, die aus der [Window](#) - Klasse abgeleitet wird. Das `x:Class` -Attribut wird verwendet, um den Markupcode mit der CodeBehind-Klasse zu verknüpfen. `InitializeComponent` wird vom Konstruktor der CodeBehind-Klasse aufgerufen, um die im Markup definierte Benutzeroberfläche mit der CodeBehind-Klasse zusammenzuführen. (`InitializeComponent` wird für Sie generiert, wenn Ihre Anwendung erstellt wird, weshalb Sie sie nicht manuell implementieren müssen.) Die Kombination von `x:Class` und `InitializeComponent` stellt sicher, dass Ihre Implementierung ordnungsgemäß initialisiert wird. In der CodeBehind-Klasse wird außerdem ein Ereignishandler für das [Click](#)-Ereignis der Schaltfläche implementiert. Wird auf die Schaltfläche geklickt, zeigt der Ereignishandler ein Meldungsfeld an, indem er die [System.Windows.MessageBox.Show](#) -Methode aufruft.

In der folgenden Abbildung ist das Ergebnis dargestellt, das nach einem Klicken auf die Schaltfläche zu sehen ist:



Steuerelemente

Die Elemente einer Benutzeroberfläche, die mit dem Anwendungsmodell bereitgestellt werden, sind konstruierte Steuerelemente. In WPF ist *Steuerelement* ein Sammelbegriff, der sich auf eine Kategorie von WPF-Klassen bezieht, die entweder in einem Fenster oder auf einer Seite gehostet werden, eine Benutzeroberfläche haben und ein bestimmtes Verhalten implementieren.

Weitere Informationen finden Sie unter [Steuerelemente](#).

WPF-Steuerelemente nach Funktion

Die integrierten WPF-Steuerelemente sind nachstehend aufgelistet:

- Schaltflächen: [Button](#) und [RepeatButton](#).
- Datenanzeige: [DataGrid](#), [ListView](#), und [TreeView](#).
- Datumsanzeige und -auswahl: [Calendar](#) und [DatePicker](#).
- Dialogfelder: [OpenFileDialog](#), [PrintDialog](#) und [SaveFileDialog](#).
- Freihandeingaben: [InkCanvas](#) und [InkPresenter](#).
- Dokumente: [DocumentViewer](#), [FlowDocumentPageViewer](#), [FlowDocumentReader](#), [FlowDocumentScrollViewer](#) und [StickyNoteControl](#).
- Eingabe: [TextBox](#), [RichTextBox](#) und [PasswordBox](#).
- Layout: [Border](#), [BulletDecorator](#), [Canvas](#), [DockPanel](#), [Expander](#), [Grid](#), [GridView](#), [GridSplitter](#), [GroupBox](#), [Panel](#), [ResizeGrip](#), [Separator](#), [ScrollBar](#), [ScrollViewer](#), [StackPanel](#), [Thumb](#), [Viewbox](#), [VirtualizingStackPanel](#), [Window](#) und [WrapPanel](#).
- Medien: [Image](#), [MediaElement](#) und [SoundPlayerAction](#).

- **Menüs:** [ContextMenu](#), [Menu](#) und [ToolBar](#).
- **Navigation:** [Frame](#), [Hyperlink](#), [Page](#), [NavigationWindow](#) und [TabControl](#).
- **Auswahl:** [CheckBox](#), [ComboBox](#), [ListBox](#), [RadioButton](#) und [Slider](#).
- **Benutzerinformationen:** [AccessText](#), [Label](#), [Popup](#), [ProgressBar](#), [StatusBar](#), [TextBlock](#) und [ToolTip](#).

Eingabe und Befehle

Steuerelemente sind üblicherweise dafür vorgesehen, Benutzereingaben zu erkennen und darauf zu reagieren. Im [WPF-Eingabesystem](#) werden sowohl direkte Ereignisse als auch Routingereignisse verwendet, um Texteingaben, Fokusverwaltung und Mauspositionierung zu unterstützen.

Anwendungen haben häufig komplexe Eingabeanforderungen. WPF stellt ein [Befehlssystem](#) bereit, über das Eingabeaktionen eines Benutzers von dem Code getrennt sind, mit dem auf diese Aktionen reagiert wird.

Layout

Wenn Sie eine Benutzeroberfläche erstellen, ordnen Sie die Steuerelemente nach Position und Größe an, um ein Layout zu formen. Eine Hauptanforderung für jedes Layout besteht darin, sich an Änderungen der Fenstergröße und Anzeigeeinstellungen anpassen zu können. WPF bietet ein erstklassiges erweiterbares Layoutsystem, sodass Sie keinen zusätzlichen Code schreiben müssen, um ein Layout in diesen Fällen anzupassen.

Die Basis des Layoutsystems ist relative Positionierung, wodurch die Fähigkeit zur Anpassung an geänderte Fenster- und Anzeigebedingungen verbessert wird. Das Layoutsystem verwaltet außerdem die Aushandlung zwischen Steuerelementen, um das Layout zu bestimmen. Die Aushandlung ist ein zweistufiger Vorgang: Zuerst teilt ein Steuerelement seinem übergeordneten Element mit, welche Position und Größe es benötigt, und anschließend teilt das übergeordnete Element dem Steuerelement mit, welcher Raum ihm zur Verfügung steht.

Das Layoutsystem wird untergeordneten Steuerelementen über WPF-Basisklassen verfügbar gemacht. Für allgemeine Layouts wie Raster, Stapeln und Andocken enthält WPF mehrere Layoutsteuerelemente:

- [Canvas](#): Untergeordnete Steuerelemente stellen ihr eigenes Layout bereit.

- **DockPanel**: Untergeordnete Steuerelemente werden an den Rändern des Bereichs ausgerichtet.
- **Grid**: Untergeordnete Steuerelemente werden anhand von Zeilen und Spalten positioniert.
- **StackPanel**: Untergeordnete Steuerelemente werden entweder vertikal oder horizontal gestapelt.
- **VirtualizingStackPanel**: Untergeordnete Steuerelemente werden virtualisiert und auf einer einzelnen Linie angeordnet, die horizontal oder vertikal verläuft.
- **WrapPanel**: Untergeordnete Steuerelemente werden der Reihe nach von links nach rechts angeordnet und, wenn sich in der jeweiligen Zeile mehr Steuerelemente befinden, als der Platz zulässt, ggf. auf die nächste Zeile umbrochen.

Im folgenden Beispiel wird ein **DockPanel**-Objekt verwendet, um ein Layout mit mehreren **TextBox**-Steuerelementen zu erstellen:

XAML

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.LayoutWindow"
  Title="Layout with the DockPanel" Height="143" Width="319">

  <!--DockPanel to layout four text boxes-->
  <DockPanel>
    <TextBox DockPanel.Dock="Top">Dock = "Top"</TextBox>
    <TextBox DockPanel.Dock="Bottom">Dock = "Bottom"</TextBox>
    <TextBox DockPanel.Dock="Left">Dock = "Left"</TextBox>
    <TextBox Background="White">This TextBox "fills" the remaining space.
  </TextBox>
  </DockPanel>

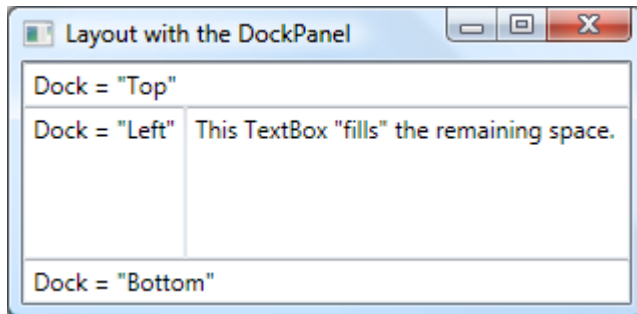
</Window>
```

Das **DockPanel**-Objekt ermöglicht es den untergeordneten **TextBox**-Steuerelementen, ihm Informationen über ihre Anordnung bereitzustellen. Dazu wird von der **DockPanel**-Klasse die angehängte **Dock**-Eigenschaft implementiert, die für die untergeordneten Steuerelemente verfügbar gemacht wird, damit jedes von ihnen eine Andockart festlegen kann.

ⓘ Hinweis

Eine Eigenschaft, die von einem übergeordneten Steuerelement zur Verwendung durch untergeordnete Steuerelemente implementiert wird, ist ein WPF-Konstrukt, das als **angefügte Eigenschaft** bezeichnet wird.

Die folgende Abbildung zeigt das Ergebnis des XAML-Markups aus dem vorangegangenen Beispiel:

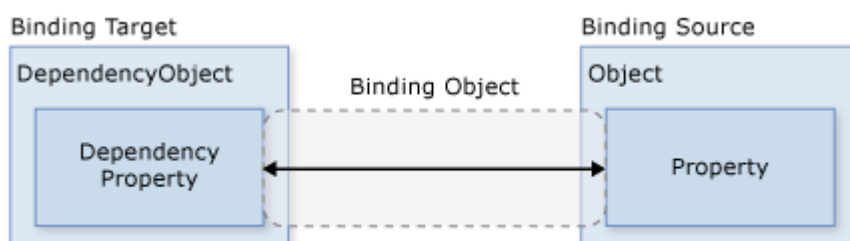


Datenbindung

Die meisten Anwendungen werden erstellt, um Benutzern die Möglichkeit zum Anzeigen und Bearbeiten von Daten bereitzustellen. Bei WPF-Anwendungen wird die Arbeit zum Speichern und Zugreifen auf Daten von Technologien wie Microsoft SQL Server und ADO.NET übernommen. Nachdem auf die Daten zugegriffen wurde und diese in die verwalteten Objekte einer Anwendung geladen wurden, beginnt die harte Arbeit für WPF-Anwendungen. Dies umfasst im Wesentlichen zwei Dinge:

1. Kopieren der Daten aus den verwalteten Objekten in Steuerelemente, wo die Daten angezeigt und bearbeitet werden können.
2. Sicherstellen, dass Änderungen, die über Steuerelemente an den Daten vorgenommen wurden, in die verwalteten Objekte kopiert werden.

Um die Entwicklung von Anwendungen zu vereinfachen, stellt WPF eine Datenbindungs-Engine bereit, mit dem diese Schritte automatisch ausgeführt werden können. Das Kernelement der Datenbindungs-Engine ist die **Binding**, deren Aufgabe es ist, ein Steuerelement (das Bindungsziel) an ein Datenobjekt (die Bindungsquelle) zu binden. Diese Beziehung wird in der folgenden Abbildung veranschaulicht:



Im nächsten Beispiel wird gezeigt, wie ein `TextBox`-Objekt an eine Instanz eines benutzerdefinierten `Person`-Objekts gebunden wird. Die `Person`-Implementierung wird im folgenden Beispielcode dargestellt:

C#

```
namespace SDKSample
{
    class Person
    {
        string name = "No Name";

        public string Name
        {
            get { return name; }
            set { name = value; }
        }
    }
}
```

Im folgenden Markupcode wird das `TextBox`-Objekt an eine Instanz eines benutzerdefinierten `Person`-Objekts gebunden:

XAML

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.DataBindingWindow">

    <!-- Bind the TextBox to the data source (TextBox.Text to Person.Name) -->
    <TextBox Name="personNameTextBox" Text="{Binding Path=Name}" />

</Window>
```

C#

```
using System.Windows; // Window

namespace SDKSample
{
    public partial class DataBindingWindow : Window
    {
        public DataBindingWindow()
        {
            InitializeComponent();

            // Create Person data source
            Person person = new Person();
        }
    }
}
```

```
        // Make data source available for binding
        this.DataContext = person;
    }
}
```

In diesem Beispiel wird die `Person` -Klasse in CodeBehind instanziiert und als Datenkontext für die `DataBindingWindow`-Klasse festgelegt. Im Markupcode wird die `Text` -Eigenschaft des `TextBox` -Steuerelements an die `Person.Name` -Eigenschaft gebunden (über die XAML-Syntax `{Binding ... }`). Dieser XAML-Code weist WPF an, das `TextBox` -Steuerelement an das `Person` -Objekt zu binden, das in der `DataContext` -Eigenschaft des Fensters gespeichert ist.

Die Datenbindungs-Engine von WPF bietet zusätzliche Unterstützung, wozu Validierung, Sortierung, Filterung und Gruppierung gehören. Darüber hinaus wird für Datenbindung die Verwendung von Datenvorlagen unterstützt, um eine benutzerdefinierte Benutzeroberfläche für gebundene Daten zu erstellen, wenn die Benutzeroberfläche nicht geeignet ist, die von den WPF-Standardsteuerelementen angezeigt wird.

Weitere Informationen finden Sie unter [Übersicht über Datenbindung](#).

Grafiken

Mit WPF wird ein umfangreicher, skalierbarer und flexibler Satz von Grafikfeatures eingeführt, die die folgenden Vorteile bieten:

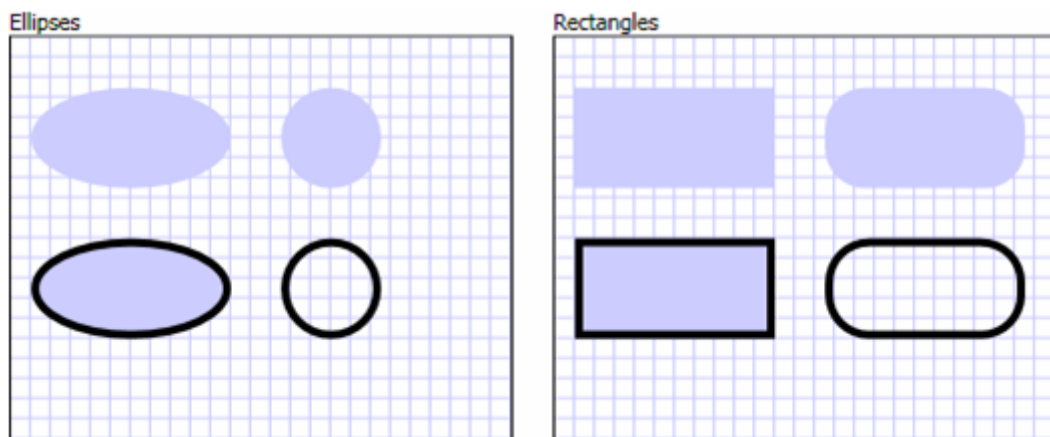
- **Auflösungsunabhängige und geräteunabhängige Grafik.** Die grundlegende Maßeinheit im WPF-Grafiksystem ist das geräteunabhängige Pixel, das, unabhängig von der jeweiligen Bildschirmauflösung, immer 1/96 Zoll entspricht. Diese Maßeinheit bildet die Grundlage für auflösungs- und geräteunabhängiges Rendering. Jedes geräteunabhängige Pixel wird automatisch skaliert, um mit der DPI-Einstellung (Dots Per Inch) des Systems übereinzustimmen, auf dem das Pixel gerendert wird.
- **Höhere Genauigkeit.** Das WPF-Koordinatensystem wird mit Gleitkommazahlen mit doppelter Genauigkeit gemessen. Transformationen und Durchlässigkeitswerte werden ebenfalls mit doppelter Genauigkeit ausgedrückt. WPF unterstützt auch eine breite Farbskala (scRGB) und unterstützt das Verwalten von Eingaben aus unterschiedlichen Farbräumen.
- **Erweiterte Unterstützung für Grafiken und Animationen.** WPF vereinfacht die Grafikprogrammierung, indem es Animationsszenen für Sie verwaltet. Sie müssen

sich keine Gedanken über Szenenverarbeitung, Renderingschleifen und bilineare Interpolation machen. Darüber bietet WPF Treffertest-Unterstützung und vollständige Alpha-Compositing-Unterstützung.

- **Hardwarebeschleunigung.** Das WPF-Grafiksystem nutzt die Vorteile der Grafikkarte, um die CPU-Auslastung zu verringern.

2D-Formen

Zu WPF gehört eine Bibliothek häufig verwendeter vektorbasierter 2D-Formen, etwa Rechtecke und Ellipsen, die in der folgenden Abbildung dargestellt sind:



Eine interessante Fähigkeit von Formen ist, dass sie nicht nur zur Anzeige vorhanden sind, sondern für sie auch viele der Features implementiert sind, die Sie von Steuerelementen erwarten, einschließlich Tastatur- und Mauseingaben. Im folgenden Beispiel wird das [MouseDown](#)-Ereignis einer [Ellipse](#) gezeigt, die verarbeitet wird:

XAML

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.EllipseEventHandlingWindow"
    Title="Click the Ellipse">
    <Ellipse Name="clickableEllipse" Fill="Blue"
    MouseUp="clickableEllipse_MouseUp" />
</Window>
```

C#

```
using System.Windows; // Window, MessageBox
using System.Windows.Input; // MouseButtonEventHandler

namespace SDKSample
{
    public partial class EllipseEventHandlingWindow : Window
```

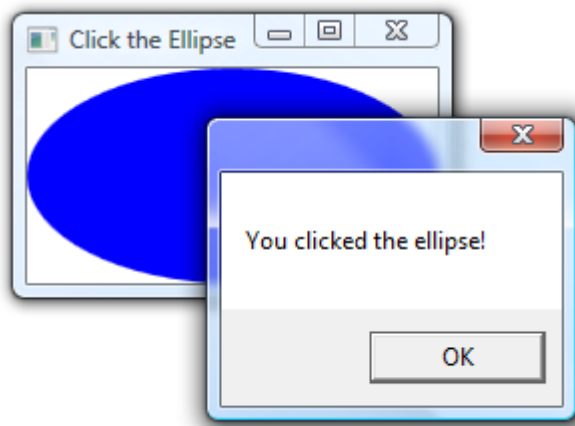
```

{
    public EllipseEventHandlingWindow()
    {
        InitializeComponent();
    }

