Hinweise zum Buch

Bevor es losgeht, finden Sie hier noch ein paar Hinweise dazu, wie Sie am besten mit diesem Buch arbeiten. Auf den folgenden Seiten finden Sie kurze Inhaltsangaben zu allen Kapiteln und