    void clickableEllipse_MouseUp(object sender, MouseButtonEventArgs e)
    {
        // Display a message
        MessageBox.Show("You clicked the ellipse!");
    }
}

```

In der folgenden Abbildung ist das Ergebnis des vorangehenden Codes dargestellt:



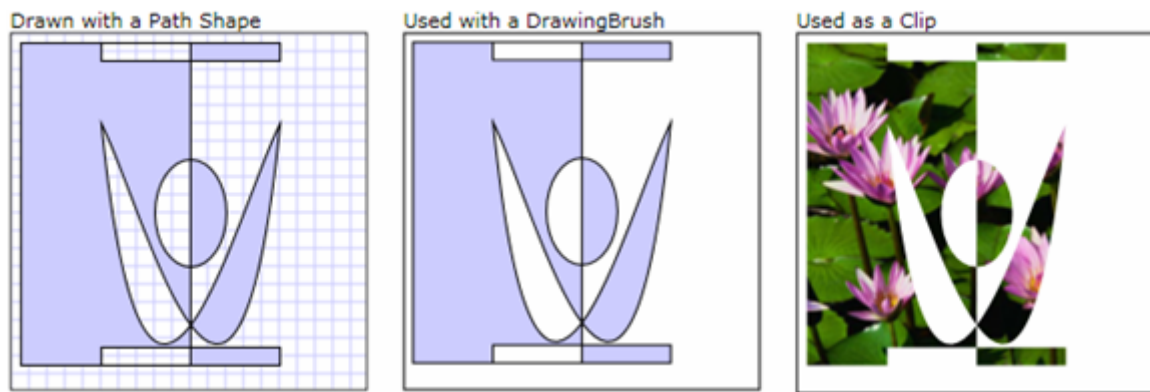
Weitere Informationen finden Sie unter [Übersicht über Formen und die grundlegenden Funktionen zum Zeichnen in WPF](#).

2D-Geometrien

Die von WPF bereitgestellten 2D-Formen decken die grundlegenden Standardformen ab. Möglicherweise müssen Sie jedoch benutzerdefinierte Formen erstellen, um den Entwurf einer angepassten Benutzeroberfläche zu ermöglichen. Zu diesem Zweck stellt WPF Geometrien bereit. In der folgenden Abbildung wird veranschaulicht, wie Geometrien verwendet werden können, um eine benutzerdefinierte Form zu erstellen, die direkt gezeichnet, als Pinsel verwendet oder dazu verwendet werden kann, andere Formen und Steuerelemente auszuschneiden.

Mit [Path](#) -Objekten können geschlossene oder offene Formen, Mehrfachformen und sogar gekrümmte Formen gezeichnet werden.

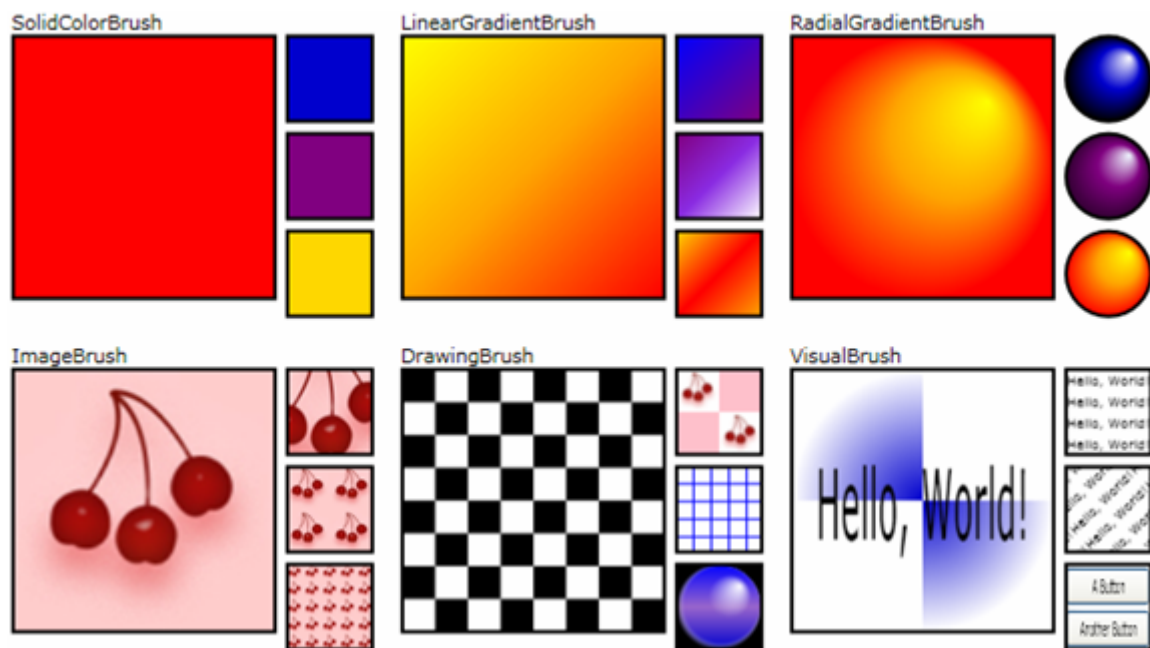
[Geometry](#)-Objekte können zum Ausschneiden, zum Ausführen von Treffertests und zum Rendern von 2D-Grafikdaten verwendet werden.



Weitere Informationen finden Sie unter [Übersicht über die Geometrie](#).

2D-Effekte

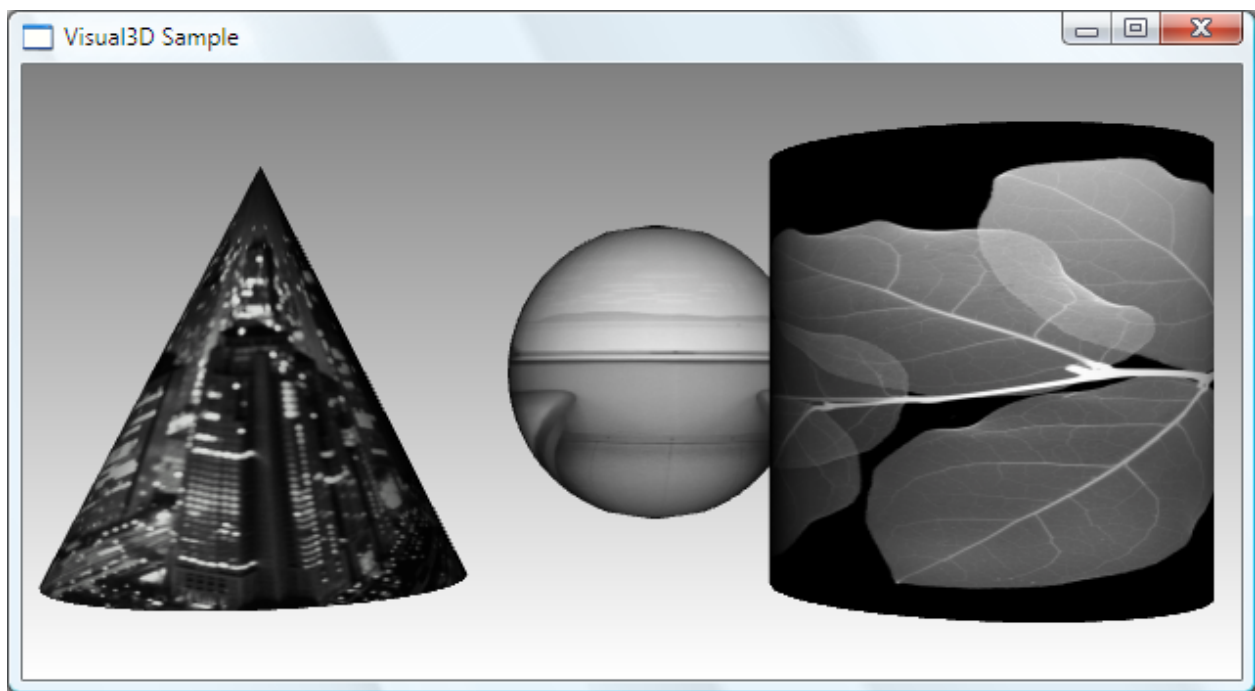
Eine Teilmenge der 2D-Funktionen von WPF umfasst visuelle Effekte wie Farbverläufe, Bitmaps, Zeichnungen, Zeichnen mit Videos, Drehung, Skalierung und Neigung. Diese Effekte werden mithilfe von Pinseln erzielt. In der folgenden Abbildung sind einige Beispiele gezeigt:



Weitere Informationen finden Sie unter [Übersicht über WPF-Pinsel](#).

3D-Rendering

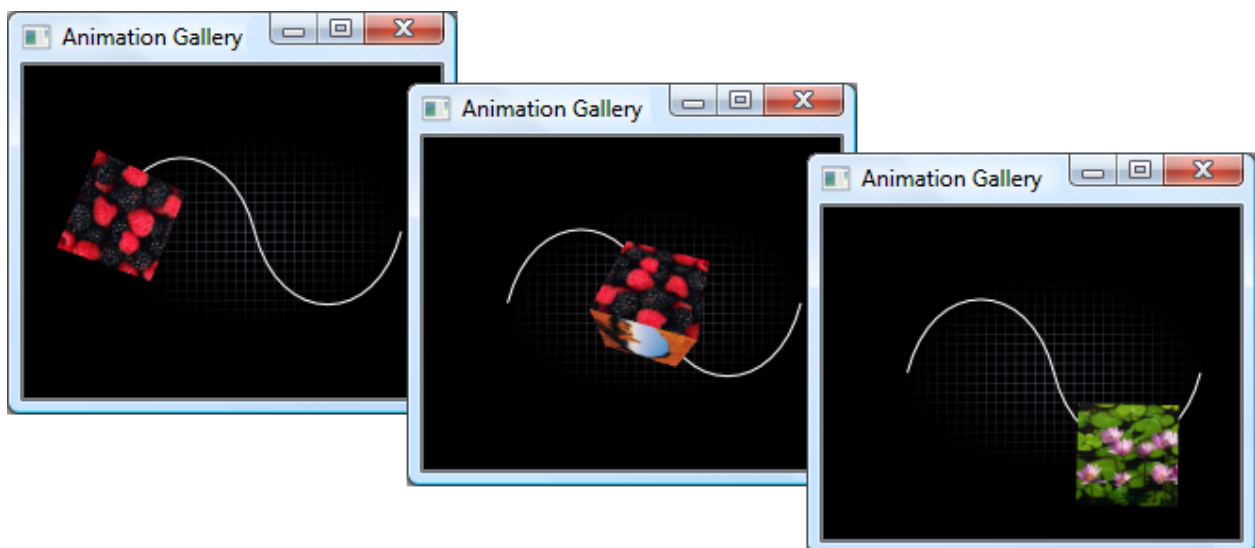
WPF beinhaltet auch 3D-Renderingfunktionen, die mit der 2D-Grafik kombiniert sind, um noch ansprechendere und interessantere Benutzeroberflächen erstellen zu können. Als Beispiel sind in der folgenden Abbildung 2D-Bilder dargestellt, die auf 3D-Formen gerendert wurden:



Weitere Informationen finden Sie unter [Übersicht über 3D-Grafiken](#).

Animation

Die WPF-Animationsunterstützung ermöglicht es Ihnen, Steuerelemente wachsen, bewegen, schütteln sowie ein- und ausblenden zu lassen, um interessante Seitenübergänge zu erstellen, und vieles mehr. Die meisten WPF-Klassen, selbst benutzerdefinierte Klassen, können animiert werden. Die folgende Abbildung zeigt eine einfache Animation in Aktion:



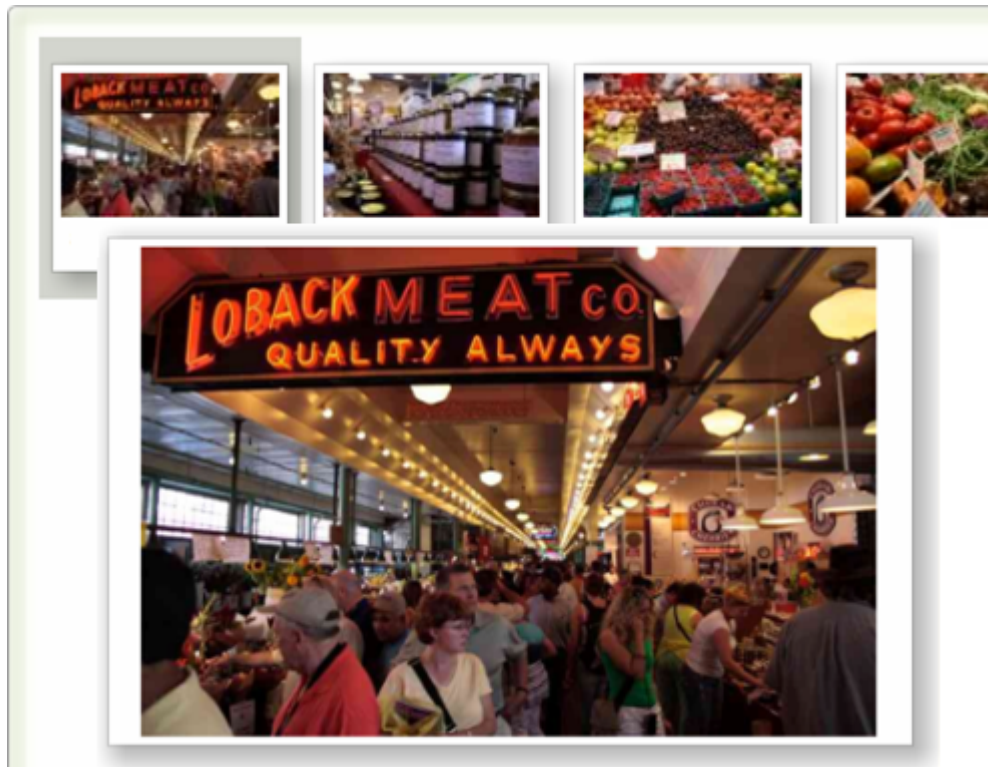
Weitere Informationen finden Sie unter [Übersicht über Animation](#).

Medien

Eine Möglichkeit, Inhalte interessant zu vermitteln, ist die Verwendung audiovisueller Medien. WPF bietet spezielle Unterstützung für Bilder, Video und Audio.

Bilder

In den meisten Anwendungen werden Bilder verwendet, und WPF bietet mehrere Möglichkeiten, Bilder zu verwenden. Die folgende Abbildung zeigt eine Benutzeroberfläche mit einem Listefeld, das Miniaturbilder enthält. Wird eine Miniaturansicht ausgewählt, wird das Bild in voller Größe angezeigt.



Weitere Informationen finden Sie unter [Übersicht über die Bildverarbeitung](#).

Video und Audio

Mit dem [MediaElement](#) -Steuerelement kann sowohl Video als auch Audio wiedergegeben werden, und es ist flexibel genug, um als Grundlage für einen benutzerdefinierten Media Player verwendet zu werden. Mit dem folgenden XAML-Markup wird ein Media Player implementiert:

XAML

```
<MediaElement
  Name="myMediaElement"
  Source="media/wpf.wmv"
  LoadedBehavior="Manual"
  Width="350" Height="250" />
```


Das Fenster in der folgenden Abbildung zeigt das [MediaElement](#)-Steuerelement in Aktion:



Weitere Informationen finden Sie unter [Grafiken und Multimedia](#).

Text und Typografie

Um ein qualitativ hochwertiges Textrendering zu ermöglichen, bietet WPF die folgenden Features:

- Unterstützung für OpenType-Schriftarten
- ClearType-Optimierungen
- Hohe Leistung durch Nutzung von Hardwarebeschleunigung
- Einbinden von Text in Medien, Grafiken und Animationen
- Unterstützung internationaler Schriftarten und Fallbackmechanismen

Zur Veranschaulichung der Textintegration mit Grafiken zeigt die folgende Abbildung die Anwendung von Textdekorationen:

Basic Text Decorations with XAML

The lazy dog ~~The lazy dog~~ The lazy dog The lazy dog

Changing the Color of a Text Decoration with XAML

The lazy dog ~~The lazy dog~~ The lazy dog The lazy dog

Creating Dash Text Decorations with XAML

The lazy dog ~~The lazy dog~~ The lazy dog The lazy dog

Weitere Informationen finden Sie unter [Typografie in WPF](#).

Anpassen von WPF-Apps

Bis zu dieser Stelle haben die WPF-Kernbausteine zur Entwicklung von Anwendungen kennen gelernt. Sie verwenden das Anwendungsmodell, um Anwendungsinhalte, die hauptsächlich aus Steuerelementen bestehen, zu hosten und bereitzustellen. Das WPF-Layoutsystem verwenden Sie, um die Anordnung von Steuerelementen in einer Benutzeroberfläche zu vereinfachen und sicherzustellen, dass die Anordnung bei Änderungen von Fenstergröße und Anzeigeeinstellungen erhalten bleibt. Da die meisten Anwendungen Benutzern ein Bearbeiten von Daten ermöglichen, verwenden Sie Datenbindung, um den Arbeitsaufwand für das Einbinden der Daten in die jeweilige Benutzeroberfläche zu reduzieren. Um die visuelle Darstellung Ihrer Anwendung zu verbessern, verwenden Sie die umfangreiche Grafik-, Animations- und Medienunterstützung, die von WPF bereitgestellt wird.

Häufig reichen die Grundelemente jedoch nicht aus, um eine wirklich herausragende und visuell eindrucksvolle Benutzeroberfläche zu erstellen und zu verwalten. Die Standardsteuerelemente von WPF passen möglicherweise nicht zum gewünschten Erscheinungsbild Ihrer Anwendung. Daten können vielleicht nicht auf die bestmögliche Art angezeigt werden. Der Gesamteindruck Ihrer Anwendung ist eventuell nicht für das standarmäßige Aussehen und Verhalten der Windows-Designs geeignet. In vielerlei Hinsicht erfordert eine Präsentationstechnologie visuelle Erweiterbarkeit ebenso wie jede andere Art von Erweiterbarkeit.

Aus diesem Grund bietet die WPF eine Vielzahl von Mechanismen zum Erzeugen einzigartiger Benutzeroberflächen, wie z. B. ein umfangreiches Inhaltsmodell für Steuerelemente, Trigger, Steuerelement- und Datenvorlagen, Stile, Ressourcen für Benutzeroberflächen, Designs und Skins.

Inhaltsmodell

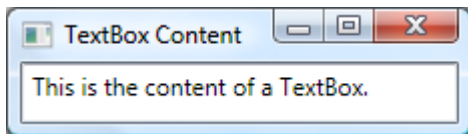
Die meisten WPF-Steuerelemente haben hauptsächlich die Aufgabe, Inhalte anzuzeigen. In WPF werden der Typ und die Anzahl von Elementen, die den Inhalt eines Steuerelements bilden können, als *Inhaltsmodell* des Steuerelements bezeichnet. Einige Steuerelemente können ein einzelnes Element und einen einzelnen Inhaltstyp enthalten. Beispielsweise ist der Inhalt eines [TextBox](#) -Steuerelements ein Zeichenfolgenwert, der der [Text](#) -Eigenschaft zugewiesen ist. Im folgenden Beispiel wird der Inhalt eines [TextBox](#)-Steuerelements festgelegt:

XAML

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.TextBoxContentWindow"
  Title="TextBox Content">

  <TextBox Text="This is the content of a TextBox." />
</Window>
```

Die folgende Abbildung zeigt das Ergebnis:



Andere Steuerelemente können dagegen mehrere Elemente verschiedener Inhaltstypen enthalten. Der Inhalt eines [Button](#)-Steuerelements, der durch die [Content](#)-Eigenschaft angegeben ist, kann eine Vielzahl von Elementen enthalten, etwa Layoutsteuerelemente, Text, Bildern und Formen. Das folgende Beispiel zeigt ein [Button](#)-Steuerelement mit Inhalt, zu dem ein [DockPanel](#)-, ein [Label](#)-, ein [Border](#)- und ein [MediaElement](#)-Objekt gehören:

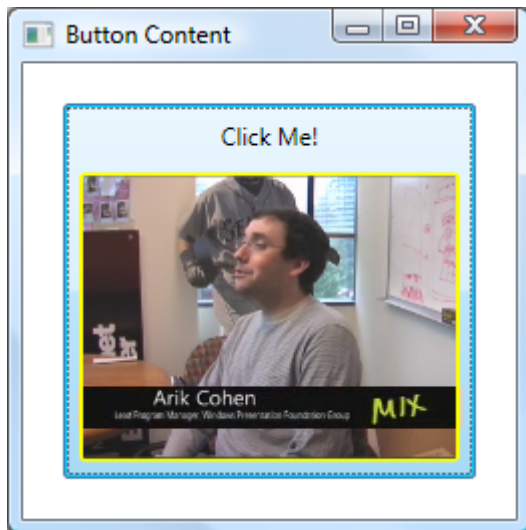
XAML

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.ButtonContentWindow"
  Title="Button Content">

  <Button Margin="20">
    <!-- Button Content -->
    <DockPanel Width="200" Height="180">
      <Label DockPanel.Dock="Top" HorizontalAlignment="Center">Click Me!
    </Label>
    <Border Background="Black" BorderBrush="Yellow" BorderThickness="2"
      CornerRadius="2" Margin="5">
      <MediaElement Source="media/wpf.wmv" Stretch="Fill" />
    </Border>
  </Button>
```

```
</Border>
</DockPanel>
</Button>
</Window>
```

In der folgenden Abbildung ist der Inhalt dieser Schaltfläche dargestellt:



Weitere Informationen zu den Inhaltstypen, die von den verschiedenen Steuerelementen unterstützt werden, finden Sie unter [WPF-Inhaltsmodell](#).

Auslöser

Obwohl der Hauptzweck von XAML-Markup in der Implementierung der Darstellung einer Anwendung besteht, lassen sich mit XAML auch einige Aspekte des Verhaltens einer Anwendung implementieren. Ein Beispiel ist die Verwendung von Triggern, um die Darstellung einer Anwendung anhand von Benutzeraktionen zu ändern. Weitere Informationen finden Sie unter [Stile und Vorlagen](#).

Steuerelementvorlagen

Die standardmäßigen Benutzeroberflächen für WPF-Steuerelemente werden üblicherweise aus anderen Steuerelementen und Formen erstellt. Beispielsweise besteht ein [Button](#) -Steuerelement aus einem [ButtonChrome](#) - und einem [ContentPresenter](#) -Steuerelement. Das [ButtonChrome](#) -Steuerelement stellt die Standarddarstellung der Schaltfläche bereit, während das [ContentPresenter](#) -Steuerelement den Inhalt der Schaltfläche anzeigt, der in der [Content](#) -Eigenschaft angegeben ist.

Manchmal passt die Standarddarstellung eines Steuerelements nicht zum Gesamterscheinungsbild einer Anwendung. In diesem Fall können Sie ein [ControlTemplate](#) -Objekt verwenden, um die Darstellung der Benutzeroberfläche des Steuerelements anzupassen, ohne dessen Inhalt und Verhalten zu ändern.

Im folgenden Beispiel wird gezeigt, wie die Darstellung eines [Button](#)-Steuerelements durch Verwenden einer [ControlTemplate](#) geändert werden kann:

XAML

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.ControlTemplateButtonWindow"
  Title="Button with Control Template" Height="158" Width="290">

  <!-- Button using an ellipse -->
  <Button Content="Click Me!" Click="button_Click">
    <Button.Template>
      <ControlTemplate TargetType="{x:Type Button}">
        <Grid Margin="5">
          <Ellipse Stroke="DarkBlue" StrokeThickness="2">
            <Ellipse.Fill>
              <RadialGradientBrush Center="0.3,0.2" RadiusX="0.5"
RadiusY="0.5">
                <GradientStop Color="Azure" Offset="0.1" />
                <GradientStop Color="CornflowerBlue" Offset="1.1" />
              </RadialGradientBrush>
            </Ellipse.Fill>
          </Ellipse>
          <ContentPresenter Name="content" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
        </Grid>
      </ControlTemplate>
    </Button.Template>
  </Button>

</Window>
```

C#

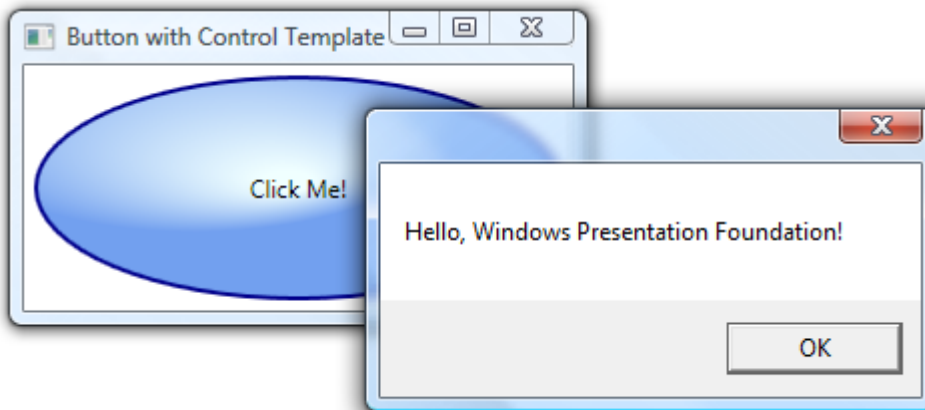
```
using System.Windows; // Window, RoutedEventArgs, MessageBox

namespace SDKSample
{
    public partial class ControlTemplateButtonWindow : Window
    {
        public ControlTemplateButtonWindow()
        {
            InitializeComponent();
        }

        void button_Click(object sender, RoutedEventArgs e)
        {
            // Show message box when button is clicked
            MessageBox.Show("Hello, Windows Presentation Foundation!");
        }
    }
}
```

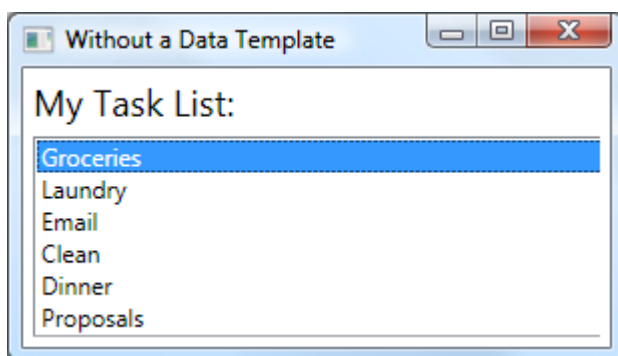
```
}  
}  
}
```

In diesem Beispiel wurde die Standardbenutzeroberfläche der Schaltfläche durch ein [Ellipse](#)-Steuerelement ersetzt, das einen dunkelblauen Rand hat und über ein [RadialGradientBrush](#)-Steuerelement gefüllt wird. Das [ContentPresenter](#)-Steuerelement zeigt den Inhalt von [Button](#) an, „Hier klicken!“. Wenn auf den [Button](#) geklickt wird, wird das [Click](#)-Ereignis trotzdem als Teil des Standardverhaltens des [Button](#)-Steuerelements trotzdem ausgelöst. Das Ergebnis ist in der folgenden Abbildung dargestellt:



Datenvorlagen

Während Sie mit einer Steuerelementvorlage die Darstellung eines Steuerelements angeben können, können Sie mit einer Datenvorlage die Darstellung des Inhalts eines Steuerelements angeben. Datenvorlagen werden häufig dazu verwendet, die Anzeige gebundener Daten zu verbessern. Die folgende Abbildung zeigt die Standarddarstellung für ein [ListBox](#)-Steuerelement, das an eine Auflistung von [Task](#)-Objekten gebunden ist, wobei jede Aufgabe einen Namen, eine Beschreibung und eine Priorität hat:



Die Standarddarstellung entspricht dem, was Sie von einem [ListBox](#)-Steuerelement erwarten. Allerdings enthält die Standarddarstellung jeder Aufgabe nur den Aufgabennamen. Um den Aufgabennamen, die Beschreibung und die Priorität anzuzeigen, muss die Standarddarstellung der gebundenen Listenelemente des [ListBox](#) -

Steuerelements über ein [DataTemplate](#)-Objekt geändert werden. Im folgenden XAML-Code wird ein solches [DataTemplate](#)-Objekt definiert, das über das [ItemTemplate](#)-Attribut auf jede Aufgabe angewendet wird:

XAML

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.DataTemplateWindow"
  Title="With a Data Template">
  <Window.Resources>
    <!-- Data Template (applied to each bound task item in the task
collection) -->
    <DataTemplate x:Key="myTaskTemplate">
      <Border Name="border" BorderBrush="DarkSlateBlue" BorderThickness="2"
        CornerRadius="2" Padding="5" Margin="5">
        <Grid>
          <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
          </Grid.RowDefinitions>
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition />
          </Grid.ColumnDefinitions>
          <TextBlock Grid.Row="0" Grid.Column="0" Padding="0,0,5,0"
Text="Task Name:"/>
          <TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding
Path=TaskName}"/>
          <TextBlock Grid.Row="1" Grid.Column="0" Padding="0,0,5,0"
Text="Description:"/>
          <TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding
Path=Description}"/>
          <TextBlock Grid.Row="2" Grid.Column="0" Padding="0,0,5,0"
Text="Priority:"/>
          <TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding
Path=Priority}"/>
        </Grid>
      </Border>
    </DataTemplate>
  </Window.Resources>

  <!-- UI -->
  <DockPanel>
    <!-- Title -->
    <Label DockPanel.Dock="Top" FontSize="18" Margin="5" Content="My Task
List:"/>

    <!-- Data template is specified by the ItemTemplate attribute -->
    <ListBox
      ItemsSource="{Binding}"
```

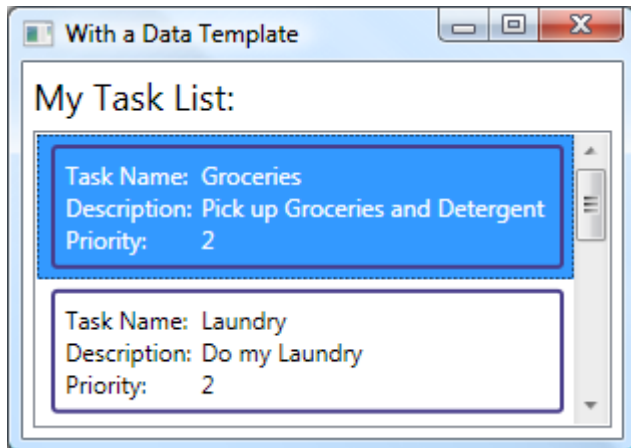
```

        ItemTemplate="{StaticResource myTaskTemplate}"
        HorizontalContentAlignment="Stretch"
        IsSynchronizedWithCurrentItem="True"
        Margin="5,0,5,5" />

</DockPanel>
</Window>

```

Die folgende Abbildung zeigt das Ergebnis dieses Codes:



Beachten Sie, dass das Verhalten und die Gesamtdarstellung des [ListBox](#)-Steuerelements beibehalten wurden. Lediglich die Darstellung der vom Listenfeld angezeigten Inhalte wurde geändert.

Weitere Informationen finden Sie unter [Übersicht über Datenvorlagen](#).

Stile

Stile ermöglichen Entwicklern und Designern die Standardisierung auf ein bestimmtes Erscheinungsbild ihres Produkts. WPF stellt ein solides Formatmodell bereit, dessen Grundlage das [Style](#)-Element bildet. Im folgenden Beispiel wird eine Formatvorlage erstellt, mit der die Hintergrundfarbe für jedes [Button](#)-Steuerelement in einem Fenster auf `Orange` festgelegt wird:

XAML

```

<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.StyleWindow"
    Title="Styles">

    <Window.Resources>
        <!-- Style that will be applied to all buttons for this window -->
        <Style TargetType="{x:Type Button}">
            <Setter Property="Background" Value="Orange" />
        </Style>
    </Window.Resources>

```



```

        <Setter Property="BorderBrush" Value="Crimson" />
        <Setter Property="FontSize" Value="20" />
        <Setter Property="FontWeight" Value="Bold" />
        <Setter Property="Margin" Value="5" />
    </Style>
</Window.Resources>
<StackPanel>

    <!-- This button will have the style applied to it -->
    <Button>Click Me!</Button>

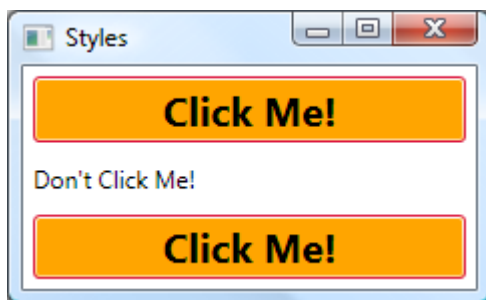
    <!-- This label will not have the style applied to it -->
    <Label>Don't Click Me!</Label>

    <!-- This button will have the style applied to it -->
    <Button>Click Me!</Button>

</StackPanel>
</Window>

```

Da sich dieser Stil auf alle **Button**-Steuerelemente bezieht, wird er automatisch auf alle Schaltflächen im Fenster angewendet. Dies ist in der folgenden Abbildung zu sehen:



Weitere Informationen finden Sie unter [Stile und Vorlagen](#).

Ressourcen

Die Steuerelemente in einer Anwendung sollten die gleiche Darstellung haben, wozu alles von Schriftarten und Hintergrundfarben bis zu Steuerelementvorlagen, Datenvorlagen und Stilen gehören kann. Über die WPF-Unterstützung für Benutzeroberflächenressourcen können diese Ressourcen in einem einzigen Speicherort kapseln, um sie wiederzuverwenden.

Im folgenden Beispiel wird eine gemeinsame Hintergrundfarbe festgelegt, die für ein **Button**- und ein **Label**-Steuerelement verwendet wird:

XAML

```

<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

```

```

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="SDKSample.ResourcesWindow"
Title="Resources Window">

<!-- Define window-scoped background color resource -->
<Window.Resources>
  <SolidColorBrush x:Key="defaultBackground" Color="Red" />
</Window.Resources>

<!-- Button background is defined by window-scoped resource -->
<Button Background="{StaticResource defaultBackground}">One
Button</Button>

<!-- Label background is defined by window-scoped resource -->
<Label Background="{StaticResource defaultBackground}">One Label</Label>
</Window>

```

In diesem Beispiel wird mit dem `Window.Resources` -Eigenschaftenelements eine Ressource für die Hintergrundfarbe implementiert. Diese Ressource ist für alle untergeordneten Elemente des `Window`-Elements verfügbar. Es gibt eine Vielzahl von Ressourcenbereichen, von denen einige nachfolgend in der Reihenfolge aufgeführt sind, in der sie aufgelöst werden:

1. Ein einzelnes Steuerelement (über die geerbte `System.Windows.FrameworkElement.Resources` -Eigenschaft)
2. Ein `Window` - oder `Page` -Element (ebenfalls über die geerbte `System.Windows.FrameworkElement.Resources` -Eigenschaft)
3. Ein `Application` -Element (über die `System.Windows.Application.Resources` -Eigenschaft)

Durch die Vielzahl von Bereichen erhalten Sie Flexibilität in Bezug auf die Art, mit der Sie Ihre Ressourcen definieren und freigeben.

Anstatt Ihre Ressourcen direkt mit einem bestimmten Bereich zu verknüpfen, können Sie eine oder mehrere Ressourcen über ein separates `ResourceDictionary` -Element verpacken, auf das in anderen Teilen einer Anwendung verwiesen werden kann. Im folgenden Beispiel wird etwa eine Standardhintergrundfarbe in einem Ressourcenwörterbuch definiert:

XAML

```

<ResourceDictionary
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <!-- Define background color resource -->

```

```
<SolidColorBrush x:Key="defaultBackground" Color="Red" />

<!-- Define other resources -->
</ResourceDictionary>
```

Im folgenden Beispiel wird auf das Ressourcenwörterbuch, das im vorherigen Beispiel definiert wurde, verwiesen, sodass es überall in einer Anwendung freigegeben ist:

XAML

```
<Application
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.App">

  <Application.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="BackgroundColorResources.xaml"/>
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

Ressourcen und Ressourcenwörterbücher bilden die Grundlage der WPF-Unterstützung für Designs und Skins.

Weitere Informationen finden Sie unter [Ressourcen](#).

Benutzerdefinierte Steuerelemente

Obwohl WPF umfangreiche Unterstützung für Anpassungen bietet, kann es Fälle geben, in denen die vorhandenen WPF-Steuerelemente den Anforderungen Ihrer Anwendung oder denen der Benutzer nicht genügen. Dies kann auftreten, wenn Folgendes zutrifft:

- Die von Ihnen gewünschte Benutzeroberfläche kann nicht erstellt werden, indem das Aussehen und Verhalten vorhandener WPF-Implementierungen angepasst wird.
- Das von Ihnen gewünschte Verhalten wird von vorhandenen WPF-Implementierungen nicht (oder nicht einfach) unterstützt.

An diesem Punkt können Sie jedoch eines der drei WPF-Modelle nutzen, um ein neues Steuerelement zu erstellen. Jedes Modell ist für ein bestimmtes Szenario vorgesehen und erfordert, dass Ihr benutzerdefiniertes Steuerelement aus einer bestimmten WPF-Basisklasse abgeleitet wird. Die drei Modelle sind hier aufgeführt:

- **Benutzersteuerelementmodell** Ein benutzerdefiniertes Steuerelement wird aus [UserControl](#) abgeleitet und besteht aus mindestens einem anderen Steuerelement.
- **Steuerelementmodell** Ein benutzerdefiniertes Steuerelement wird aus [Control](#) abgeleitet und dazu verwendet, eine Implementierung zu erstellen, deren Verhalten mithilfe von Vorlagen von ihrer Darstellung getrennt wird, so wie beim Großteil der WPF-Steuerelemente. Durch Ableiten aus [Control](#) erhalten Sie für ein Erstellen einer benutzerdefinierten Benutzeroberfläche mehr Freiheit als mit Benutzersteuerelementen, dies kann jedoch höheren Aufwand erfordern.
- **Frameworkelementmodell.** Ein benutzerdefiniertes Steuerelement wird aus [FrameworkElement](#) abgeleitet, wenn seine Darstellung durch benutzerdefinierte Renderinglogik (nicht durch Vorlagen) definiert ist.

Im folgenden Beispiel wird ein benutzerdefiniertes numerisches „Nach oben/Nach unten“-Steuerelement gezeigt, das aus [UserControl](#) abgeleitet ist:

XAML

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.NumericUpDown">

  <Grid>

    <Grid.RowDefinitions>
      <RowDefinition/>
      <RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <!-- Value text box -->
    <Border BorderThickness="1" BorderBrush="Gray" Margin="2"
Grid.RowSpan="2"
      VerticalAlignment="Center" HorizontalAlignment="Stretch">
      <TextBlock Name="valueText" Width="60" TextAlignment="Right"
Padding="5"/>
    </Border>

    <!-- Up/Down buttons -->
    <RepeatButton Name="upButton" Click="upButton_Click" Grid.Column="1"
      Grid.Row="0">Up</RepeatButton>
    <RepeatButton Name="downButton" Click="downButton_Click" Grid.Column="1"
      Grid.Row="1">Down</RepeatButton>

  </Grid>
```

```
</UserControl>
```

C#

```
using System; // EventArgs
using System.Windows; // DependencyObject,
DependencyPropertyChangedEventArgs,
// FrameworkPropertyMetadata, PropertyChangedCallback,
// RoutedPropertyChangedEventArgs
using System.Windows.Controls; // UserControl

namespace SDKSample
{
    public partial class NumericUpDown : UserControl
    {
        // NumericUpDown user control implementation
    }
}
```

Im nächsten Beispiel ist der XAML-Code dargestellt, der dazu erforderlich ist, das Benutzersteuerelement in ein [Window](#)-Element zu integrieren:

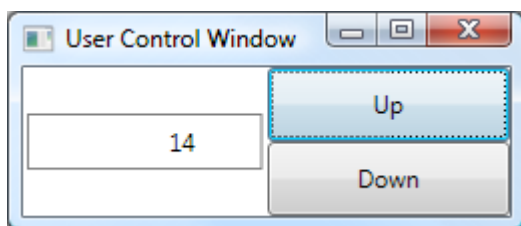
XAML

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.UserControlWindow"
    xmlns:local="clr-namespace:SDKSample"
    Title="User Control Window">

    <!-- Numeric Up/Down user control -->
    <local:NumericUpDown />

</Window>
```

Die folgende Abbildung zeigt das in einem [Window](#)-Element gehostete `NumericUpDown`-Steuerelement:



Weitere Informationen zu benutzerdefinierten Steuerelementen finden Sie unter [Übersicht über das Erstellen von Steuerelementen](#).

Bewährte Methoden für WPF

Wie bei jeder Entwicklungsplattform kann das gewünschte Ergebnis mit WPF auf verschiedene Arten erreicht werden. Um sicherzugehen zu können, dass Ihre WPF-Anwendungen die geforderte Benutzerfreundlichkeit bereitstellen und grundsätzlich den Ansprüche der Zielgruppe entsprechen, gibt es empfohlene bewährte Methoden in Bezug auf Barrierefreiheit, Globalisierung, Lokalisierung und Leistung. Weitere Informationen finden Sie unter

- [Bedienungshilfen](#)
- [Übersicht über WPF-Globalisierung und -Lokalisierung](#)
- [Optimieren der WPF-Anwendungsleistung](#)
- [WPF-Sicherheit](#)

Nächste Schritte

In diesem Artikel wurden die Schlüsselfeatures von WPF erläutert. Jetzt können Sie Ihre erste WPF-App erstellen.

[Exemplarische Vorgehensweise: Meine erste WPF-Desktopanwendung](#)

Weitere Informationen

- [Erste Schritte mit WPF](#)
- [Windows Presentation Foundation](#)
- [Ressourcen für die WPF-Community](#)

Erste Schritte (WPF)


Artikel • 18.10.2023

Windows Presentation Foundation (WPF) ist ein Benutzeroberflächen-Framework, mit dem Desktopclientanwendungen erstellt werden können. Die WPF-Entwicklungsplattform unterstützt eine breite Palette an Anwendungsentwicklungsfeatures, darunter ein Anwendungsmodell, Ressourcen, Steuerelemente, Grafik, Layout, Datenbindung, Dokumente und Sicherheit. Es handelt sich um eine Teilmenge von .NET Framework. Wenn Sie also bisher Anwendungen mit .NET Framework mithilfe von ASP.NET oder Windows Forms erstellt haben, sollte Ihnen die Programmiererfahrung vertraut sein. WPF verwendet XAML (Extensible Application Markup Language), um ein deklaratives Modell für die Anwendungsprogrammierung bereitzustellen. Dieser Abschnitt enthält Themen, die Sie mit WPF bekannt machen und Ihnen den Einstieg erleichtern.

Wie sollte ich vorgehen?

Aktion	Gehe zu
Ich möchte direkt beginnen...	Exemplarische Vorgehensweise: Walkthrough: My first WPF desktop application (Exemplarische Vorgehensweise: Meine erste WPF-Desktopanwendung)
Wie wird die Anwendungsbenutzeroberfläche entworfen?	Entwerfen von XAML-Code in Visual Studio
Neu bei .NET?	Übersicht über .NET Framework Erste Schritte mit Visual C# und Visual Basic
Weitere Informationen über WPF...	Einführung in WPF in Visual Studio XAML in WPF Steuerelemente Übersicht zur Datenbindung
Sind Sie ein Windows Forms-Entwickler?	Windows Forms-Steuerelemente und entsprechende WPF-Steuerelemente Interaktion zwischen WPF und Windows Forms

Siehe auch

- [Klassenbibliothek](#)
- [Anwendungsentwicklung](#)
- [.NET Framework Developer Center](#) 

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET Desktop feedback

The .NET Desktop documentation is open source. Provide feedback here.



[Open a documentation issue](#)



[Provide product feedback](#)

Anwendungsentwicklung

Artikel • 04.05.2023

Windows Presentation Foundation (WPF) ist ein Präsentationsframework, mit dem die folgenden Anwendungstypen entwickelt werden können:

- **Eigenständige Anwendungen:** Herkömmliche Windows-Anwendungen, die als ausführbare Assemblys erstellt, auf dem Clientcomputer installiert und von dort ausgeführt werden.
- **XAML-Browseranwendungen (XAML Browser Applications, XBAPs):** Aus Navigationsseiten bestehende Anwendungen, die als ausführbare Assemblys erstellt und von Webbrowsern wie Microsoft Internet Explorer oder Mozilla Firefox gehostet werden.
- **Benutzerdefinierte Steuerelementbibliotheken:** Nicht ausführbare Assemblys mit wiederverwendbaren Steuerelementen.
- **Klassenbibliotheken:** Nicht ausführbare Assemblys mit wiederverwendbaren Klassen.

ⓘ Hinweis

Es wird dringend davon abgeraten, WPF-Typen in einem Windows-Dienst zu verwenden. Diese Funktionen funktionieren in einem Windows-Dienst möglicherweise nicht wie erwartet.

Um diese Anwendungen zu erstellen, implementiert WPF eine Reihe von Diensten. Dieses Thema bietet eine Übersicht über diese Dienste sowie Verweise auf weitere Informationen.

Anwendungsverwaltung

Ausführbare WPF-Anwendungen erfordern im Allgemeinen eine Reihe wesentlicher Funktionen, die unter anderem folgende Vorgänge ermöglichen:

- Erstellen und Verwalten einer allgemeinen Anwendungsinfrastruktur (einschließlich der Erstellung einer Einstiegspunktmethode und einer Windows-Meldungsschleife zum Empfangen von System- und Eingabenachrichten)
- Verfolgen und Beeinflussen der Lebensdauer einer Anwendung

- Befehlszeilenparameter abrufen und verarbeiten
- Freigeben von Eigenschaften für den Anwendungsbereich und Ressourcen der Benutzeroberfläche
- Erkennen und Verarbeiten von Ausnahmefehlern (unbehandelten Ausnahmen)
- Zurückgeben von Exitcodes
- Fenster in eigenständigen Anwendungen verwalten
- Verfolgen der Navigation in XAML-Browseranwendungen (XBAPs) und eigenständigen Anwendungen mit Navigationsfenstern und Frames

Diese Funktionen werden von der [Application](#)-Klasse implementiert, die Sie Ihren Anwendungen mit einer *Anwendungsdefinition* hinzufügen.

Weitere Informationen finden Sie unter [Übersicht über die Anwendungsverwaltung](#).

WPF-Anwendungsressource, Inhalts- und Datendateien

WPF erweitert die in Microsoft .NET Framework verfügbare Basisunterstützung für eingebettete Ressourcen und unterstützt drei weitere Arten von nicht ausführbaren Datendateien: Ressourcen, Inhalt und Daten. Weitere Informationen finden Sie unter [WPF-Anwendungsressource, Inhalts- und Datendateien](#).

Ein wesentlicher Faktor der Unterstützung von nicht ausführbaren WPF-Datendateien ist die Möglichkeit, diese mit einem eindeutigen URI zu identifizieren und zu laden. Weitere Informationen finden Sie unter [Paket-URI in WPF](#).

Fenster und Dialogfelder

Benutzer interagieren über Fenster mit eigenständigen WPF-Anwendungen. Fenster dienen dazu, Anwendungsinhalte zu hosten und Anwendungsfunktionen, über die Benutzer mit diesen Inhalten interagieren können, verfügbar zu machen. In WPF werden Fenster von der [Window](#)-Klasse gekapselt, die folgende Vorgänge unterstützt:

- Erstellen und Anzeigen von Fenstern
- Einrichten von Beziehungen „Besitzer/zum Besitzer gehöriges Fenster“

- Konfigurieren der Fensterdarstellung (z. B. Größe, Position, Symbole, Titelleistentext und Rahmen)
- Verfolgen und Beeinflussen der Lebensdauer eines Fensters

Weitere Informationen finden Sie unter [Übersicht über WPF-Fenster](#).

[Window](#) unterstützt die Erstellung einer besonderen Art von Fenster, nämlich von Dialogfeldern. Es können sowohl modale als auch nicht modale Dialogfelder (Dialogfelder ohne Modus) erstellt werden.

Aus Gründen der Benutzerfreundlichkeit, der Wiederverwendbarkeit und einer anwendungsübergreifend konsistenten Benutzeroberfläche stellt WPF drei der gängigen Windows-Dialogfelder zur Verfügung: [OpenFileDialog](#), [SaveFileDialog](#) und [PrintDialog](#).

Ein Meldungsfeld ist eine besondere Art von Dialogfeld, in dem Benutzern wichtige Informationen angezeigt und einfache Ja/Nein/OK/Abbrechen-Fragen gestellt werden. Sie verwenden die [MessageBox](#)-Klasse zum Erstellen und Anzeigen von Meldungsfeldern.

Weitere Informationen finden Sie unter [Übersicht über Dialogfelder](#).

Navigation

WPF unterstützt die Navigation im Webstil über Seiten ([Page](#)) und Links ([Hyperlink](#)). Die Navigation kann unter anderem auf folgende Weise implementiert werden:

- Eigenständige Seiten, die in einem Webbrowser gehostet werden
- In eine XBAP kompilierte Seiten, die in einem Webbrowser gehostet werden
- Als eigenständige Anwendung kompilierte Seiten, die von einem Navigationsfenster ([NavigationWindow](#)) gehostet werden
- Von einem Frame gehostete Seiten ([Frame](#)), die in einer eigenständigen Seite oder in einer als XBAP oder eigenständige Anwendung kompilierten Seite gehostet werden können

Zur Unterstützung der Navigation implementiert WPF folgende Elemente:

- [NavigationService](#), die freigegebene Navigations-Engine für die Verarbeitung von Navigationsanforderungen, das von [Frame](#), [NavigationWindow](#) und XBAPs zur Unterstützung der Navigation innerhalb von Anwendungen verwendet wird
- Navigationsmethoden zur Einleitung der Navigation

- Navigationsereignisse zum Verfolgen und Beeinflussen der Navigationslebensdauer
- Aufzeichnung von Vorwärts- und Rückwärtsnavigation mithilfe eines „Journals“, das auch überprüft und bearbeitet werden kann

Weitere Informationen finden Sie unter [Übersicht über die Navigation](#).

Darüber hinaus unterstützt WPF die strukturierte Navigation. Die strukturierte Navigation kann zum Aufruf einer oder mehrerer Seiten verwendet werden, die Daten auf strukturierte und vorhersagbare Weise in Übereinstimmung mit den aufrufenden Funktionen zurückgeben. Diese Funktion ist abhängig von der [PageFunction<T>](#)-Klasse, die unter [Übersicht über die strukturierte Navigation](#) genauer beschrieben wird. [PageFunction<T>](#) vereinfacht zudem die Erstellung komplexer Navigationstopologien, die unter [Übersicht über Navigationstopologien](#) vorgestellt werden.

Hosting

XBAPs können in Microsoft Internet Explorer oder Firefox gehostet werden. Bei jedem Hostmodell sind spezifische Punkte und Einschränkungen zu beachten, die unter [Hosten](#) erläutert werden.

Erstellen und Bereitstellen

Einfache WPF-Anwendungen können über eine Eingabeaufforderung mithilfe von Befehlszeilencompilern erstellt werden. WPF kann jedoch auch in Visual Studio integriert werden und bietet dann zusätzliche Funktionen, die den Entwicklungs- und Erstellungsprozess vereinfachen. Weitere Informationen finden Sie unter [Erstellen einer WPF-Anwendung](#).

Je nach Art der erstellten Anwendung können Sie eine oder mehrere Bereitstellungsoptionen wählen. Weitere Informationen finden Sie unter [Bereitstellen von WPF-Anwendungen](#).

Verwandte Themen

Titel	BESCHREIBUNG
Übersicht über die Anwendungsverwaltung	Hier finden Sie eine Übersicht über die Application -Klasse. Erörtert wird u.a. die Verwaltung von Anwendungslebensdauer, Fenstern, Anwendungsressourcen und Navigation.

Titel	BESCHREIBUNG
Windows in WPF	Hier finden Sie Informationen zur Verwaltung von Fenstern in der Anwendung. Erörtert wird u.a. die Verwendung der Window -Klasse und von Dialogfeldern.
Übersicht über die Navigation	Hier finden Sie eine Übersicht über die Verwaltung der Navigation zwischen Seiten der Anwendung.
Hosting	Hier finden Sie eine Übersicht über XAML-Browseranwendungen (XBAPs).
Erstellen und Bereitstellen	Hier wird beschrieben, wie Sie die WPF-Anwendung erstellen und bereitstellen.
Einführung in WPF in Visual Studio	Hier werden die wichtigsten Funktionen von WPF beschrieben.
Exemplarische Vorgehensweise: Walkthrough: My first WPF desktop application (Exemplarische Vorgehensweise: Meine erste WPF-Desktopanwendung)	In dieser exemplarischen Vorgehensweise wird gezeigt, wie Sie eine WPF-Anwendung mit Seitennavigation, Layout, Steuerelementen, Bildern, Stilen und Bindung erstellen.

Erweitert (Windows Presentation Foundation)

Artikel • 18.10.2023

Dieser Abschnitt beschreibt einige der erweiterten Bereiche in WPF.

In diesem Abschnitt

[WPF-Architektur](#)

[XAML in WPF](#)

[Basiselementklassen](#)

[Elementstruktur und Serialisierung](#)

[WPF-Eigenschaftensystem](#)

[Ereignisse in WPF](#)

[Eingabe](#)

[Drag & Drop](#)

[Ressourcen](#)

[Dokumente](#)

[Globalisierung und Lokalisierung](#)

[Layout](#)

[Aus WPF zu System.Xaml migrierte Typen](#)

[Migration und Interoperabilität](#)

[Leistung](#)

[Threadmodell](#)

[Referenz zur nicht verwalteten WPF-API](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET Desktop feedback

The .NET Desktop documentation is open source. Provide feedback here.

 [Open a documentation issue](#)

 [Provide product feedback](#)

Steuerelemente

Artikel • 06.03.2024

Windows Presentation Foundation (WPF) beinhaltet viele gängige Benutzeroberflächenkomponenten, die in nahezu jeder Windows-Anwendung zum Einsatz kommen, beispielsweise [Button](#), [Label](#), [TextBox](#), [Menu](#) und [ListBox](#). In der Vergangenheit wurden diese Objekte als Steuerelemente bezeichnet. Während im WPF-SDK weiterhin der Begriff „Control“ (Steuerelement) für jede Klasse verwendet wird, die ein sichtbares Objekt in einer Anwendung repräsentiert, ist es wichtig zu beachten, dass eine Klasse nicht notwendigerweise von der [Control](#)-Klasse erben muss, um eine sichtbare Präsenz zu haben. Klassen, die von der [Control](#)-Klasse erben, enthalten ein [ControlTemplate](#)-Objekt, das es dem Benutzer eines Steuerelements ermöglicht, die Darstellung des Steuerelements radikal zu ändern, ohne eine neue Unterklasse erstellen zu müssen. In diesem Thema wird erläutert, wie Steuerelemente (sowohl die, die von der [Control](#)-Klasse erben, als auch die, die nicht von ihr erben) gewöhnlich in WPF verwendet werden.

Erstellen einer Instanz eines Steuerelements

Sie können einer Anwendung ein Steuerelement hinzufügen, indem Sie XAML (Extensible Application Markup Language) oder Code verwenden. Im folgenden Beispiel wird veranschaulicht, wie Sie eine einfache Anwendung erstellen, die Vor- und Nachname eines Benutzers abfragt. In diesem Beispiel werden sechs Steuerelemente in XAML erstellt: zwei Bezeichnungen, zwei Textfelder und zwei Schaltflächen. Alle Steuerelemente können auf ähnliche Weise erstellt werden.

XAML

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition/>
    <ColumnDefinition/>
  </Grid.ColumnDefinitions>

  <Label>
    Enter your first name:
  </Label>
  <TextBox Grid.Row="0" Grid.Column="1">
```

```

        Name="firstName" Margin="0,5,10,5"/>

<Label Grid.Row="1" >
    Enter your last name:
</Label>
<TextBox Grid.Row="1" Grid.Column="1"
    Name="lastName" Margin="0,5,10,5"/>

<Button Grid.Row="2" Grid.Column="0"
    Name="submit" Margin="2">
    View message
</Button>

<Button Grid.Row="2" Grid.Column="1"
    Name="Clear" Margin="2">
    Clear Name
</Button>
</Grid>

```

Im folgenden Beispiel wird die gleiche Anwendung in Code erstellt. Aus Gründen der Übersichtlichkeit wurde die Erstellung von `Grid`, `grid1`, aus dem Beispiel ausgeschlossen. `grid1` verfügt über die gleichen Spalten- und Zeilendefinitionen wie im vorangehenden XAML-Beispiel.

C#

```

Label firstNameLabel;
Label lastNameLabel;
TextBox firstName;
TextBox lastName;
Button submit;
Button clear;

void CreateControls()
{
    firstNameLabel = new Label();
    firstNameLabel.Content = "Enter your first name:";
    grid1.Children.Add(firstNameLabel);

    firstName = new TextBox();
    firstName.Margin = new Thickness(0, 5, 10, 5);
    Grid.SetColumn(firstName, 1);
    grid1.Children.Add(firstName);

    lastNameLabel = new Label();
    lastNameLabel.Content = "Enter your last name:";
    Grid.SetRow(lastNameLabel, 1);
    grid1.Children.Add(lastNameLabel);

    lastName = new TextBox();
    lastName.Margin = new Thickness(0, 5, 10, 5);
    Grid.SetColumn(lastName, 1);
}

```



```

Grid.SetRow(lastName, 1);
grid1.Children.Add(lastName);

submit = new Button();
submit.Content = "View message";
Grid.SetRow(submit, 2);
grid1.Children.Add(submit);

clear = new Button();
clear.Content = "Clear Name";
Grid.SetRow(clear, 2);
Grid.SetColumn(clear, 1);
grid1.Children.Add(clear);
}

```

Ändern der Darstellung eines Steuerelements

Es ist üblich, die Darstellung eines Steuerelements zu ändern, sodass es zum Aussehen und Verhalten der Anwendung passt. Sie können die Darstellung eines Steuerelements je nach Absicht auf eine der folgenden Weisen ändern:

- Ändern des Werts einer Eigenschaft des Steuerelements.
- Erstellen Sie eine [Style](#)-Klasse für das Steuerelement.
- Erstellen Sie eine neue [ControlTemplate](#) für das Steuerelement.

Ändern eines Eigenschaftswerts eines Steuerelements

Viele Steuerelemente haben Eigenschaften, mit denen Sie die Darstellung des Steuerelements verändern können, z. B. den [Background](#) eines [Button](#)-Objekts. Sie können die Werteigenschaften sowohl in XAML als auch in Code festlegen. Im folgenden Beispiel werden die Eigenschaften [Background](#), [FontSize](#) und [FontWeight](#) für ein [Button](#)-Objekt in XAML festgelegt.

XAML

```

<Button FontSize="14" FontWeight="Bold">
  <!--Set the Background property of the Button to
  a LinearGradientBrush.-->
  <Button.Background>
    <LinearGradientBrush StartPoint="0,0.5"
                        EndPoint="1,0.5">
      <GradientStop Color="Green" Offset="0.0" />
      <GradientStop Color="White" Offset="0.9" />
    </LinearGradientBrush>
  </Button.Background>
</Button>

```

```
</Button.Background>  
View message  
</Button>
```

Im folgenden Beispiel werden die gleichen Eigenschaften in Code festgelegt.

C#

```
LinearGradientBrush buttonBrush = new LinearGradientBrush();  
buttonBrush.StartPoint = new Point(0, 0.5);  
buttonBrush.EndPoint = new Point(1, 0.5);  
buttonBrush.GradientStops.Add(new GradientStop(Colors.Green, 0));  
buttonBrush.GradientStops.Add(new GradientStop(Colors.White, 0.9));  
  
submit.Background = buttonBrush;  
submit.FontSize = 14;  
submit.FontWeight = FontWeights.Bold;
```

Erstellen eines Stils für ein Steuerelement

WPF ermöglicht Ihnen, die Darstellung von Steuerelementen global zu bestimmen, anstatt die Eigenschaften auf jeder Instanz in der Anwendung durch Erstellen eines [Style](#)-Objekts festzulegen. Im folgenden Beispiel wird eine [Style](#)-Klasse erstellt, die auf jedes [Button](#)-Objekt in der Anwendung angewendet wird. [Style](#)-Definitionen werden in der Regel in XAML in einem [ResourceDictionary](#) definiert, wie z. B. die Eigenschaft [Resources](#) für das [FrameworkElement](#).

XAML

```
<Style TargetType="Button">  
  <Setter Property="FontSize" Value="14"/>  
  <Setter Property="FontWeight" Value="Bold"/>  
  <Setter Property="Background">  
    <Setter.Value>  
      <LinearGradientBrush StartPoint="0,0.5"  
                           EndPoint="1,0.5">  
        <GradientStop Color="Green" Offset="0.0" />  
        <GradientStop Color="White" Offset="0.9" />  
      </LinearGradientBrush>  
    </Setter.Value>  
  </Setter>  
</Style>
```

Sie können einen Stil auch nur auf einige Steuerelemente eines bestimmten Typs anwenden, indem Sie dem Stil einen Schlüssel zuweisen und diesen Schlüssel in der

`Style`-Eigenschaft des Steuerelements angeben. Weitere Informationen zu Stilen finden Sie unter [Erstellen von Formaten und Vorlagen](#).

Erstellen einer Steuerelementvorlage

Mit einer [Style](#)-Klasse können Sie Eigenschaften für mehrere Steuerelemente gleichzeitig festlegen. Es kann jedoch vorkommen, dass Sie die Darstellung eines [Control](#)-Steuerelements noch weiter ändern möchten, als Ihnen das Erstellen einer [Style](#)-Klasse erlaubt. Klassen, die von der [Control](#)-Klasse erben, verfügen über eine [ControlTemplate](#), in der die Struktur und die Darstellung eines [Control](#)-Objekts definiert sind. Die [Template](#)-Eigenschaft eines [Control](#)-Objekts ist öffentlich, sodass Sie für ein [Control](#)-Objekt eine [ControlTemplate](#) festlegen können, die von der Standardeinstellung abweicht. Oft können Sie für ein [Control](#)-Objekt eine neue [ControlTemplate](#) angeben, statt sie von einem Steuerelement zu übernehmen, um die Darstellung eines [Control](#)-Objekts anzupassen.

Sehen Sie sich das sehr häufig verwendete Steuerelement [Button](#) an. Der Hauptzweck eines [Button](#)-Objekts besteht darin, einer Anwendung die Durchführung einer Aktion zu ermöglichen, wenn der Benutzer darauf klickt. In der Standardeinstellung wird das [Button](#)-Objekt in WPF als ein erhöhtes Rechteck angezeigt. Beim Entwickeln einer Anwendung können Sie das Verhalten eines [Button](#)-Steuerelements nutzen, d. h. das Klickereignis der Schaltfläche verarbeiten. Zudem haben Sie jedoch die Möglichkeit, die Darstellung der Schaltfläche weitergehend zu verändern, als dies durch Ändern der Schaltflächeneigenschaften möglich ist. In diesem Fall können Sie eine neue [ControlTemplate](#) erstellen.

Im folgenden Beispiel wird eine [ControlTemplate](#) für ein [Button](#)-Objekt erstellt. Die [ControlTemplate](#) erstellt ein [Button](#)-Objekt mit abgerundeten Ecken und einem Hintergrund mit Farbverlauf. Die [ControlTemplate](#) enthält ein [Border](#)-Objekt, dessen [Background](#) aus einem [LinearGradientBrush](#) mit zwei [GradientStop](#)-Objekten besteht. Das erste [GradientStop](#)-Objekt verwendet die Datenbindung, um die [Color](#)-Eigenschaft des [GradientStop](#)-Objekts an die Farbe des Schaltflächenhintergrunds zu binden. Wenn Sie die [Background](#)-Eigenschaft des [Button](#)-Objekts festlegen, wird die Farbe dieses Werts als erstes [GradientStop](#)-Objekt verwendet. Weitere Informationen zu Datenbindungen finden Sie unter [Übersicht über Datenbindung](#). Im Beispiel wird außerdem ein [Trigger](#) erstellt, der die Darstellung des [Button](#)-Objekts ändert, wenn [IsPressed](#) den Wert `true` aufweist.

XAML

```
<!--Define a template that creates a gradient-colored button.-->
<Style TargetType="Button">
```

```

<Setter Property="Template">
  <Setter.Value>
    <ControlTemplate TargetType="Button">
      <Border
        x:Name="Border"
        CornerRadius="20"
        BorderThickness="1"
        BorderBrush="Black">
        <Border.Background>
          <LinearGradientBrush StartPoint="0,0.5"
                                EndPoint="1,0.5">
            <GradientStop Color="{Binding Background.Color,
                                RelativeSource={RelativeSource TemplatedParent}}"
                            Offset="0.0" />
            <GradientStop Color="White" Offset="0.9" />
          </LinearGradientBrush>
        </Border.Background>
        <ContentPresenter
          Margin="2"
          HorizontalAlignment="Center"
          VerticalAlignment="Center"
          RecognizesAccessKey="True"/>
      </Border>
      <ControlTemplate.Triggers>
        <!--Change the appearance of
        the button when the user clicks it.-->
        <Trigger Property="IsPressed" Value="true">
          <Setter TargetName="Border" Property="Background">
            <Setter.Value>
              <LinearGradientBrush StartPoint="0,0.5"
                                    EndPoint="1,0.5">
                <GradientStop Color="{Binding Background.Color,
                                    RelativeSource={RelativeSource TemplatedParent}}"
                                Offset="0.0" />
                <GradientStop Color="DarkSlateGray" Offset="0.9" />
              </LinearGradientBrush>
            </Setter.Value>
          </Setter>
        </Trigger>

      </ControlTemplate.Triggers>
    </ControlTemplate>
  </Setter.Value>
</Setter>
</Style>

```

XAML

```

<Button Grid.Row="2" Grid.ColumnSpan="2" Name="submitName"
        Background="Green">View message</Button>

```

ⓘ Hinweis

Die Background-Eigenschaft des Button-Objekts muss auf SolidColorBrush festgelegt werden, damit dieses Beispiel ordnungsgemäß funktioniert.

Abonnieren von Ereignissen

Sie können das Ereignis eines Steuerelements abonnieren, indem Sie entweder XAML oder Code verwenden, aber Sie können ein Ereignis nur in Code behandeln. Im folgenden Beispiel wird das Abonnieren des `Click`-Ereignisses für ein `Button`-Objekt veranschaulicht.

XAML

```
<Button Grid.Row="2" Grid.ColumnSpan="2" Name="submitName"
Click="submit_Click"
Background="Green">View message</Button>
```

C#

```
submit.Click += new RoutedEventHandler(submit_Click);
```

Im folgenden Beispiel wird das `Click`-Ereignis eines `Button`-Objekts verarbeitet.

C#

```
void submit_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Hello, " + firstName.Text + " " + lastName.Text);
}
```

Umfangreicher Inhalt in Steuerelementen

Die meisten Klassen, die von der `Control`-Klasse erben, besitzen die Kapazität für umfangreiche Inhalte. Beispielsweise kann ein `Label`-Objekt jedes Objekt enthalten, z. B. eine Zeichenfolge, ein `Image` oder ein `Panel`. Die folgenden Klassen unterstützen umfangreichen Inhalt und dienen als Basisklassen für die meisten Steuerelemente in WPF.

- `ContentControl`: Einige Beispiele für Klassen, die von dieser Klasse erben, sind `Label`, `Button` und `ToolTip`.

- [ItemsControl](#): Einige Beispiele für Klassen, die von dieser Klasse erben, sind [ListBox](#), [Menu](#) und [StatusBar](#).
- [HeaderedContentControl](#): Einige Beispiele für Klassen, die von dieser Klasse erben, sind [TabItem](#), [GroupBox](#) und [Expander](#).
- [HeaderedItemsControl](#): Einige Beispiele für Klassen, die von dieser Klasse erben, sind [MenuItem](#), [TreeViewItem](#) und [ToolBar](#).

Weitere Informationen zu diesen Basisklassen finden Sie unter [WPF-Inhaltsmodell](#).

Weitere Informationen

- [Erstellen von Formaten und Vorlagen](#)
- [Kategorien von Steuerelementen](#)
- [Steuerelementbibliothek](#)
- [Übersicht über Datenvorlagen](#)
- [Übersicht zur Datenbindung](#)
- [Eingabe](#)
- [Aktivieren eines Befehls](#)
- [Exemplarische Vorgehensweisen: Erstellen einer benutzerdefinierten animierten Schaltfläche](#)
- [Anpassung von Steuerelementen](#)

Zusammenarbeit auf GitHub

Die Quelle für diesen Inhalt finden Sie auf GitHub, wo Sie auch Issues und Pull Requests erstellen und überprüfen können. Weitere Informationen finden Sie in unserem [Leitfaden für Mitwirkende](#).



Feedback zu .NET Desktop feedback

.NET Desktop feedback ist ein Open Source-Projekt. Wählen Sie einen Link aus, um Feedback zu geben:

 [Problem in der Dokumentation öffnen](#)

 [Abgeben von Produktfeedback](#)

Daten

Artikel • 04.05.2023

Die Datenbindung in Windows Presentation Foundation (WPF) bietet für Anwendungen eine einfache und konsistente Möglichkeit, Daten darzustellen und mit ihnen zu interagieren. Elemente können an Daten aus unterschiedlichen Typen von Datenquellen in Form von CLR-Objekten (Common Language Runtime) und XML gebunden werden. Windows Presentation Foundation (WPF) bietet außerdem einen Mechanismus für die Übertragung von Daten durch Drag & Drop-Vorgänge.

In diesem Abschnitt

[Datenbindung](#)

[Drag & Drop](#)

Referenz

[System.Windows.Data](#)

[Binding](#)

[DataTemplate](#)

[DataTemplateSelector](#)

Verwandte Abschnitte

[Steuerelemente](#)

[Erstellen von Formaten und Vorlagen](#)

[Datenbindung](#)

Siehe auch

- [Exemplarische Vorgehensweise: Walkthrough: My first WPF desktop application \(Exemplarische Vorgehensweise: Meine erste WPF-Desktopanwendung\)](#)
- [Exemplarische Vorgehensweise: Zwischenspeichern von Anwendungsdaten in einer WPF-Anwendung](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET Desktop feedback

The .NET Desktop documentation is open source. Provide feedback here.

 [Open a documentation issue](#)

 [Provide product feedback](#)

Grafiken und Multimedia

Artikel • 04.05.2023

Windows Presentation Foundation (WPF) bietet Unterstützung für Multimedia, Vektorgrafiken, Animation und Inhaltskomposition und erleichtert Entwicklern das Erstellen interessanter Benutzeroberflächen und Inhalte. Mithilfe von Visual Studio können Sie Vektorgrafiken oder komplexe Animationen erstellen und Medien in Ihre Anwendung integrieren.

In diesem Thema werden die Grafik-, Animations- und Medienfeatures von WPF vorgestellt, mit denen Sie den Anwendungen Grafiken, Übergangseffekte, Ton und Videos hinzufügen können.

ⓘ Hinweis

Es wird dringend davon abgeraten, WPF-Typen in einem Windows-Dienst zu verwenden. Wenn Sie versuchen, WPF-Typen in einem Windows-Dienst zu verwenden, funktioniert der Dienst möglicherweise nicht wie erwartet.

Neue Grafik. und Multimediafunktionen in WPF 4

Im Zusammenhang mit Grafiken und Animationen wurden mehrere Änderungen vorgenommen.

- Layoutglättung

Wenn ein Objektrand in die Mitte eines Pixelgeräts fällt, kann das DPI-unabhängige Grafiksystem Renderingartefakte erstellen, z.B. verschwommene oder semitransparente Ränder. Frühere Versionen der WPF haben die Pixelausrichtung für diese Fälle verwendet. Silverlight 2 hat die Layoutglättung eingeführt, die eine andere Möglichkeit bietet, um Elemente so zu verschieben, dass Ränder auf ganzen Pixelgrenzen liegen. WPF unterstützt nun die Layoutglättung mit der angefügten [UseLayoutRounding](#)-Eigenschaft auf der [FrameworkElement](#)-Klasse.

- Zwischengespeicherte Komposition

Mithilfe der neuen Klassen [BitmapCache](#) und [BitmapCacheBrush](#) können Sie einen komplexen Teil der visuellen Struktur als Bitmap zwischenspeichern und so die Renderingzeit erheblich verbessern. Die Bitmap reagiert weiterhin auf

Benutzereingaben wie Mausklicks, und Sie können sie wie jeden Pinsel auf anderen Elementen zeichnen.

- Unterstützung von Pixel Shader 3

WPF 4 basiert auf der in WPF 3.5 SP1 eingeführten [ShaderEffect](#)-Unterstützung, indem Anwendungen ermöglicht wird, Effekte mithilfe von Pixel Shader Version 3.0 (PS) zu schreiben. Das PS 3.0-Shadermodell ist ausgereifter als PS 2.0 und ermöglicht noch mehr Effekte auf unterstützter Hardware.

- Beschleunigungsfunktionen

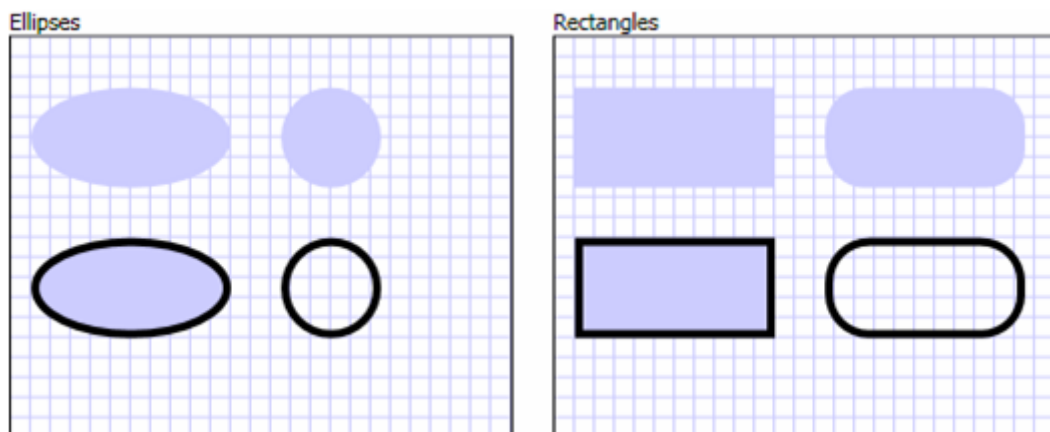
Sie können Animationen mit Beschleunigungsfunktionen verbessern, die Ihnen zusätzliche Kontrolle über das Verhalten von Animationen geben. Sie können beispielsweise eine [ElasticEase](#) auf eine Animation anwenden, um dieser ein Federverhalten zuzuweisen. Weitere Informationen finden Sie unter Beschleunigungstypen im [System.Windows.Media.Animation](#)-Namespace.

Grafiken und Rendering

WPF unterstützt hochwertige 2D-Grafiken. Die Funktionalität umfasst die Pinsel, Geometrien, Bilder, Formen und Transformationen. Weitere Informationen finden Sie unter [Grafik](#). Das Rendern grafischer Elemente basiert auf der [Visual](#)-Klasse. Die Struktur von visuellen Objekten auf dem Bildschirm wird durch die visuelle Struktur beschrieben. Weitere Informationen finden Sie unter [Übersicht über das WPF-Grafikenrendering](#).

2D-Formen

WPF enthält eine Bibliothek häufig verwendeter vektorbasierter 2D-Formen wie Rechtecke und Ellipsen, die in der folgenden Abbildung dargestellt sind.



Diese systeminternen WPF-Formen sind mehr als nur Formen: Es handelt sich um programmierbare Elemente, die viele der Features implementieren, die Sie von den gebräuchlichsten Steuerelementen erwarten, etwa Tastatur- und Mauseingaben. Die folgenden Beispiele zeigen, wie das ausgelöste [MouseDown](#)-Ereignis durch Klicken auf ein [Ellipse](#)-Element verarbeitet wird.

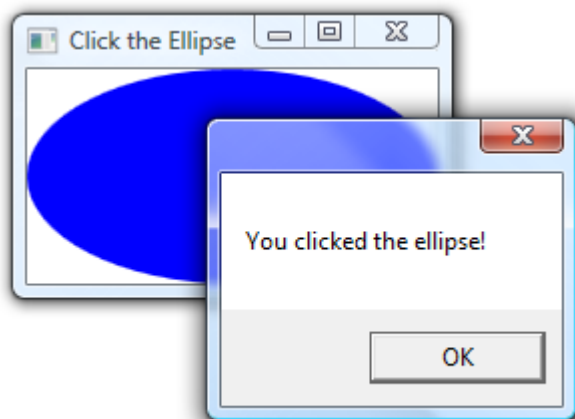
XAML

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="Window1" >
  <Ellipse Fill="LightBlue" MouseUp="ellipseButton_MouseUp" />
</Window>
```

C#

```
public partial class Window1 : Window
{
    void ellipseButton_MouseUp(object sender, MouseButtonEventArgs e)
    {
        MessageBox.Show("You clicked the ellipse!");
    }
}
```

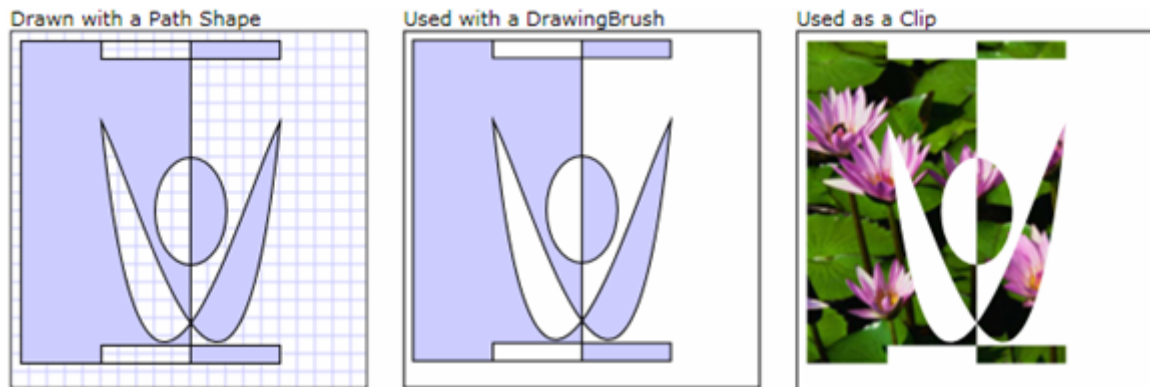
Die folgende Abbildung zeigt die Ausgabe für das vorangehende XAML-Markup und CodeBehind.



Weitere Informationen finden Sie unter [Übersicht über Formen und die grundlegenden Funktionen zum Zeichnen in WPF](#). Ein einführendes Beispiel finden Sie unter [Beispiel für Formelemente](#) [↗](#).

2D-Geometrien

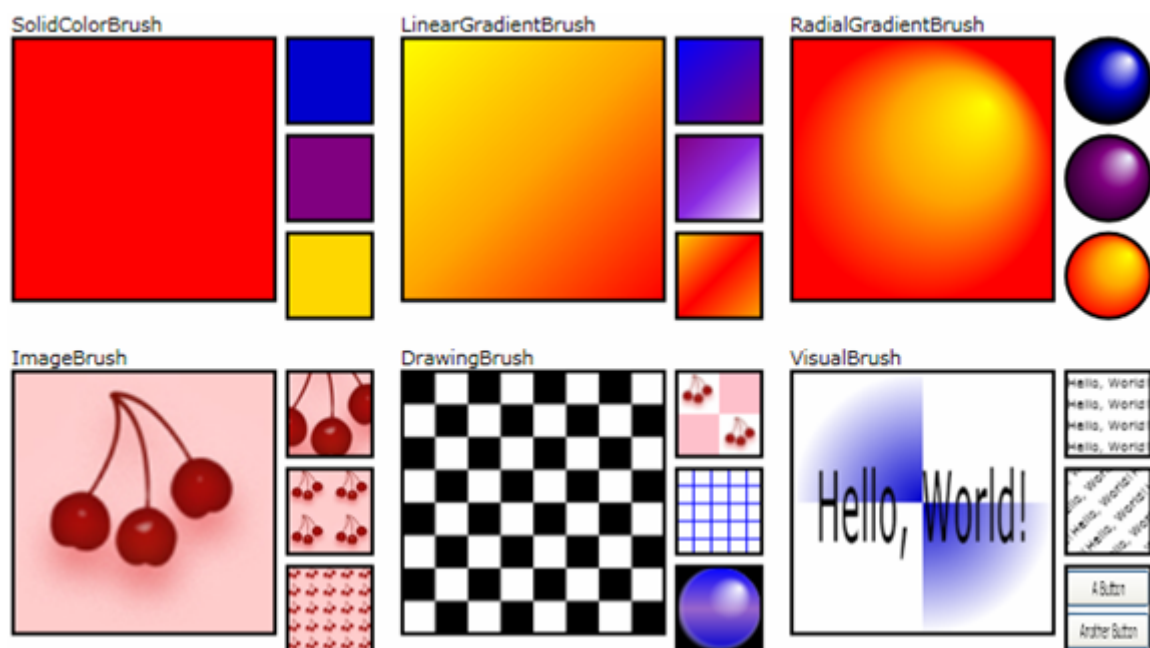
Wenn die 2D-Formen von WPF nicht ausreichen, können Sie mit der WPF-Unterstützung für Geometrien und Pfade eigene Formen erstellen. Die folgende Abbildung zeigt, wie Sie Geometrien nutzen können, um Formen zu erstellen, wie etwa einen Zeichenpinsel, und andere WPF-Elemente ausschneiden zu können.



Weitere Informationen finden Sie unter [Übersicht über die Geometrie](#). Ein einführendes Beispiel finden Sie unter [Beispiele zu Geometrie](#).

2D-Effekte

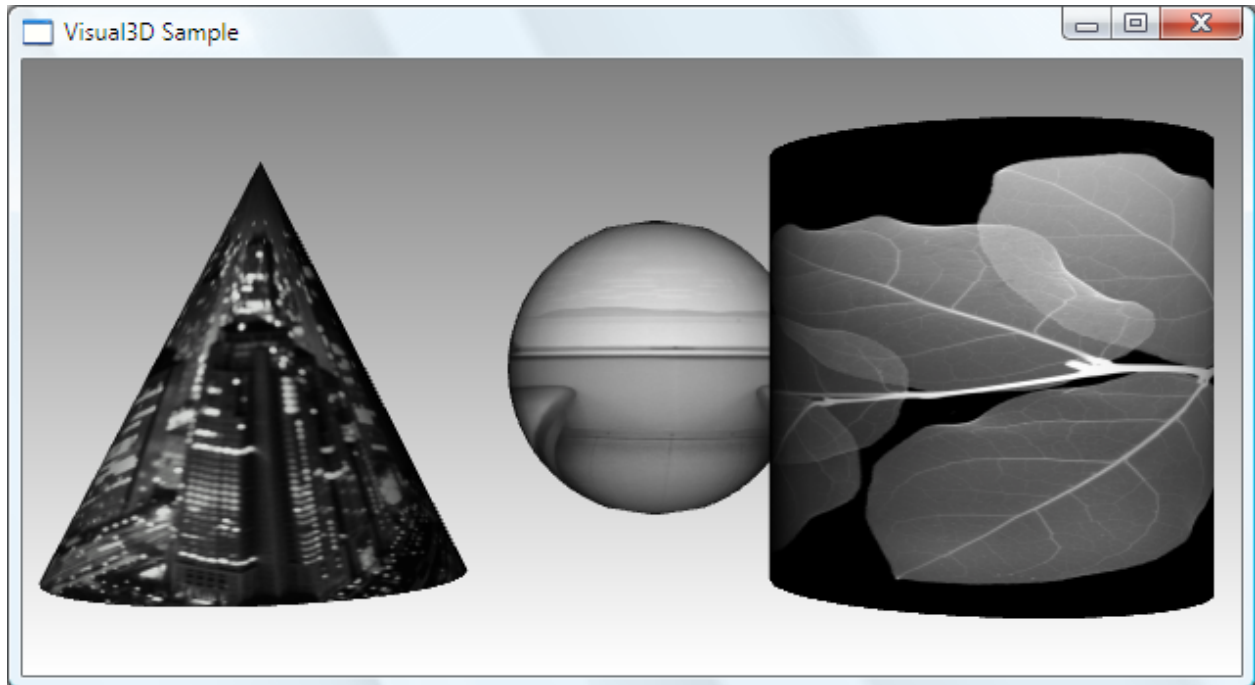
WPF stellt eine Bibliothek mit 2D-Klassen zur Verfügung, mit denen Sie verschiedene Effekte erstellen können. Die 2D-Renderingfunktion von WPF bietet die Möglichkeit, Elemente der Benutzeroberfläche mit Farbverläufen, Bitmaps, Zeichnungen und Videos zu zeichnen, und diese durch Drehung, Skalierung und Neigung zu bearbeiten. Die folgende Abbildung zeigt ein Beispiel für die vielen Effekte, die Sie mit WPF-Pinsels erreichen können.



Weitere Informationen finden Sie unter [Übersicht über WPF-Pinsel](#). Ein einführendes Beispiel finden Sie unter [Pinselbeispiel](#).

3D-Rendering

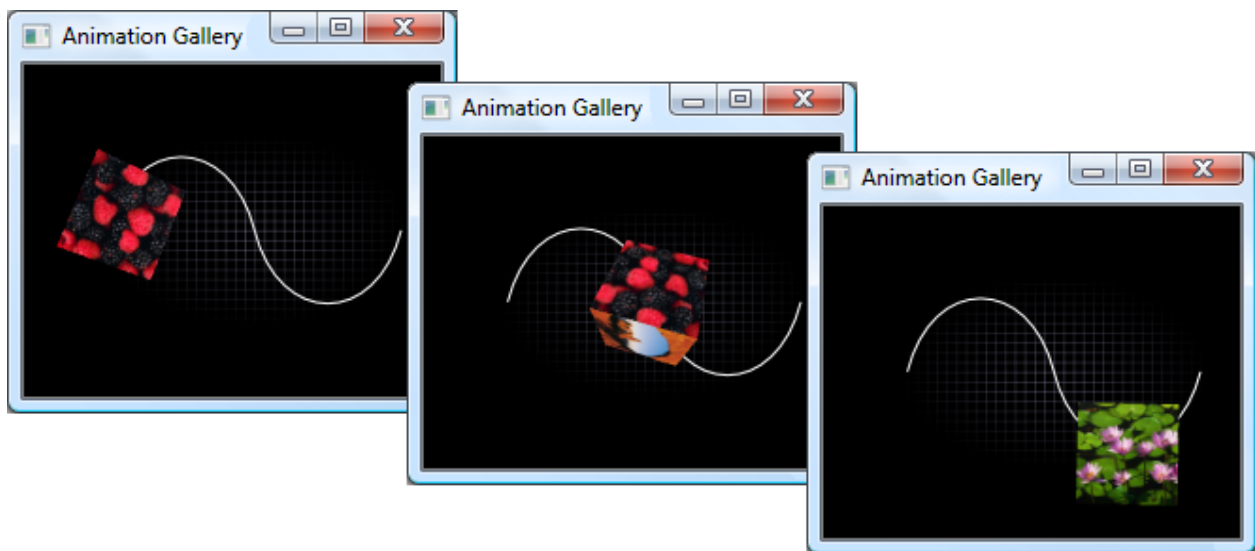
WPF bietet eine Reihe von 3D-Renderingfunktionen, die in der 2D-Grafikunterstützung in WPF integriert sind, damit Sie Layout, die Benutzeroberfläche und Datenvisualisierung noch ansprechender gestalten können. Mit WPF können Sie sogar 2D-Bilder auf den Oberflächen von 3D-Formen rendern, was die folgende Abbildung veranschaulicht.



Weitere Informationen finden Sie unter [Übersicht über 3D-Grafiken](#). Ein einführendes Beispiel finden Sie unter [Beispiel zu 3D-Festkörpern](#) [↗](#).

Animation

Mit Animation können Sie Steuerelemente und Elemente wachsen, bewegen, drehen sowie ein- und ausblenden lassen und z.B. interessante Seitenübergänge erzeugen. Da Sie mit WPF die meisten Eigenschaften animieren können, können Sie nicht nur die meisten WPF-Objekte animieren, sondern Sie können WPF auch verwenden, um benutzerdefinierte Objekte, die Sie erstellen, zu animieren.



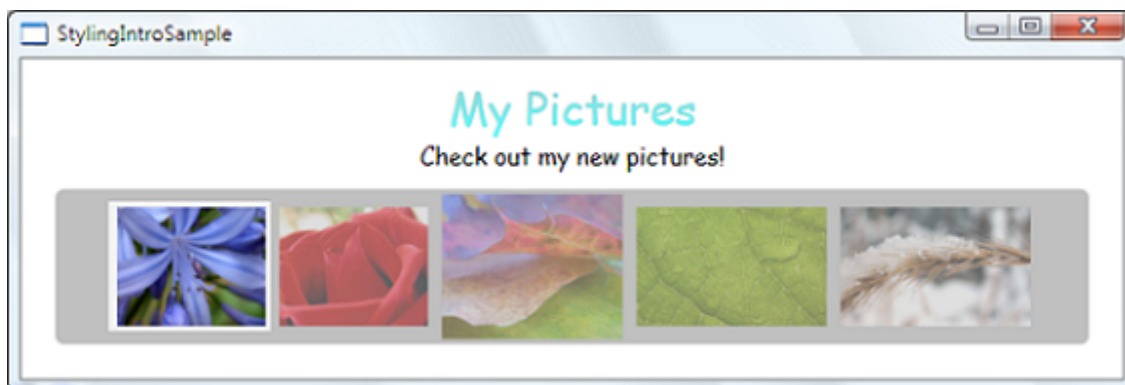
Weitere Informationen finden Sie unter [Übersicht über Animation](#). Ein einführendes Beispiel finden Sie unter [Beispielsammlung zu Animationen](#).

Medien

Bilder, Videos und Audiodateien bieten eine umfangreiche Medienunterstützung, um Informationen zu vermitteln und für Benutzerfreundlichkeit zu sorgen.

Bilder

Bilder, z.B. Symbole, Hintergründe und sogar Teile von Animationen, sind ein wesentlicher Bestandteil der meisten Anwendungen. Da Sie häufig Bilder verwenden müssen, bietet WPF die Möglichkeit, sie auf vielfältige Weise einzusetzen. In der folgenden Abbildung wird lediglich eine der Möglichkeiten dargestellt.



Weitere Informationen finden Sie unter [Übersicht über die Bildverarbeitung](#).

Video und Audio

Ein Hauptfeature der Grafikfunktionen von WPF besteht in der nativen Unterstützung für die Arbeit mit Multimedia, z. B. Video und Audio. Im folgenden Beispiel wird

veranschaulicht, wie einen MediaPlayer in eine Anwendung eingefügt wird.

XAML

```
<MediaElement Source="media\numbers.wmv" Width="450" Height="250" />
```

[MediaElement](#) kann sowohl Video- als auch Audiodateien abspielen und kann ausreichend erweitert werden, damit die einfache Erstellung von benutzerdefinierten Benutzeroberflächenelementen möglich ist.

Weitere Informationen finden Sie unter [Übersicht über Multimedia](#).

Weitere Informationen

- [System.Windows.Media](#)
- [System.Windows.Media.Animation](#)
- [System.Windows.Media.Media3D](#)
- [2D-Grafiken und Bildverarbeitung](#)
- [Übersicht über Formen und die grundlegenden Funktionen zum Zeichnen in WPF](#)
- [Übersicht über das Zeichnen mit Volltonfarben und Farbverläufen](#)
- [Zeichnen mit Bildern, Zeichnungen und visuellen Elementen](#)
- [Gewusst-wie-Themen zu Animation und Zeitsteuerung](#)
- [Übersicht über 3D-Grafiken](#)
- [Übersicht über Multimedia](#)

Sicherheit (WPF)

Artikel • 04.05.2023

Beim Entwickeln eigenständiger und im Browser gehosteter WPF-Anwendungen (Windows Presentation Foundation) müssen Sie das Sicherheitsmodell beachten. Eigenständige WPF-Anwendungen werden mit uneingeschränkten Berechtigungen (CAS-Berechtigungssatz **FullTrust**) ausgeführt, unabhängig davon, ob sie mit Windows Installer (MSI), XCopy oder ClickOnce bereitgestellt werden. Die Bereitstellung teilweise vertrauenswürdiger eigenständiger WPF-Anwendungen mit ClickOnce wird nicht unterstützt. Eine voll vertrauenswürdige Hostanwendung kann jedoch mithilfe des .NET Framework-Add-In-Modells eine teilweise vertrauenswürdige [AppDomain](#) erstellen. Weitere Informationen finden Sie unter [Übersicht über WPF-Add-Ins](#).

Im Browser gehostete WPF-Anwendungen werden von Windows Internet Explorer oder Firefox gehostet und können entweder XAML-Browseranwendungen (XBAPs) oder Loose XAML-Dokumente (Extensible Application Markup Language) sein. Weitere Informationen finden Sie unter [Übersicht über WPF-XAML-Browseranwendungen](#).

Im Browser gehostete WPF-Anwendungen werden standardmäßig in einem teilweise vertrauenswürdigen Sicherheitsbereich (Sandbox) ausgeführt, der auf den Standardberechtigungsatz für die CAS-Zone **Internet** beschränkt ist. Dadurch werden in einem Browser ausgeführte WPF-Anwendungen effektiv vom Clientcomputer isoliert, wie Sie es auch von typischen Webanwendungen erwarten. Eine XBAP kann, abhängig von der Sicherheitszone der Bereitstellungs-URL und der Sicherheitskonfiguration des Clients, Berechtigungen bis zur vollen Vertrauenswürdigkeit erhöhen. Weitere Informationen finden Sie unter [WPF-Sicherheit mit teilweiser Vertrauenswürdigkeit](#).

In diesem Thema wird das Sicherheitsmodell für eigenständige und im Browser gehostete WPF-Anwendungen (Windows Presentation Foundation) erläutert.

Dieses Thema enthält folgende Abschnitte:

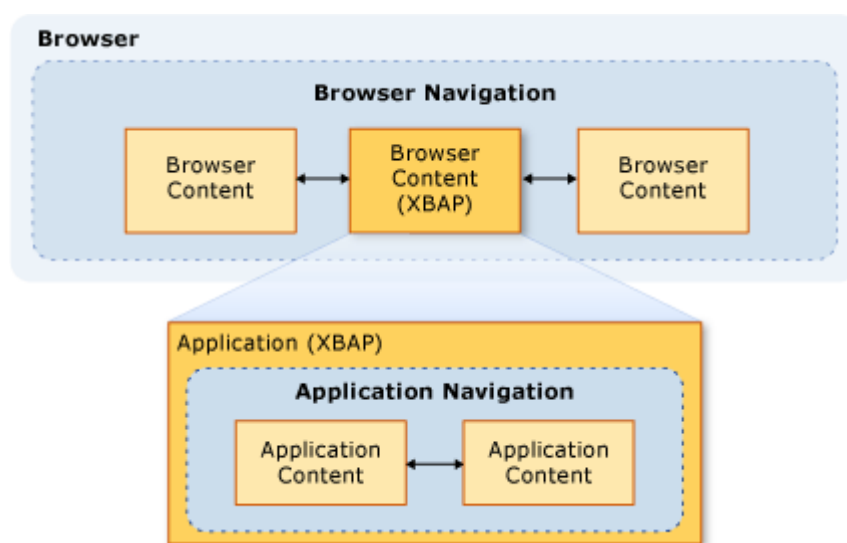
- [Sichere Navigation](#)
- [Sicherheitseinstellungen für webbrowsende Software](#)
- [WebBrowser-Steuerelement und Funktionssteuerelemente](#)
- [Deaktivieren von APTCA-Assemblys für teilweise vertrauenswürdige Clientanwendungen](#)
- [Sandkastenverhalten für Loose XAML-Dateien](#)

- [Ressourcen zum Entwickeln von WPF-Anwendungen, die die Sicherheit erhöhen](#)

Sichere Navigation

Bei XBAPs unterscheidet WPF zwei Navigationstypen: Anwendungsnavigation und Browsernavigation.

Als *Anwendungsnavigation* wird die Navigation zwischen Inhaltselementen in einer Anwendung bezeichnet, die in einem Browser gehostet wird. Als *Browsernavigation* wird die Navigation bezeichnet, die die Inhalts- und Speicherort-URL eines Browsers selbst ändert. Die Beziehung zwischen Anwendungsnavigation (in der Regel XAML) und Browsernavigation (in der Regel HTML) ist in der folgenden Abbildung dargestellt:



Der Inhaltstyp, der als sicheres Navigationsziel für eine XBAP gilt, wird hauptsächlich dadurch bestimmt, ob Anwendungsnavigation oder Browsernavigation verwendet wird.

Sicherheit von Anwendungsnavigation

Anwendungsnavigation wird als sicher betrachtet, wenn sie mit einem Paket-URI identifiziert werden kann, der vier Inhaltstypen unterstützt:

Inhaltstyp	BESCHREIBUNG	URI-Beispiel
Resource	Dateien, die einem Projekt mit einem Resource -Buildtyp hinzugefügt werden	<code>pack://application:,,,/MyResourceFile.xaml</code>

Inhaltstyp	BESCHREIBUNG	URI-Beispiel
Inhalt	Dateien, die einem Projekt mit einem Content -Buildtyp hinzugefügt werden	<code>pack://application:,,,/MyContentFile.xaml</code>
Ursprungswebsite	Dateien, die einem Projekt mit einem None -Buildtyp hinzugefügt werden	<code>pack://siteoforigin:,,,/MySiteOfOriginFile.xaml</code>
Anwendungscode	<p>XAML-Ressourcen mit kompiliertem Code-Behind</p> <p>Oder</p> <p>XAML-Dateien, die einem Projekt mit einem Page-Buildtyp hinzugefügt werden</p>	<code>pack://application:,,,/MyResourceFile .xaml</code>

⚠ Hinweis

Weitere Informationen zu Anwendungsdatendateien und Paket-URIs finden Sie unter **WPF-Anwendungsressource, Inhalts- und Datendateien**.

Ein Navigieren zu Dateien dieser Inhaltstypen kann sowohl durch einen Benutzer als auch programmgesteuert erfolgen:

- **Benutzernavigation.** Die Benutzer*innen navigieren durch Klicken auf ein [Hyperlink](#)-Element.
- **Programmgesteuerte Navigation** Die Anwendung navigiert, ohne die Benutzer*innen einzubinden, z. B. durch Festlegen der [NavigationWindow.Source](#)-Eigenschaft.

Sicherheit von Browsernavigation

Browsernavigation wird nur unter den folgenden Bedingungen als sicher betrachtet:

- **Benutzernavigation.** Die Benutzer*innen navigieren durch Klicken auf ein [Hyperlink](#)-Element, das sich im Haupt-[NavigationWindow](#) und nicht in einem geschachtelten [Frame](#) befindet.

- **Zone:** Der Inhalt, in dem navigiert wird, befindet sich im Internet oder im lokalen Intranet.
- **Protokoll.** Es wird eines der folgenden Protokolle verwendet: **http**, **https**, **file** oder **mailto**.

Wenn eine XBAP versucht, in einer Weise zu Inhalten zu navigieren, die nicht diesen Bedingungen entspricht, wird eine Sicherheitsausnahme ([SecurityException](#)) ausgelöst.

Sicherheitseinstellungen für webbrowsende Software

Die Sicherheitseinstellungen auf dem Computer bestimmen, welcher Zugriff jeder webbrowsenden Software gewährt wird. Zu webbrowsender Software gehören alle Anwendungen oder Komponenten, die die [WinINet](#)-API oder die [UrlMon](#)-API verwenden, einschließlich Internet Explorer und „PresentationHost.exe“.

Internet Explorer stellt einen Mechanismus bereit, mit dem Sie die Funktionalität konfigurieren können, deren Ausführung durch oder aus Internet Explorer zulässig ist. Dazu zählt Folgendes:

- Von .NET Framework abhängige Komponenten
- ActiveX-Steuerelemente und Plug-Ins
- Downloads
- Skripterstellung
- Benutzerauthentifizierung

Die Sammlung der Funktionen, die auf diese Weise geschützt werden können, wird auf Zonenbasis für die Zonen **Internet**, **Intranet**, **Vertrauenswürdige Sites** und **eingeschränkte Sites** konfiguriert. Die folgenden Schritte beschreiben, wie die Sicherheitseinstellungen konfiguriert werden:

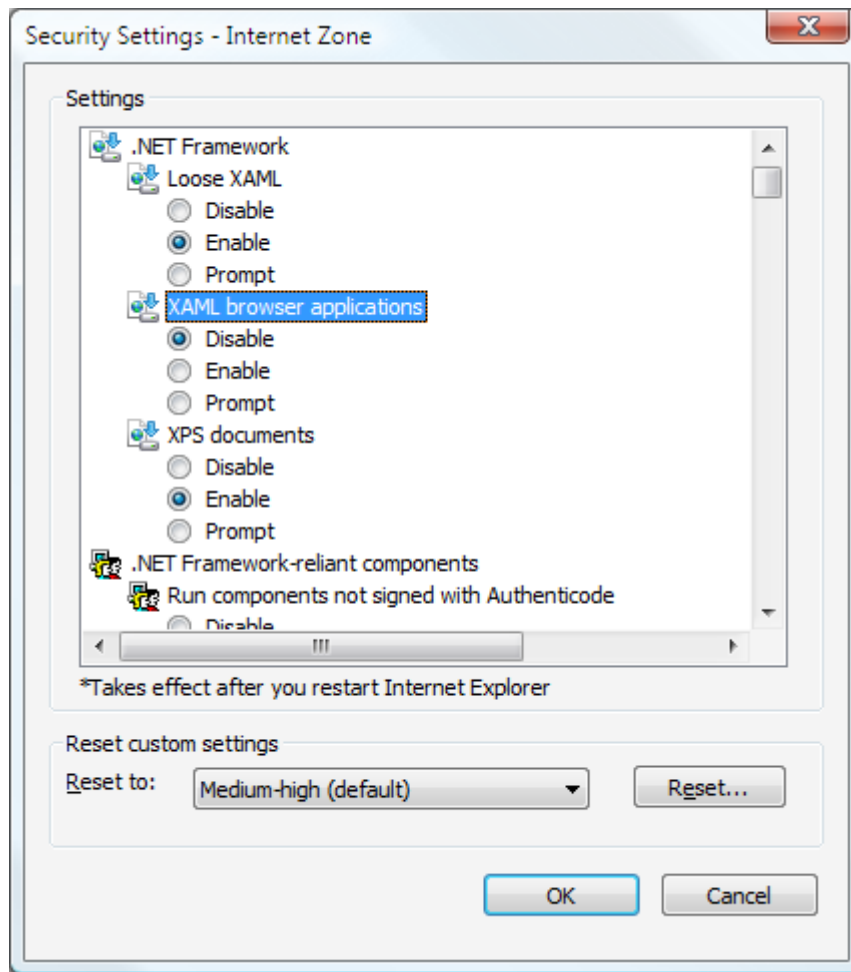
1. Öffnen Sie die **Systemsteuerung**.
2. Klicken Sie auf **Netzwerk und Internet** und dann auf **Internetoptionen**.

Das Dialogfeld "Internetoptionen" wird angezeigt.

3. Wählen Sie auf der Registerkarte **Sicherheit** die Zone aus, für die die Sicherheitseinstellungen konfiguriert werden sollen.

4. Klicken Sie auf die Schaltfläche **Stufe anpassen**.

Das Dialogfeld **Sicherheitseinstellungen** wird geöffnet, und Sie können die Sicherheitseinstellungen für die ausgewählte Zone konfigurieren.



! Hinweis

Sie können auch aus Internet Explorer zum Dialogfeld „Internetoptionen“ gelangen. Klicken Sie auf **Extras** und dann auf **Internetoptionen**.

Ab Windows Internet Explorer 7 sind die folgenden Sicherheitseinstellungen speziell für .NET Framework enthalten:

- **Loose XAML.** Steuert, ob Internet Explorer zu Loose XAML-Dateien navigieren kann (Optionen „Aktivieren“, „Deaktivieren“ und „Bestätigen“.)
- **XAML-Browseranwendungen.** Steuert, ob Internet Explorer zu XBAPs navigieren und diese ausführen kann (Optionen „Aktivieren“, „Deaktivieren“ und „Bestätigen“.)

Standardmäßig sind diese Einstellungen für die Zonen **Internet**, **Lokales Intranet** und **Vertrauenswürdige Sites** aktiviert und für die Zone **Eingeschränkte Sites** deaktiviert.

Sicherheitsbezogene WPF-Registrierungseinstellungen

Zusätzlich zu den Sicherheitseinstellungen, die über die Internetoptionen verfügbar sind, sind die folgenden Registrierungswerte für die selektive Blockierung einiger sicherheitsrelevanter WPF-Funktionen verfügbar. Die Werte werden unter dem folgenden Schlüssel definiert:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ .NETFramework\Windows Presentation  
Foundation\Features
```

In der folgenden Tabelle sind die Werte aufgelistet, die festgelegt werden können.

Wertname	Werttyp	Wertdaten
XBAPDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen
LooseXamlDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen
WebBrowserDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen
MediaAudioDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen
MediaImageDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen
MediaVideoDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen
ScriptInteropDisallow	REG_DWORD	1 = nicht zulassen; 0 = zulassen

WebBrowser-Steuerelement und Funktionssteuerelemente

Das [WebBrowser](#)-WPF-Steuerelement kann zum Hosten von Webinhalten verwendet werden. Das [WebBrowser](#)-WPF-Steuerelement umschließt das zugrunde liegende WebBrowser-ActiveX-Steuerelement. WPF bietet eine gewisse Unterstützung zum Schützen der Anwendung, wenn Sie mit dem [WebBrowser](#)-WPF-Steuerelement Webinhalt hosten, der nicht vertrauenswürdig ist. Einige Sicherheitsfunktionen müssen jedoch direkt durch die Anwendungen mit dem [WebBrowser](#)-Steuerelement angewandt werden. Weitere Informationen zum WebBrowser-ActiveX-Steuerelement finden Sie unter [WebBrowser-Steuerelement: Übersicht und Tutorials](#).

ⓘ Hinweis

Dieser Abschnitt gilt auch für das **Frame**-Steuerelement, da es über **WebBrowser** zu HTML-Inhalt navigiert.

Wenn das [WebBrowser](#)-WPF-Steuerelement zum Hosten von nicht vertrauenswürdigen Webinhalten verwendet wird, sollte die Anwendung eine teilweise vertrauenswürdige [AppDomain](#) verwenden, um den Anwendungscode von potenziell bösartigem HTML-Skriptcode zu isolieren. Dies trifft insbesondere zu, wenn die Anwendung über die [InvokeScript](#)-Methode und die [ObjectForScripting](#)-Eigenschaft mit dem gehosteten Skript interagiert. Weitere Informationen finden Sie unter [Übersicht über WPF-Add-Ins](#).

Wenn die Anwendung das [WebBrowser](#)-WPF-Steuerelement verwendet, ist das Aktivieren der Internet Explorer-Funktionssteuerelemente eine weitere Möglichkeit, die Sicherheit zu erhöhen und Angriffe zu verhindern. Funktionssteuerelemente sind Ergänzungen zu Internet Explorer, mit denen Administrator*innen und Entwickler*innen Funktionen von Internet Explorer und Anwendungen konfigurieren können, die das WebBrowser-ActiveX-Steuerelement hosten, das vom [WebBrowser](#)-WPF-Steuerelement umschlossen ist. Funktionssteuerelemente können mithilfe der [ColInternetSetFeatureEnabled](#)-Funktion oder durch Ändern von Werten in der Registrierung konfiguriert werden. Weitere Informationen zu Funktionssteuerelementen finden Sie unter [Einführung in Funktionssteuerelemente](#) und [Internet-Funktionssteuerelemente](#).

Wenn Sie eine eigenständige WPF-Anwendung entwickeln, in der das [WebBrowser](#)-WPF-Steuerelement verwendet wird, aktiviert WPF automatisch die folgenden Funktionssteuerelemente für die Anwendung.

Funktionssteuerelement
FEATURE_MIME_HANDLING
FEATURE_MIME_SNIFFING
FEATURE_OBJECT_CACHING
FEATURE_SAFE_BINDTOOBJECT
FEATURE_WINDOW_RESTRICTIONS
FEATURE_ZONE_ELEVATION
FEATURE_RESTRICT_FILEDOWNLOAD
FEATURE_RESTRICT_ACTIVEXINSTALL
FEATURE_ADDON_MANAGEMENT
FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
FEATURE_SECURITYBAND

Funktionssteuerelement
FEATURE_UNC_SAVEDFILECHECK
FEATURE_VALIDATE_NAVIGATE_URL
FEATURE_DISABLE_TELNET_PROTOCOL
FEATURE_WEBOC_POPUPMANAGEMENT
FEATURE_DISABLE_LEGACY_COMPRESSION
FEATURE_SSLUX

Da diese Funktionssteuerelemente bedingungslos aktiviert werden, können sie möglicherweise voll vertrauenswürdige Anwendungen beeinträchtigen. In diesem Fall kann das entsprechende Funktionssteuerelement deaktiviert werden, wenn kein Sicherheitsrisiko für die jeweilige Anwendung und den gehosteten Inhalt besteht.

Funktionssteuerelemente werden von dem Prozess angewandt, der das WebBrowser-ActiveX-Objekt instanziiert. Daher wird unbedingt empfohlen, beim Erstellen einer eigenständigen Anwendung, die zu nicht vertrauenswürdigen Inhalt navigieren kann, zusätzliche Funktionssteuerelemente zu aktivieren.

ⓘ Hinweis

Diese Empfehlung basiert auf allgemeinen Empfehlungen für MSHTML- und SHDOCVW-Hostsicherheit. Weitere Informationen finden Sie unter **The MSHTML Host Security FAQ: Part I of II** [↗](#) (Häufig gestellte Fragen zur MSHTML-Hostsicherheit: Teil I von II) und **The MSHTML Host Security FAQ: Part II of II** [↗](#) (Häufig gestellte Fragen zur MSHTML-Hostsicherheit: Teil II von II).

Für eine ausführbare Datei sollten die folgenden Funktionssteuerelemente aktiviert werden, indem der Registrierungswert auf 1 festgelegt wird.

Funktionssteuerelement
FEATURE_ACTIVEX_REPURPOSEDETECTION
FEATURE_BLOCK_LMZ_IMG
FEATURE_BLOCK_LMZ_OBJECT
FEATURE_BLOCK_LMZ_SCRIPT
FEATURE_RESTRICT_RES_TO_LMZ

Funktionssteuerelement
FEATURE_RESTRICT_ABOUT_PROTOCOL_IE7
FEATURE_SHOW_APP_PROTOCOL_WARN_DIALOG
FEATURE_LOCALMACHINE_LOCKDOWN
FEATURE_FORCE_ADDR_AND_STATUS
FEATURE_RESTRICTED_ZONE_WHEN_FILE_NOT_FOUND

Für eine ausführbare Datei sollte das folgenden Funktionssteuerelement deaktiviert werden, indem der Registrierungswert auf 0 festgelegt wird.

Funktionssteuerelement
FEATURE_ENABLE_SCRIPT_PASTE_URLACTION_IF_PROMPT

Wenn Sie eine teilweise vertrauenswürdige XBAP (XAML-Browseranwendung) ausführen, die ein [WebBrowser](#)-WPF-Steuerelement in Windows Internet Explorer enthält, hostet WPF das WebBrowser-ActiveX-Steuerelement im Adressbereich des Internet Explorer-Prozesses. Da das WebBrowser-ActiveX-Steuerelement im Internet Explorer-Prozess gehostet wird, sind alle Funktionssteuerelemente für Internet Explorer auch für das WebBrowser-ActiveX-Steuerelement aktiviert.

XBAPs, die in Internet Explorer ausgeführt werden, haben im Vergleich zu normalen eigenständigen Anwendungen ebenfalls ein höheres Maß an Sicherheit. Diese zusätzliche Sicherheit ergibt sich daraus, dass Internet Explorer, und somit auch das WebBrowser-ActiveX-Steuerelement, unter Windows Vista und Windows 7 standardmäßig im geschützten Modus ausgeführt wird. Weitere Informationen zum geschützten Modus finden Sie unter [Grundlegendes und Arbeiten mit Internet Explorer im geschützten Modus](#).

ⓘ Hinweis

Wenn Sie versuchen, in Firefox in der Internetzone eine XBAP auszuführen, die ein [WebBrowser](#)-WPF-Steuerelement enthält, wird eine Sicherheitsausnahme (**SecurityException**) ausgelöst. Dies geschieht aufgrund der WPF-Sicherheitsrichtlinie.

Deaktivieren von APTCA-Assemblys für teilweise vertrauenswürdige Clientanwendungen

Wenn verwaltete Assemblys im globalen Assemblycache (GAC) installiert werden, gelten sie als voll vertrauenswürdig, da die Benutzer*innen für ihre Installation eine explizite Berechtigung bereitstellen müssen. Da sie voll vertrauenswürdig sind, können sie nur von voll vertrauenswürdigen verwalteten Clientanwendungen verwendet werden. Damit sie von teilweise vertrauenswürdigen Anwendungen verwendet werden können, müssen sie mit [AllowPartiallyTrustedCallersAttribute](#) (APTCA) markiert werden. Mit diesem Attribut sollten nur Assemblys markiert werden, für die bei Tests nachgewiesen wurde, dass sie bei Ausführung in teilweiser Vertrauenswürdigkeit sicher sind.

Für eine APTCA-Assembly kann sich jedoch, nachdem sie im globalen Assemblycache installiert wurde, ein Sicherheitsrisiko zeigen. Nachdem ein Sicherheitsrisiko entdeckt wurde, können Assemblyherausgeber ein Sicherheitsupdate erstellen, um das Problem in vorhandenen Installationen zu beheben und Installationen zu schützen, die nach der Entdeckung des Problems erfolgen. Eine Option für ein solches Update besteht darin, die Assembly zu deinstallieren, was jedoch zum Versagen anderer voll vertrauenswürdiger Clientanwendungen führen kann, die diese Assembly verwenden.

WPF stellt einen Mechanismus bereit, mit dem eine APTCA-Assembly für teilweise vertrauenswürdige XBAPs deaktiviert werden kann, ohne die APTCA-Assembly zu deinstallieren.

Um eine APTCA-Assembly zu deaktivieren, müssen Sie einen speziellen Registrierungsschlüssel erstellen:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\policy\APTCA\  
<AssemblyFullName>, FileVersion=<AssemblyFileVersion>
```

Es folgt ein Beispiel:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\policy\APTCA\aptcagac,  
Version=1.0.0.0, Culture=neutral, PublicKeyToken=215e3ac809a0fea7,  
FileVersion=1.0.0.0
```

Dieser Schlüssel erstellt einen Eintrag für die APTCA-Assembly. Sie müssen außerdem einen Wert in diesem Schlüssel erstellen, der die Assembly aktiviert oder deaktiviert. Die Details des Werts sehen wie folgt aus:

- Wertname: **APTCA_FLAG**

- Werttyp: **REG_DWORD**
- Wertdaten: **1** = deaktivieren; **0** = aktivieren

Muss eine Assembly für teilweise vertrauenswürdige Clientanwendungen deaktiviert werden, können Sie ein Update schreiben, in dem der Registrierungsschlüssel und dessen Wert aktualisiert werden.

ⓘ Hinweis

.NET Framework-Kernassemblies sind von dieser Art der Deaktivierung nicht betroffen, da sie für die Ausführung von verwalteten Anwendungen erforderlich sind. Die Unterstützung zur Deaktivierung von APTCA-Assemblies zielt hauptsächlich auf Anwendungen von Drittanbietern ab.

Sandkastenverhalten für Loose XAML-Dateien

Loose XAML-Dateien sind Markup-XAML-Dateien, die von keiner CodeBehind-, Ereignishandler- oder anwendungsspezifischen Assembly abhängen. Wird direkt aus dem Browser zu Loose XAML-Dateien navigiert, werden diese entsprechend dem Standardberechtigungssatz für die Internetzone in eine Sicherheitssandbox geladen.

Das Sicherheitsverhalten ist jedoch anders, wenn aus einem [NavigationWindow](#) oder [Frame](#) in einer eigenständigen Anwendung zu Loose XAML-Dateien navigiert wird.

In beiden Fällen erbt die Loose XAML-Datei, zu der navigiert wird, die Berechtigungen ihrer Hostanwendung. Dieses Verhalten kann jedoch aus Sicherheitsgründen unerwünscht sein, besonders wenn eine Loose XAML-Datei von einer Entität erzeugt wurde, die entweder nicht vertrauenswürdig oder unbekannt ist. Dieser Inhaltstyp wird als *externer Inhalt* bezeichnet, und sowohl [Frame](#) als auch [NavigationWindow](#) können so konfiguriert werden, dass der Inhalt isoliert wird, wenn zu ihm navigiert wird. Die Isolation wird durch Festlegen der **SandboxExternalContent**-Eigenschaft auf TRUE erreicht, wie in den folgenden Beispielen für [Frame](#) und [NavigationWindow](#) gezeigt:

XAML

```
<Frame
  Source="ExternalContentPage.xaml"
  SandboxExternalContent="True">
</Frame>
```

XAML

```
<!-- Sandboxing external content using NavigationWindow-->
<NavigationWindow
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  Source="ExternalContentPage.xaml"
  SandboxExternalContent="True">
</NavigationWindow>
```

Mit dieser Einstellung wird der externe Inhalt in einen Prozess geladen, der von dem Prozess getrennt ist, der die Anwendung hostet. Dieser Prozess ist auf den Standardberechtigungssatz für die Internetzone beschränkt, wodurch er effizient von der Hostanwendung und dem Clientcomputer isoliert ist.

ⓘ Hinweis

Obwohl die Navigation zu Loose XAML-Dateien aus einem **NavigationWindow** oder **Frame** in einer eigenständigen Anwendung auf Grundlage der WPF-Browserhostinginfrastruktur implementiert wird und den PresentationHost-Prozess beinhaltet, ist die Sicherheitsstufe etwas niedriger, als wenn der Inhalt unter Windows Vista und Windows 7 direkt in Internet Explorer geladen wird (ebenfalls über PresentationHost). Dies liegt daran, dass eine eigenständige WPF-Anwendung, die einen Webbrowser verwendet, die zusätzliche „Geschützter Modus“-Sicherheitsfunktion von Internet Explorer nicht bereitstellt.

Ressourcen zum Entwickeln von WPF-Anwendungen, die die Sicherheit erhöhen

Nachfolgend sind einige zusätzliche Ressourcen zum Entwickeln von WPF-Anwendungen aufgeführt, mit denen Sie die Sicherheit erhöhen können:

Bereich	Resource
Verwalteter Code	Leitfäden zur Sicherheit von Anwendungen mit Mustern und Vorgehensweisen
CAS	Codezugriffssicherheit
ClickOnce	ClickOnce-Sicherheit und Bereitstellung
WPF	WPF-Sicherheit mit teilweiser Vertrauenswürdigkeit

Siehe auch

- [WPF-Sicherheit mit teilweiser Vertrauenswürdigkeit](#)

- WPF-Sicherheitsstrategie – Plattformsicherheit
- WPF-Sicherheitsstrategie – Sicherheitsentwicklung
- Leitfäden zur Sicherheit von Anwendungen mit Mustern und Vorgehensweisen
- Codezugriffssicherheit
- ClickOnce-Sicherheit und Bereitstellung
- XAML in WPF

WPF-Sicherheit mit teilweiser Vertrauenswürdigkeit

Artikel • 14.03.2024

Im Allgemeinen sollte der direkte Zugriff von Internetanwendungen auf wichtige Systemressourcen eingeschränkt werden, um böswillige Schäden zu vermeiden. Standardmäßig können HTML und clientseitige Skriptsprachen nicht auf wichtige Systemressourcen zugreifen. Da in einem Browser gehostete Windows Presentation Foundation WPF-Anwendungen aus dem Browser gestartet werden können, sollten für sie ähnliche Beschränkungen gelten. Um diese Beschränkungen durchzusetzen, verwendet WPF sowohl Code Access Security (CAS) als auch ClickOnce (weitere Informationen finden Sie unter [WPF-Sicherheitsstrategie – Plattformsicherheit](#)). Standardmäßig fordern im Browser gehostete Anwendungen den CAS-Berechtigungssatz für die Internetzone an, und zwar unabhängig davon, ob sie aus dem Internet, aus dem lokalen Intranet oder vom lokalen Computer gestartet wurden. Für Anwendungen, die nicht mit dem vollständigen, sondern einem eingeschränkten Berechtigungssatz ausgeführt werden, wird formuliert, dass sie mit teilweiser Vertrauenswürdigkeit ausgeführt werden.

WPF bietet viele Arten der Unterstützung, um sicherzustellen, dass bei teilweiser Vertrauenswürdigkeit möglichst viel Funktionalität sicher ausgeführt werden kann, und bietet zusammen mit CAS zusätzliche Unterstützung für die Programmierung mit teilweiser Vertrauenswürdigkeit.

Dieses Thema enthält folgende Abschnitte:

- [WPF-Funktionen für die Unterstützung von teilweiser Vertrauenswürdigkeit](#)
- [Programmieren für teilweise Vertrauenswürdigkeit](#)
- [Verwalten von Berechtigungen](#)

WPF-Funktionen für die Unterstützung von teilweiser Vertrauenswürdigkeit

Die folgende Tabelle enthält die allgemeinen Funktionen von Windows Presentation Foundation WPF, die in den Grenzen des Berechtigungssatzes für die Internetzone bedenkenlos verwendet werden können.

Tabelle 1: WPF-Funktionen, die bei teilweiser Vertrauenswürdigkeit sicher sind

Featurebereich	Funktion
Allgemein	<p>Browserfenster</p> <p>Zugriff auf Ursprungswebsite</p> <p>IsolatedStorage (beschränkt auf 512 KB)</p> <p>UIAutomation-Anbieter</p> <p>Befehle</p> <p>Eingabemethoden-Editoren (Input Method Editors, IMEs)</p> <p>Tablettstift und Freihandeingaben</p> <p>Simuliertes Drag & Drop mit Ereignissen für Mauseaufzeichnung und Bewegen</p> <p>OpenFileDialog</p> <p>XAML-Deserialisierung (über XamlReader.Load)</p>
Webintegration	<p>Download-Dialogfeld des Browsers</p> <p>Vom Benutzer initiierte Navigation auf oberster Ebene</p> <p>mailto:links</p> <p>Uniform Resource Identifier-Parameter</p> <p>HTTPWebRequest</p> <p>In einem IFRAME gehosteter WPF-Inhalt</p> <p>Hosten von HTML-Seiten aus derselben Website mithilfe eines Frames</p> <p>Hosten von HTML-Seiten aus derselben Website mithilfe von WebBrowser</p> <p>Webdienste (ASMX)</p> <p>Webdienste (über Windows Communication Foundation)</p> <p>Skripterstellung</p> <p>Dokumentobjektmodell</p>
Visuals	2D und 3D


Featurebereich	Funktion
	Animation Medien (Ursprungswebsite und domänenübergreifend) Imaging/Audio/Video
Aktuell gelesen	FlowDocuments XPS-Dokumente Eingebettete Schriftarten und Systemschriftarten CFF- und TrueType-Schriftarten
Bearbeitung läuft	Rechtschreibprüfung RichTextBox Unterstützung der Zwischenablage bei Nur-Text und Freihand Vom Benutzer initiiertes Einfügen Kopieren von markiertem Inhalt
Steuerelemente	Allgemeine Steuerelemente

In dieser Tabelle sind die WPF-Funktionen in abstrahierter Form aufgeführt. Um ausführlichere Informationen bereitzustellen, sind im Windows SDK die Berechtigungen dokumentiert, die für jedes Element in WPF erforderlich sind. Zusätzlich gibt es für die folgenden Funktionen ausführlichere Informationen hinsichtlich des Ausführens bei teilweiser Vertrauenswürdigkeit, wozu auch spezielle Aspekte gehören.

- XAML (siehe [XAML in WPF](#)).
- Popups (siehe [System.Windows.Controls.Primitives.Popup](#)).
- Drag & Drop (siehe [Übersicht über Drag & Drop](#))
- Zwischenablage (siehe [System.Windows.Clipboard](#)).
- Abbildung (siehe [System.Windows.Controls.Image](#)).
- Serialisierung (siehe [XamlReader.Load](#), [XamlWriter.Save](#)).
- Dialogfeld "Datei öffnen" (siehe [Microsoft.Win32.OpenFileDialog](#)).

In der folgende Tabelle sind die WPF-Funktionen erläutert, die in den Grenzen des Berechtigungssatzes für die Internetzone nicht bedenkenlos ausgeführt werden können.

Tabelle 2: WPF-Funktionen, die bei teilweiser Vertrauenswürdigkeit nicht sicher sind

 Tabelle erweitern

Featurebereich	Funktion
Allgemein	Fenster (anwendungsdefinierte Fenster und Dialogfelder)
	SaveFileDialog
	Dateisystem
	Zugriff auf die Registrierung
	Drag & Drop
	XAML-Serialisierung (über XamlWriter.Save)
	UIAutomation-Clients
	Zugriff auf das Quellcodefenster (HwndHost)
	Vollständige Sprachunterstützung
	Windows Forms-Interoperabilität
Visuals	Bitmapeffekte
	Bildcodierung
Bearbeitung läuft	Rich-Text-Format-Zwischenablage
	Vollständige XAML-Unterstützung

Programmieren für teilweise Vertrauenswürdigkeit

Bei XBAP-Anwendungen hat Code, der über den Standardberechtigungssatz hinausgeht, abhängig von der Sicherheitszone ein anderes Verhalten. In einigen Fällen erhält der Benutzer bei der Installation eine Warnung. Der Benutzer kann auswählen, ob die Installation fortgesetzt oder abgebrochen werden soll. In der folgenden Tabelle werden das Verhalten der Anwendung für jede Sicherheitszone und die erforderlichen Schritte für die volle Vertrauenswürdigkeit der Anwendung beschrieben.

Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

 **Tabelle erweitern**

Sicherheitszone	Verhalten	Erhalten der vollen Vertrauenswürdigkeit
Lokalem Computer	Automatisch volle Vertrauenswürdigkeit	Es ist keine Aktion erforderlich.
Intranet und vertrauenswürdige Websites	Eingabeaufforderung für volle Vertrauenswürdigkeit	Signieren Sie die XBAP mit einem Zertifikat, damit der Benutzer die Quelle in der Eingabeaufforderung sieht.
Internet	Schlägt fehl mit „Vertrauenswürdigkeit nicht gewährt“	Signieren Sie die XBAP mit einem Zertifikat.

Hinweis

Das in der vorherigen Tabelle beschriebene Verhalten gilt für vollständig vertrauenswürdige XBAPs, die nicht dem ClickOnce-Modell für vertrauenswürdige Bereitstellung entsprechen.

Bei Code, der die zulässigen Berechtigungen überschreitet, handelt es sich normalerweise um gemeinsamen Code, der von eigenständigen und im Browser gehosteten Anwendungen gemeinsam verwendet wird. CAS und WPF bieten mehrere Verfahren für die Verwaltung dieses Szenarios.

Erkennen von Berechtigungen mit CAS

In manchen Situationen kann freigegebener Code in Bibliothekassemblies sowohl von eigenständigen Anwendungen als auch von XBAPs verwendet werden. In diesen Fällen werden im Code möglicherweise Funktionen ausgeführt, die mehr Berechtigungen erfordern, als der Berechtigungssatz zulässt, der der Anwendung zugewiesen ist. Mithilfe der Microsoft .NET Framework-Sicherheit kann die Anwendung erkennen, ob sie eine

bestimmte Berechtigung hat. Insbesondere kann sie prüfen, ob sie eine bestimmte Berechtigung hat, indem sie die [Demand](#)-Methode für die Instanz der gewünschten Berechtigung aufruft. Dies wird im folgenden Beispiel gezeigt, das Code enthält, der überprüft, ob er eine Datei auf dem lokalen Datenträger speichern darf:

C#

```
using System.IO;
using System.IO.IsolatedStorage;
using System.Security;
using System.Security.Permissions;
using System.Windows;

namespace SDKSample
{
    public class FileHandling
    {
        public void Save()
        {
            if (IsPermissionGranted(new
FileIOPermission(FileIOPermissionAccess.Write, @"c:\newfile.txt")))
            {
                // Write to local disk
                using (FileStream stream = File.Create(@"c:\newfile.txt"))
                using (StreamWriter writer = new StreamWriter(stream))
                {
                    writer.WriteLine("I can write to local disk.");
                }
            }
            else
            {
                MessageBox.Show("I can't write to local disk.");
            }
        }

        // Detect whether or not this application has the requested
        permission
        bool IsPermissionGranted(CodeAccessPermission requestedPermission)
        {
            try
            {
                // Try and get this permission
                requestedPermission.Demand();
                return true;
            }
            catch
            {
                return false;
            }
        }
    }
}
```

C#

```
}  
}
```

Hat eine Anwendung nicht die gewünschte Berechtigung, löst der Aufruf von [Demand](#) eine Sicherheitsausnahme aus. Andernfalls wurde die Berechtigung erteilt.

`IsPermissionGranted` kapselt dieses Verhalten und gibt entsprechend `true` oder `false` zurück.

Sinnvolle Herabstufung der Funktionalität

Für Code, der aus verschiedenen Zonen ausgeführt werden, ist es sinnvoll, erkennen zu können, ob er die Berechtigung für seine auszuführenden Schritte hat. Während das Erkennen der Zone eine Geschichte ist, ist es weitaus besser, nach Möglichkeit eine Alternative für den Benutzer bereitzustellen. Beispielsweise ermöglicht eine Anwendung mit voller Vertrauenswürdigkeit Benutzern normalerweise, Dateien in beliebigen Speicherorten zu erstellen, während eine Anwendung mit teilweiser Vertrauenswürdigkeit Dateien nur in isoliertem Speicher erstellen kann. Wenn der Code zum Erstellen einer Datei in einer Assembly vorhanden ist, die sowohl von Anwendungen mit voller Vertrauenswürdigkeit (eigenständige Anwendungen) als auch von Anwendungen mit teilweiser Vertrauenswürdigkeit (im Browser gehostete Anwendungen) gemeinsam genutzt wird, und beide Anwendungen es Benutzern ermöglichen sollen, Dateien zu erstellen, muss der freigegebene Code erkennen, ob er mit voller oder teilweiser Vertrauenswürdigkeit ausgeführt wird, bevor eine Datei am entsprechenden Speicherort erstellt wird. Im folgenden Code werden beide Fälle veranschaulicht.

C#

```
using System.IO;  
using System.IO.IsolatedStorage;  
using System.Security;  
using System.Security.Permissions;  
using System.Windows;  
  
namespace SDKSample  
{  
    public class FileHandlingGraceful  
    {  
        public void Save()  
        {  
            if (IsPermissionGranted(new  
FileIOPermission(FileIOPermissionAccess.Write, @"c:\newfile.txt"))  
            {  

```

```

        // Write to local disk
        using (FileStream stream = File.Create(@"c:\newfile.txt"))
        using (StreamWriter writer = new StreamWriter(stream))
        {
            writer.WriteLine("I can write to local disk.");
        }
    }
    else
    {
        // Persist application-scope property to
        // isolated storage
        IsolatedStorageFile storage =
        IsolatedStorageFile.GetUserStoreForApplication();
        using (IsolatedStorageFileStream stream =
            new IsolatedStorageFileStream("newfile.txt",
        FileMode.Create, storage))
        using (StreamWriter writer = new StreamWriter(stream))
        {
            writer.WriteLine("I can write to Isolated Storage");
        }
    }
}

// Detect whether or not this application has the requested
permission
bool IsPermissionGranted(CodeAccessPermission requestedPermission)
{
    try
    {
        // Try and get this permission
        requestedPermission.Demand();
        return true;
    }
    catch
    {
        return false;
    }
}

```

C#

```

    }
}

```

In vielen Fällen sollte es Ihnen möglich sein, eine Alternative zu teilweiser Vertrauenswürdigkeit zu finden.

In einer kontrollierten Umgebung, beispielsweise einem Intranet, können benutzerdefinierte verwaltete Frameworks über die Clientbasis im globalen Assemblycache (GAC) installiert werden. Diese Bibliotheken können Code ausführen, der

volle Vertrauenswürdigkeit erfordert, und auf sie kann aus Anwendungen mit teilweiser Vertrauenswürdigkeit durch Verwenden von [AllowPartiallyTrustedCallersAttribute](#) verwiesen werden (weitere Informationen finden Sie unter [Sicherheit](#) und [WPF-Sicherheitsstrategie – Plattformsicherheit](#)).

Browserhosterkennung

Verwenden von CAS, um auf Berechtigungen zu prüfen, ist ein geeignetes Verfahren, wenn Sie auf Berechtigungsbasis prüfen müssen. Allerdings ist für dieses Verfahren das Abfangen von Ausnahmen als Teil der normalen Verarbeitung erforderlich, was grundsätzlich nicht empfehlenswert ist und Leistungsprobleme verursachen kann. Wenn Ihre XAML-Browseranwendung (XBAP) stattdessen nur innerhalb der Internetzone-Sandkasten ausgeführt wird, können Sie die [BrowserInteropHelper.IsBrowserHosted](#)-Eigenschaft verwenden, die für XAML-Browseranwendungen (XBAPs) true zurückgibt.

Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Hinweis

[IsBrowserHosted](#) erkennt nur, ob eine Anwendung in einem Browser ausgeführt wird, sie erkennt nicht, mit welchem Berechtigungssatz die Anwendung ausgeführt wird.

Verwalten von Berechtigungen

Standardmäßig werden XBAPs mit teilweiser Vertrauenswürdigkeit (Standardberechtigungssatz für die Internetzone) ausgeführt. Je nach Anforderungen der Anwendung ist es jedoch möglich, den Standardberechtigungssatz durch einen anderen Berechtigungssatz zu ersetzen. Werden beispielsweise XBAPs aus einem lokalen Intranet gestartet, kann für sie ein erweiterter Berechtigungssatz genutzt werden, der in der folgenden Tabelle gezeigt ist.

Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Tabelle 3: LocalIntranet- und Internetberechtigungen

[Tabelle erweitern](#)

Berechtigung	attribute	LocalIntranet	Internet
DNS	Zugriff auf DNS-Server	Ja	Nein
Umgebungsvariablen	Überwachungsdaten	Ja	Nein
Dateidialogfelder	Öffnen	Ja	Ja
Dateidialogfelder	Nicht eingeschränkt	Ja	Nein
Isolierte Speicherung	Assemblyisolation durch Benutzer	Ja	Nein
Isolierte Speicherung	Unbekannte isolation	Ja	Ja
Isolierte Speicherung	Unbegrenztes Benutzerkontingent	Ja	Nein
Medien	Sicherheit für Audio, Video und Bilder	Ja	Ja
Drucken	Standarddruck	Ja	Nein
Drucken	Sicheres Drucken	Ja	Ja
Spiegelung	Ausgabe	Ja	Nein
Sicherheit	Ausführen von verwaltetem Code	Ja	Ja
Sicherheit	Bestätigen von erteilten Berechtigungen	Ja	Nein
Benutzeroberfläche	Nicht eingeschränkt	Ja	Nein
Benutzeroberfläche	Sichere Fenster der obersten Ebene	Ja	Ja
Benutzeroberfläche	Eigene Zwischenablage	Ja	Ja
Webbrowser	Sichere Frame-Navigation zu HTML	Ja	Ja

ⓘ Hinweis

Ausschneiden und Einfügen ist bei teilweiser Vertrauenswürdigkeit nur bei Ausführen durch den Benutzer zulässig.

Müssen Sie Berechtigungen erhöhen, müssen Sie die Projekteinstellungen und das ClickOnce-Anwendungsmanifest ändern. Weitere Informationen finden Sie unter [Übersicht über WPF-XAML-Browseranwendungen](#). Möglicherweise sind auch die folgenden Dokumente hilfreich.

- [Mage.exe \(Tool zum Generieren und Bearbeiten von Manifesten\)](#)
- [MageUI.exe \(Tool zum Generieren und Bearbeiten von Manifesten, grafischer Client\)](#)
- [Sichern von ClickOnce-Anwendungen](#)

Wenn Ihre XBAP volle Vertrauenswürdigkeit erfordert, können Sie dieselben Tools verwenden, um die erforderlichen Berechtigungen zu erhöhen. Allerdings erhält eine XBAP nur dann volle Vertrauenswürdigkeit, wenn sie auf dem lokalen Computer installiert und von diesem Computer, aus dem Intranet oder über eine URL gestartet wird, die in den vertrauenswürdigen oder zugelassenen Websites des Browsers aufgelistet ist. Wird die Anwendung aus dem Intranet oder einer vertrauenswürdigen Website installiert, wird der Benutzer durch die Standard-ClickOnce-Eingabeaufforderung über die erhöhten Berechtigungen informiert. Der Benutzer kann auswählen, ob die Installation fortgesetzt oder abgebrochen werden soll.

Alternativ können Sie das ClickOnce-Modell für vertrauenswürdige Bereitstellung für eine Bereitstellung mit voller Vertrauenswürdigkeit aus einer beliebigen Sicherheitszone verwenden. Weitere Informationen finden Sie unter [Überblick über die Bereitstellung vertrauenswürdiger Anwendungen](#) und [Sicherheit](#).

Siehe auch

- [Sicherheit](#)
- [WPF-Sicherheitsstrategie – PlattformSicherheit](#)
- [WPF-Sicherheitsstrategie – Sicherheitsentwicklung](#)



**Zusammenarbeit auf
GitHub**

Die Quelle für diesen Inhalt
finden Sie auf GitHub, wo Sie

.NET

**Feedback zu .NET Desktop
feedback**

.NET Desktop feedback ist ein Open
Source-Projekt. Wählen Sie einen

auch Issues und Pull Requests erstellen und überprüfen können. Weitere Informationen finden Sie in unserem [Leitfaden für Mitwirkende](#).

Link aus, um Feedback zu geben:

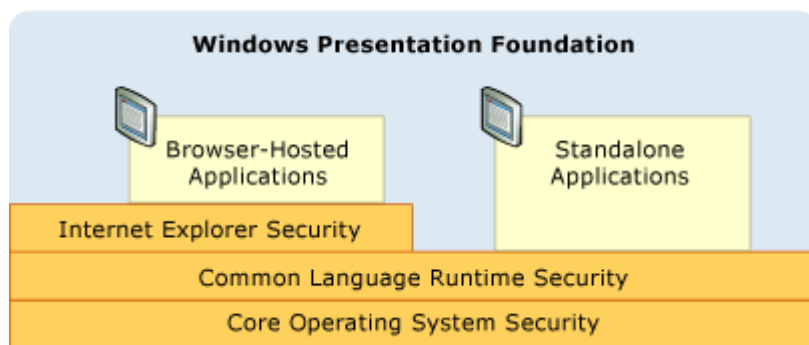
 [Problem in der Dokumentation öffnen](#)

 [Abgeben von Produktfeedback](#)

WPF-Sicherheitsstrategie – Plattformsicherheit

Artikel • 14.03.2024

Windows Presentation Foundation (WPF) stellt nicht nur eine Vielzahl von Sicherheitsdiensten bereit, sondern nutzt auch die Sicherheitsfeatures der zugrundeliegenden Plattform, wozu das Betriebssystem, die CLR und Internet Explorer gehören. Im Zusammenspiel stellen diese Ebenen für WPF ein leistungsfähiges Modell für tiefgreifende, vorbeugende Sicherheitsmaßnahmen (Defense-in-Depth-Modell) bereit, das eine einzelne Fehlerquelle zu vermeiden sucht, wie aus der folgenden Abbildung hervorgeht:



Im weiteren Verlauf dieses Themas werden die Features auf den einzelnen Ebenen erläutert, die besonders WPF betreffen.

Sicherheit des Betriebssystems

Der Kern von Windows enthält verschiedene Sicherheitsfeatures, die die Sicherheitsgrundlage für alle Windows-Anwendungen bilden, auch für die mit WPF erstellten Anwendungen. In diesem Thema wird der Umfang dieser Sicherheitsfeatures behandelt, die für WPF wichtig sind. Erläutert wird auch die Integration mit WPF, um weitere tiefgreifende Vorbeugungsmaßnahmen bereitzustellen.

Microsoft Windows XP Service Pack 2 (SP2)

Neben einer allgemeinen Überprüfung und Stärkung von Windows gibt es drei wichtige Funktionen von Windows XP SP2, die in diesem Thema behandelt werden:

- /GS-Kompilierung
- Microsoft Windows Update.

/GS-Kompilierung

Windows XP SP2 bietet Schutz durch das erneute Kompilieren vieler zentraler Systembibliotheken, einschließlich aller WPF-Abhängigkeiten wie die CLR, um das Risiko von Pufferüberläufen zu verringern. Dies wird mit dem /GS-Parameter und dem C/C++-Befehlszeilencompiler erreicht. Obwohl Pufferüberläufe ausdrücklich vermieden werden sollten, ist die Kompilierung mit /GS ein Beispiel für eine tiefgreifende Verteidigungsmaßnahme gegen potenzielle Sicherheitslücken, die versehentlich oder böswillig durch Pufferüberläufe erzeugt werden.

In der Vergangenheit waren Pufferüberläufe die Ursache für viele Sicherheitslücken mit gravierenden Folgen. Ein Pufferüberlauf tritt auf, wenn ein Angreifer die Sicherheitsanfälligkeit eines Codes nutzt, die das Einfügen von schädlichem Code ermöglicht, der über die Begrenzungen eines Puffers schreibt. Ein Angreifer kann dann durch Überschreiben der Rückgabeadresse einer Funktion den Prozess, in dem der Code ausgeführt wird, hacken und die Ausführung des Angreifercodes auslösen. Daraus entsteht bösartiger Code, der beliebigen Code mit den gleichen Berechtigungen wie der gehackte Prozess ausführt.

Auf hoher Ebene schützt das -GS-Compilerflag vor einigen möglichen Pufferüberläufen, indem es zum Schutz der Rückgabeadresse einer Funktion, die über lokale Zeichenfolgepuffer verfügt, ein spezielles Sicherheitscookie einschleust. Nach der Rückgabe einer Funktion wird das Sicherheitscookie mit seinem vorherigen Wert verglichen. Hat sich der Wert geändert, ist möglicherweise ein Pufferüberlauf aufgetreten, und der Prozess wird mit einer Fehlerbedingung beendet. Das Beenden des Prozesses verhindert die Ausführung von potenziell bösartigem Code. Ausführlichere Informationen finden Sie unter [-GS \(Puffersicherheitsüberprüfung\)](#).

WPF wird mit dem /GS-Flag kompiliert, um eine weitere Schutzebene für WPF-Anwendungen bereitzustellen.

Windows Vista

WPF-Benutzer mit Windows Vista profitieren von zusätzlichen Verbesserungen des Betriebssystems in puncto Sicherheit, z. B. Benutzerkontensteuerung, Integritätsprüfungen für Code und Rechteisolierung.

Benutzerkontensteuerung (User Account Control, UAC)

Derzeit neigen Benutzer von Windows dazu, mit Administratorrechten zu arbeiten, da diese für viele Anwendungen (entweder für die Installation und/oder die Ausführung)

erforderlich sind. Das Schreiben von Standardanwendungseinstellungen in die Registrierung ist nur ein Beispiel.

Das Arbeiten mit Administratorrechten bedeutet im Grunde, dass Anwendungen von Prozessen ausgeführt werden, denen Administratorrechte gewährt werden. Das hat Auswirkungen auf die Sicherheit, denn bösartiger Code, der zum Hacken eines mit Administratorrechten ausgeführten Prozesses verwendet wird, übernimmt automatisch dessen Berechtigungen, einschließlich des Zugriffs auf kritische Systemressourcen.

Anwendungen nur mit den unbedingt erforderlichen Berechtigungen auszuführen, ist eine Möglichkeit, sich vor dieser Sicherheitsbedrohung zu schützen. Dieses sogenannte „Prinzip der geringsten Rechte“ ist ein zentrales Feature des Betriebssystems Windows. Diese als Benutzerkontensteuerung bezeichnete Funktion wird von Windows auf zweierlei Weise verwendet:

- Die meisten Anwendungen werden standardmäßig mit den Rechten der Benutzerkontensteuerung ausgeführt, selbst wenn der Benutzer ein Administrator ist; nur Anwendungen, die Administratorrechte benötigen, werden mit Administratorrechten ausgeführt. Um mit Administratorrechten ausgeführt zu werden, müssen Anwendungen entweder in ihrem Anwendungsmanifest oder als Eintrag in der Sicherheitsrichtlinie besonders gekennzeichnet werden.
- Bereitstellen von Kompatibilitätslösungen wie die Virtualisierung. Viele Anwendungen versuchen beispielsweise, in eingeschränkte Speicherorte wie "C:\Programme" zu schreiben. Für Anwendungen, die über die Benutzerkontensteuerung ausgeführt werden, steht ein alternativer benutzerbezogener Speicherort zur Verfügung, der für Schreibvorgänge keine Administratorrechte erfordert. Für Anwendungen, die über die Benutzerkontensteuerung ausgeführt werden, wird "C:\Programme" virtualisiert. Dadurch wird der Anschein erweckt, dass die Anwendungen in dieses Verzeichnis schreiben, obwohl sie stattdessen in den alternativen, benutzerbezogenen Speicherort schreiben. Dank dieser Art von Kompatibilität kann das Betriebssystem viele Anwendungen ausführen, die zuvor nicht mit der Benutzerkontensteuerung ausgeführt werden konnten.

Integritätsprüfungen für Code

Windows Vista enthält strengere Integritätsprüfungen für Code, um zu verhindern, dass bösartiger Code zur Lade-/Laufzeit in Systemdateien oder den Kernel eingeschleust wird. Dies geht über den Schutz der Systemdateien hinaus.

Prozess mit eingeschränkten Rechten für im Browser gehostete Anwendungen

Im Browser gehostete WPF-Anwendungen werden in der Sicherheitssandbox der Internetzone ausgeführt. Die WPF-Integration in Microsoft Internet Explorer erweitert diesen Schutz durch zusätzliche Unterstützung.

Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Da XAML-Browseranwendungen (XBAPs) im Allgemeinen durch den Berechtigungssatz für die Internetzone in einer Sandbox ausgeführt werden, beeinträchtigt das Entfernen dieser Berechtigungen XAML-Browseranwendungen (XBAPs) aus Kompatibilitätsperspektive nicht. Stattdessen wird eine zusätzliche Defense-in-Depth-Ebene geschaffen. Wenn eine in einem Sicherheitssandkasten ausgeführte Anwendung in der Lage ist, andere Ebenen anzugreifen und den Prozess zu hacken, verfügt auch der Prozess nur über eingeschränkte Berechtigungen.

Weitere Informationen finden Sie unter [Verwenden eines Benutzerkontos mit den geringsten Berechtigungen](#).

Common Language Runtime-Sicherheit

Die Common Language Runtime (CLR) bietet eine Reihe wichtiger Vorteile in puncto Sicherheit. Dazu zählen Validierung und Überprüfung, Codezugriffssicherheit (CAS) sowie die sicherheitsrelevante Methode.

Validierung und Prüfung

Assemblyisolation und Integrität wird von der CLR über einen Validierungsprozess bereitgestellt. Mit der CLR-Validierung wird sichergestellt, dass Assemblys isoliert werden. Dazu wird deren PE-Dateiformat (Portable Executable) für Adressen validiert, die auf eine Stelle außerhalb der Assembly verweisen. Die CLR-Validierung validiert auch die Integrität der in eine Assembly eingebetteten Metadaten.

Um die Typsicherheit zu gewährleisten, häufige Sicherheitsprobleme (z. B. Pufferüberläufe) zu vermeiden und durch die Isolation von Unterprozessen die Ausführung in einem Sicherheitssandbox zu ermöglichen, setzt die CLR-Sicherheit das Prüfprinzip ein.

Verwaltete Anwendungen werden in Microsoft Intermediate Language (MSIL) kompiliert. Wenn Methoden in einer verwalteten Anwendung ausgeführt werden, wird die MSIL über Just-in-Time (JIT)-Kompilierung in systemeigenen Code kompiliert. Die JIT-Kompilierung enthält einen Prüfprozess mit vielen Sicherheits- und Stabilitätsregeln, die sicherstellen, dass der Code keine der folgenden Eigenschaften aufweist:

- Verletzen von Typverträgen
- Verursachen von Pufferüberläufen
- Unkontrollierter Zugriff auf den Arbeitsspeicher

Verwalteter Code, der den Prüfregeln nicht entspricht, darf nur dann ausgeführt werden, wenn er als vertrauenswürdiger Code gilt.

Der Vorteil des prüfbaren Codes ist einer der wichtigsten Gründe, warum WPF auf .NET Framework aufbaut. Mit zunehmender Verwendung prüfbaren Codes sinkt die Wahrscheinlichkeit, dass potenzielle Schwachstellen ausgenutzt werden, ganz erheblich.

Codezugriffssicherheit

Ein Clientcomputer stellt eine Vielzahl von Ressourcen bereit, auf die eine verwaltete Anwendung zugreifen kann. Dazu zählen u. a. Dateisystem, Registrierung, Druckdienste, Benutzeroberfläche, Reflektion und Umgebungsvariablen. Bevor eine verwaltete Anwendung auf eine Ressource auf einem Clientcomputer zugreifen kann, benötigt sie dafür die .NET Framework-Berechtigung. Eine Berechtigung in CAS ist eine Unterklasse von [CodeAccessPermission](#). CAS implementiert eine Unterklasse für jede Ressource, auf die verwaltete Anwendungen zugreifen können.

Der Satz von Rechten, den CAS einer verwalteten Anwendung beim Starten gewährt, wird als Berechtigungssatz bezeichnet. Er richtet sich nach den von der Anwendung bereitgestellten Nachweisen. Bei WPF-Anwendungen wird als Nachweis der Speicherort oder die Zone bereitgestellt, von dem bzw. der die Anwendungen gestartet werden. CAS identifiziert die folgenden Zonen:

- **Arbeitsplatz.** Anwendungen, die auf dem Clientcomputer gestartet werden (voll vertrauenswürdig).

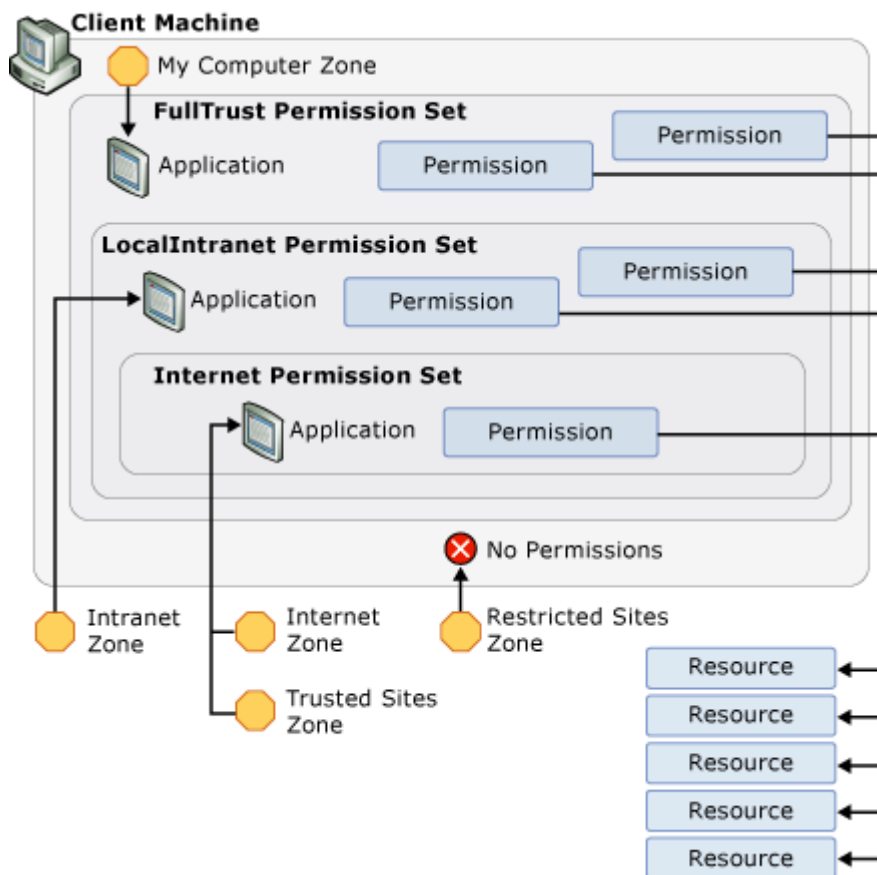
- **Lokales Intranet.** Anwendungen, die aus dem Intranet gestartet werden (in gewisser Weise vertrauenswürdig).
- **Internet.** Anwendungen, die aus dem Internet gestartet werden (wenig vertrauenswürdig).
- **Vertrauenswürdige Sites.** Anwendungen, die von einem Benutzer als vertrauenswürdig identifiziert wurden (wenig vertrauenswürdig).
- **Nicht vertrauenswürdige Sites.** Anwendungen, die von einem Benutzer als nicht vertrauenswürdig identifiziert wurden (nicht vertrauenswürdig).

Für jede dieser Zonen stellt CAS einen vordefinierten Berechtigungssatz bereit, der die Berechtigungen enthält, die der jeweils zugeordneten Vertrauensebene entsprechen. Dazu zählen unter anderem folgende Einstellungen:

- **FullTrust.** Für Anwendungen, die aus der Zone **Arbeitsplatz** gestartet werden. Alle möglichen Berechtigungen werden gewährt.
- **LocalIntranet.** Für Anwendungen, die aus der Zone **Lokales Intranet** gestartet werden. Für einen moderaten Zugriff auf die Ressourcen des Clientcomputers wird eine Teilmenge von Berechtigungen gewährt. Dazu zählen isolierter Speicher, uneingeschränkter Zugriff auf die Benutzeroberfläche und Dateidialogfelder, eingeschränkte Reflektion sowie eingeschränkter Zugriff auf Umgebungsvariablen. Berechtigungen für kritische Ressourcen wie die Registrierung werden nicht bereitgestellt.
- **Internet.** Für Anwendungen, die aus der Zone **Internet** oder **Vertrauenswürdige Sites** gestartet werden. Für einen eingeschränkten Zugriff auf die Ressourcen des Clientcomputers wird eine Teilmenge von Berechtigungen gewährt. Dazu zählen isolierter Speicher, Öffnen von Dateien sowie eingeschränkter Zugriff auf die Benutzeroberfläche. Im Grunde isolieren diese Berechtigungssätze die Anwendungen vom Clientcomputer.

Anwendungen, die aus der Zone **Nicht vertrauenswürdige Sites** stammen, gewährt CAS überhaupt keine Berechtigungen. Daher gibt es für diese Anwendungen keinen vordefinierten Berechtigungssatz.

Die folgende Abbildung veranschaulicht die Beziehung zwischen Zonen, Berechtigungssätzen, Berechtigungen und Ressourcen:



Die Einschränkungen der Sicherheitssandbox der Internetzone gelten in gleicher Weise für jeden Code, den eine XBAP aus einer Systembibliothek importiert (einschließlich WPF). Dadurch wird sichergestellt, dass jedes Bit des Codes (selbst WPF) gesperrt wird. Unglücklicherweise muss eine XBAP jedoch, um ausgeführt zu werden, Funktionalität ausführen, die mehr Berechtigungen erfordert, als im Rahmen der Sicherheitssandbox der Internetzone gewährt werden.

⚠️ Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Nehmen Sie beispielsweise eine XBAP-Anwendung, die die folgende Seite enthält:

C#

```
FileIOPermission fp = new FileIOPermission(PermissionState.Unrestricted);
fp.Assert();

// Perform operation that uses the assert
```

```
// Revert the assert when operation is completed
CodeAccessPermission.RevertAssert();
```

Zum Ausführen dieser XBAP muss der zugrunde liegende WPF-Code mehr Funktionalität ausführen, als der aufrufenden XBAP zur Verfügung steht. Dazu zählen:

- Erstellen eines Fensterhandles (HWND) für das Rendering
- Verteilen von Nachrichten
- Laden der Schriftart Tahoma

Unter dem Gesichtspunkt der Sicherheit wäre die Gewährung des direkten Zugriffs auf diese Vorgänge für die im Sicherheitssandbox ausgeführte Anwendung eine Katastrophe.

Zum Glück bietet WPF einen Ausweg aus dieser Lage, indem diesen Vorgängen die Ausführung mit erhöhten Rechten im Namen der in der Sicherheitssandbox ausgeführten Anwendung erlaubt wird. Während alle WPF-Vorgänge anhand der eingeschränkten Sicherheitsberechtigungen der Internetzone der XBAP-Anwendungsdomäne überprüft werden, wird WPF (wie anderen Systembibliotheken) ein Berechtigungssatz gewährt, der alle möglichen Berechtigungen enthält.

Dies setzt voraus, dass WPF erhöhte Rechte eingeräumt werden und gleichzeitig verhindert wird, dass diese Berechtigungen vom Berechtigungssatz der Internetzone der Hostanwendungsdomäne gesteuert werden.

WPF verwendet dazu für eine Berechtigung die **Assert**-Methode. Der folgende Code beschreibt die Vorgehensweise.

```
C#

FileIOPermission fp = new FileIOPermission(PermissionState.Unrestricted);
fp.Assert();

// Perform operation that uses the assert

// Revert the assert when operation is completed
CodeAccessPermission.RevertAssert();
```

Mit **Assert** wird im Grunde verhindert, dass die für WPF erforderlichen uneingeschränkten Berechtigungen durch die Berechtigungen der Internetzone der XBAP eingeschränkt werden.

Aus Plattformsicht ist WPF für die richtige Anwendung von **Assert** verantwortlich; eine falsche Verwendung von **Assert** könnte schädlichem Code die Möglichkeit geben, die

Rechte zu erhöhen. Daher ist es wichtig, **Assert** nur bei Bedarf aufzurufen und dabei sicherzustellen, dass die Einschränkungen der Sicherheitssandbox unverändert erhalten bleiben. Beispielsweise darf im Sicherheitssandkasten ausgeführter Code zwar keine beliebigen Dateien öffnen, aber Schriftarten verwenden. WPF bietet durch den Aufruf von **Assert** den in der Sicherheitssandbox ausgeführten Anwendungen die Möglichkeit, die Schriftartfunktion zu verwenden, und WPF liest dann im Namen dieser Anwendung in der Sandbox die Dateien, die bekanntermaßen diese Schriftarten enthalten.

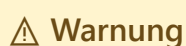
ClickOnce-Bereitstellung

ClickOnce ist eine umfassende Bereitstellungstechnologie, die in .NET Framework enthalten ist und sich in Visual Studio integrieren lässt (detaillierte Informationen finden Sie unter [ClickOnce-Sicherheit und -Bereitstellung](#)). Eigenständige WPF-Anwendungen können mit ClickOnce bereitgestellt werden, während im Browser gehostete Anwendungen mit ClickOnce bereitgestellt werden müssen.

Mit ClickOnce bereitgestellte Anwendungen werden durch Codezugriffssicherheit (CAS) mit einer weiteren Sicherheitsebene ausgestattet; im Grunde fordern mit ClickOnce bereitgestellte Anwendungen nur die von ihnen benötigten Berechtigungen an. Diese Berechtigungen werden nur gewährt, wenn sie nicht höher sind als die Berechtigungen im Berechtigungssatz für die Zone, von der die Anwendung bereitgestellt wird. Durch die Reduzierung des Berechtigungssatzes auf die benötigten Berechtigungen (selbst wenn diese niedriger sind als die Berechtigungen im Berechtigungssatz der Startzone) wird die Anzahl der Ressourcen, auf welche die Anwendung zugreifen kann, auf ein absolutes Minimum reduziert. Dadurch wird der mögliche Schaden für den Clientcomputer reduziert, falls die Anwendung gehackt wird.

Sicherheitsrelevante Methode

Für den WPF-Code, der mithilfe von Berechtigungen die Sicherheitssandbox der Internetzone für XBAP-Anwendungen aktiviert, muss stets das höchstmögliche Maß an Sicherheitsüberwachung und -kontrolle gelten. Um diese Anforderung zu erleichtern, stellt .NET Framework neue Unterstützung für die Verwaltung von Code bereit, der Rechte erhöht. Insbesondere die CLR ermöglicht es Ihnen, Code zu identifizieren, der Rechte erhöht, und diesen mit [SecurityCriticalAttribute](#) zu kennzeichnen. Nicht mit [SecurityCriticalAttribute](#) gekennzeichneter Code wird mit dieser Methode *transparent*. Umgekehrt wird verhindert, dass verwalteter Code, der nicht mit dem [SecurityCriticalAttribute](#) gekennzeichnet ist, die Rechte erhöht.



Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Die sicherheitsrelevante Methode ermöglicht die Organisation von WPF-Code, der Rechte bis in den *sicherheitskritischen Kernel* erhöht, während der Rest transparent dargestellt wird. Durch das Isolieren des sicherheitskritischen Codes kann sich das WPF-Entwicklungsteam auf eine zusätzliche Sicherheitsanalyse und Quellcodekontrolle für den sicherheitskritischen Kernel konzentrieren, die weit über die Standardsicherheitsmaßnahmen hinausgehen (weitere Informationen finden Sie unter [WPF-Sicherheitsstrategie – Sicherheitsentwicklung](#)).

Beachten Sie, dass .NET Framework vertrauenswürdigen Code erlaubt, die Sicherheitssandbox der Internetzone für die XBAP zu erweitern, indem es Entwicklern gestattet wird, verwaltete Assemblys zu schreiben, die mit [AllowPartiallyTrustedCallersAttribute](#) (APTCA) gekennzeichnet und im globalen Assemblycache (GAC) des Benutzers bereitgestellt werden. Das Kennzeichnen einer Assembly mit dem APTCA-Attribut ist ein extrem kritischer Sicherheitsvorgang, da jeder Code, auch bösartiger Code aus dem Internet, diese Assembly aufrufen kann. Bei diesem Vorgang sind äußerste Vorsicht und die Verwendung bewährter Methoden ein absolutes Muss. Die Benutzer müssen zuerst angeben, dass sie dieser Software vertrauen, bevor sie installiert werden kann.

Microsoft Internet Explorer-Sicherheit

Neben Funktionen zum Reduzieren von Sicherheitsproblemen und zum Vereinfachen der Sicherheitskonfiguration enthält Microsoft Internet Explorer 6 (SP2) verschiedene Features, die die Sicherheit für die Benutzer von XAML-Browseranwendungen (XBAPs) noch weiter erhöhen. Hauptziel dieser Features ist eine bessere Kontrolle des Benutzers über seine Arbeit im Browser.

Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Vor IE6 SP2 mussten Benutzer mit Folgendem rechnen:

- Zufällige Popupfenster
- Verwirrende Skriptumleitung
- Zahlreiche Sicherheitsdialogfelder auf manchen Websites

In einigen Fällen versuchten nicht vertrauenswürdige Websites, Benutzer durch Spoofing der Installationsbenutzeroberfläche (UI) oder wiederholtes Anzeigen eines Microsoft ActiveX-Installationsdialogfelds zu täuschen, obwohl der Benutzer den Vorgang bereits abgebrochen hatte. Mit diesen Techniken ist es möglich, eine große Anzahl von Benutzern zu schlechten Entscheidungen zu verleiten, die zur Installation von Spyware-Anwendungen führen.

IE6 SP2 enthält mehrere Features, um diese Art von Problemen rund um das Prinzip der Benutzerinitiierung zu minimieren. IE6 SP2 erkennt, wenn ein Benutzer vor einer Aktion auf einen Link oder ein Seitenelement geklickt hat (als *Benutzerinitiierung* bezeichnet), und behandelt den Vorgang anders als eine ähnliche Aktion, die von dem Skript auf der Seite ausgeführt wird. Beispielsweise enthält IE6 SP2 einen **Popupblocker**, der erkennt, wenn ein Benutzer auf eine Schaltfläche klickt, bevor auf der Seite ein Popup erstellt wird. Auf diese Weise kann IE6 SP2 die meisten unschädlichen Popups zulassen und gleichzeitig die Popups verhindern, die Benutzer weder anfordern noch wünschen. Blockierte Popups werden unter der neuen **Informationsleiste** erfasst. Dadurch kann der Benutzer die Blockierung überschreiben und das Popup anzeigen.

Die Benutzerinitiierungslogik gilt auch für die Sicherheitshinweise beim **Öffnen/Speichern**. ActiveX-Installationsdialogfelder werden immer unter der Informationsleiste erfasst, sofern es sich nicht um das Upgrade eines bereits installierten Steuerelements handelt. All diese Maßnahmen sorgen für eine Benutzererfahrung mit mehr Sicherheit und Kontrolle und bieten Schutz vor Websites, die den Benutzer belästigen und zur Installation von unerwünschter oder schädlicher Software verleiten wollen.

Diese Funktionen schützen auch die Kunden, die mit IE6 SP2 zu Websites navigieren, auf denen sie WPF-Anwendungen herunterladen und installieren können. Hauptgrund dafür ist die bessere Benutzererfahrung, die IE6 SP2 ermöglicht und die weniger Gelegenheit zum Installieren schädlicher oder undurchsichtiger Anwendungen bietet, und zwar ungeachtet der zum Erstellen verwendeten Technologie (einschließlich WPF). WPF erweitert durch die Verwendung von ClickOnce diese Schutzmaßnahmen noch und vereinfacht damit das Herunterladen der zugehörigen Anwendungen über das Internet. Da XAML-Browseranwendungen (XBAPs) innerhalb einer Sicherheitssandbox der Internetzone ausgeführt werden, können sie nahtlos gestartet werden. Andererseits

setzen eigenständige WPF-Anwendungen vollständige Vertrauenswürdigkeit voraus, um ausgeführt werden zu können. Für diese Anwendungen zeigt ClickOnce beim Startvorgang ein Sicherheitsdialogfeld an, um auf die Verwendung der zusätzlichen Sicherheitsanforderungen der Anwendung hinzuweisen. Da dies vom Benutzer initiiert werden muss, unterliegt der Vorgang außerdem der Benutzerinitiiierungslogik und kann abgebrochen werden.

Im Rahmen unseres stetigen Engagements für die Sicherheit enthält Internet Explorer 7 die Sicherheitsfunktionen von IE6 SP2 und erweitert diese noch.

Siehe auch

- [Codezugriffssicherheit](#)
- [Security](#)
- [WPF-Sicherheit mit teilweiser Vertrauenswürdigkeit](#)
- [WPF-Sicherheitsstrategie – Sicherheitsentwicklung](#)

Zusammenarbeit auf GitHub

Die Quelle für diesen Inhalt finden Sie auf GitHub, wo Sie auch Issues und Pull Requests erstellen und überprüfen können. Weitere Informationen finden Sie in unserem [Leitfaden für Mitwirkende](#).

 .NET

Feedback zu .NET Desktop feedback

.NET Desktop feedback ist ein Open Source-Projekt. Wählen Sie einen Link aus, um Feedback zu geben:

 [Problem in der Dokumentation öffnen](#)

 [Abgeben von Produktfeedback](#)

WPF-Sicherheitsstrategie – Sicherheitsentwicklung

Artikel • 14.03.2024

Trustworthy Computing ist eine Microsoft-Initiative, die sicherstellen soll, dass sicherer Code entwickelt wird. Ein wichtiges Element der vertrauenswürdigen Computing-Initiative ist der Microsoft Security Development Lifecycle (SDL). Der SDL ist ein Entwicklungsverfahren, das in Verbindung mit standardmäßigen Entwicklungsprozessen verwendet wird, um die Erstellung von sicherem Code zu erleichtern. Der SDL besteht aus zehn Phasen, die bewährte Methoden mit Formalisierung, Messbarkeit und zusätzlichen Strukturen kombinieren, darunter:

- Analyse des Sicherheitsentwurfs
- Qualitätsprüfungen mithilfe von Tools
- Penetrationstests
- Abschließende Sicherheitsüberprüfung
- Verwaltung der Produktsicherheit nach der Veröffentlichung

WPF im Einzelnen

Das WPF-Entwicklungsteam wendet den SDL an und ist auch für dessen Erweiterung zuständig; diese Kombination beinhaltet die folgenden Schlüsselaspekte:

[Erstellen von Gefahrenmodellen](#)

[Sicherheitsanalyse und Bearbeitungstools](#)

[Testverfahren](#)

[Verwaltung von sicherheitsrelevantem Code](#)

Erstellen von Gefahrenmodellen

Das Erstellen von Gefahrenmodellen ist eine Kernkomponente des SDL und wird zum Erstellen von Systemprofilen verwendet, um potenzielle Sicherheitsanfälligkeiten zu bestimmen. Sobald die Schwachstellen identifiziert sind, kann mithilfe der Gefahrenmodelle zudem sichergestellt werden, dass entsprechende Maßnahmen zur Risikominderung zum Einsatz kommen.

Im Überblick umfasst die Erstellung von Gefahrenmodellen die folgenden Hauptschritte, hier am Beispiel eines Lebensmittelmarkts verdeutlicht:

1. **Ressourcenermittlung.** Zu den Ressourcen eines Lebensmittelmarkts können etwa die Mitarbeiter, ein Tresor, Registrierkassen und der Warenbestand gehören.
2. **Aufzählen der Einstiegspunkte.** Die Einstiegspunkte eines Lebensmittelmarkts können etwa die Vorder- und Hintertüren, Fenster, das Ladedeck und die Öffnungen der Klimaanlage zählen.
3. **Untersuchen von Angriffen auf Ressourcen mithilfe der Einstiegspunkte.** Ein mögliches Angriffsszenario könnte auf die Ressource *Tresor* des Lebensmittelmarkts über den Einstiegspunkt *Öffnungen der Klimaanlage* abzielen; die Klimaanlage könnte demontiert werden, um das Herausziehen des Tresors aus dem Lebensmittelmarkt durch die Lüftungsschächte zu ermöglichen.

Die Erstellung von Gefahrenmodellen wird in WPF durchgängig angewendet und schließt folgende Punkte ein:

- Die Weise, in der der XAML-Parser Dateien liest, Text zu entsprechenden Objektmodellklassen zuordnet und den tatsächlichen Code erstellt.
- Wie ein Fensterhandle (hWnd) erstellt wird, Nachrichten sendet und zum Rendern der Inhalte eines Fensters verwendet wird.
- Wie die Datenbindung Ressourcen erhält und mit dem System interagiert.

Diese Gefahrenmodelle sind wichtig, um die Anforderungen an den Sicherheitsentwurf und die Maßnahmen der Gefahrenabwehr während des Entwicklungsprozesses zu bestimmen.

Quellcodeanalyse und Bearbeitungstools

Über die manuellen Elemente der Codesicherheitsprüfung im SDL hinaus verwendet das WPF-Team eine Reihe von Tools für die Quellcodeanalyse und die zugeordneten Bearbeitungsschritte, um Sicherheitsschwachstellen zu vermindern. Es wird eine breite Palette von Tools für den Quellcode verwendet, darunter die folgenden:

- **FXCop:** Findet häufige Sicherheitsprobleme in verwaltetem Code, beginnend mit Vererbungsregeln über die Verwendung der Zugriffssicherheit im Code bis hin zum sicheren Zusammenwirken mit nicht verwaltetem Code. Weitere Informationen finden Sie unter [FXCop](#).

- **Prefix/Prefast:** Findet Schwachstellen der Sicherheit und häufige Sicherheitsprobleme in nicht verwaltetem Code, wie etwa Pufferüberläufe, Probleme bei Formatzeichenfolgen und Fehlerprüfung.
- **Gesperrte APIs:** Durchsucht den Quellcode, um die versehentliche Verwendung von Funktionen zu erkennen, die für Sicherheitsprobleme bekannt sind, wie etwa `strcpy`. Nach der Erkennung werden diese Funktionen durch Alternativen ersetzt, die mehr Sicherheit bieten.

Testverfahren

WPF verwendet eine Reihe von Techniken zum Testen der Sicherheit, darunter:

- **Whiteboxtests:** Tester untersuchen den Quellcode und entwickeln dann Exploittests.
- **Blackboxtests:** Tester suchen Sicherheitsexploits, indem Sie APIs und Funktionen untersuchen und dann versuchen, das Produkt anzugreifen.
- **Zurückverfolgen von Sicherheitsproblemen anderer Produkte:** Sofern sie relevant sind, werden Sicherheitsprobleme von verwandten Produkten getestet. Beispielsweise wurden entsprechende Varianten von nahezu 60 Sicherheitsproblemen bei Internet Explorer erkannt und auf ihre Gültigkeit für WPF hin überprüft.
- **Toolbasierte Penetrationstests durch Dateitests mit zufälligen Daten:** Dateitests mit zufälligen Daten stellen die Ausnutzung des Eingabebereichs von Dateilesemodulen durch eine Vielzahl von Eingaben dar. Ein Beispiel, wo diese Technik in WPF verwendet wird, besteht in der Prüfung von Code zur Bildentschlüsselung auf Fehler.

Verwaltung von sicherheitsrelevantem Code

Für XAML Browseranwendungen (XBAPs) erstellt WPF eine Sicherheitssandbox mithilfe der .NET Framework-Unterstützung für das Kennzeichnen und Nachverfolgen von sicherheitskritischem Code, der Berechtigungen heraufstuft (weitere Informationen finden Sie unter **Sicherheitsrelevante Methode** in [WPF-Sicherheitsstrategie – Plattformsicherheit](#)). Angesichts der hohen Qualitätsanforderungen bei sicherheitskritischem Code wird derartiger Code durch eine zusätzliche Ebene der Quellcodeverwaltung und Sicherheitsüberwachung geschützt. Annähernd 5 % bis 10 % von WPF bestehen aus sicherheitskritischem Code, der von einem dedizierten Team überprüft wird. Der Quellcode und der Eincheckvorgang werden verwaltet, indem

sicherheitskritischer Code nachverfolgt und jede kritischen Entität (d. h. eine Methode, die kritischen Code enthält) ihrem abgezeichneten Zustand zugeordnet wird. Der abgezeichnete Zustand schließt die Namen eines oder mehrerer Prüfer ein. Bei jedem täglichen Build von WPF wird der kritische Code mit dem in vorhergehenden Builds verglichen, um nicht genehmigte Änderungen aufzuspüren. Wenn ein Programmierer kritischen Code ohne Genehmigung des Prüferteams ändert, wird der betreffende Code erkannt und sofort ersetzt. Dieses Vorgehen ermöglicht die Anwendung und Aufrechterhaltung eines sehr hohen Maßes an Genauigkeit bei WPF-Sandboxcode.

Warnung

XBAPs erfordern Legacybrowser, z. B. Internet Explorer und Firefox. Diese älteren Browserversionen werden unter Windows 10 und Windows 11 normalerweise nicht unterstützt. Moderne Browser unterstützen die für XBAP-Apps erforderliche Technologie aufgrund von Sicherheitsrisiken nicht mehr. Plug-Ins, die XBAPs aktivieren, werden nicht mehr unterstützt.

Weitere Informationen

- [Security](#)
- [WPF-Sicherheit mit teilweiser Vertrauenswürdigkeit](#)
- [WPF-Sicherheitsstrategie – Plattformsicherheit](#)
- [Sicherheit in .NET](#)

Zusammenarbeit auf GitHub

Die Quelle für diesen Inhalt finden Sie auf GitHub, wo Sie auch Issues und Pull Requests erstellen und überprüfen können. Weitere Informationen finden Sie in unserem [Leitfaden für Mitwirkende](#).



Feedback zu .NET Desktop feedback

.NET Desktop feedback ist ein Open Source-Projekt. Wählen Sie einen Link aus, um Feedback zu geben:

 [Problem in der Dokumentation öffnen](#)

 [Abgeben von Produktfeedback](#)

WPF-Beispiele

Artikel • 04.05.2023

Beispiele, die Windows Presentation Foundation (WPF) demonstrieren, finden Sie auf GitHub im [Repository „Microsoft/WPF-Samples“](#) [↗](#).

Klassenbibliothek (WPF)

Artikel • 04.05.2023

Die folgenden Links beziehen sich auf Namespaces, die Windows Presentation Foundation (WPF)-APIs enthalten.

In diesem Abschnitt

Verweis

- [Microsoft.Build.Tasks.Windows](#)
- [Microsoft.Win32](#) (freigegeben)
- [Microsoft.Windows.Themes](#)
- [System.Collections.ObjectModel](#) (freigegeben)
- [System.Collections.Specialized](#) (freigegeben)
- [System.ComponentModel](#) (freigegeben)
- [System.Diagnostics](#) (freigegeben)
- [System.IO](#) (freigegeben)
- [System.IO.Packaging](#)
- [System.Printing](#)
- [System.Printing.IndexedProperties](#)
- [System.Printing.Interop](#)
- [System.Security.Permissions](#) (freigegeben)
- [System.Security.RightsManagement](#)
- [System.Windows](#)
- [System.Windows.Annotations](#)
- [System.Windows.Annotations.Storage](#)
- [System.Windows.Automation](#)

- [System.Windows.Automation.Peers](#)
- [System.Windows.Automation.Provider](#)
- [System.Windows.Automation.Text](#)
- [System.Windows.Controls](#)
- [System.Windows.Controls.Primitives](#)
- [System.Windows.Converters](#)
- [System.Windows.Data](#)
- [System.Windows.Documents](#)
- [System.Windows.Documents.DocumentStructures](#)
- [System.Windows.Documents.Serialization](#)
- [System.Windows.Forms.Integration](#)
- [System.Windows.Ink](#)
- [System.Windows.Input](#)
- [System.Windows.Input.StylusPlugIns](#)
- [System.Windows.Interop](#)
- [System.Windows.Markup \(freigegeben\)](#)
- [System.Windows.Markup.Localizer](#)
- [System.Windows.Markup.Primitives](#)
- [System.Windows.Media](#)
- [System.Windows.Media.Animation](#)
- [System.Windows.Media.Converters](#)
- [System.Windows.Media.Effects](#)
- [System.Windows.Media.Imaging](#)
- [System.Windows.Media.Media3D](#)
- [System.Windows.Media.Media3D.Converters](#)

- [System.Windows.Media.TextFormatting](#)
- [System.Windows.Navigation](#)
- [System.Windows.Resources](#)
- [System.Windows.Shapes](#)
- [System.Windows.Threading](#)
- [System.Windows.Xps](#)
- [System.Windows.Xps.Packaging](#)
- [System.Windows.Xps.Serialization](#)
- [UIAutomationClientsideProviders](#)

XAML-Unterstützung in .NET 4

Die folgenden Namespaces enthalten Typen aus der System.Xaml-Assembly.

System.Xaml bietet allgemeine XAML-Sprachunterstützung für Frameworks wie WPF, die auf .NET Framework 4 basieren.

- [System.Windows.Markup](#) (freigegeben)
- [System.Xaml](#)
- [System.Xaml.Permissions](#)
- [System.Xaml.Schema](#)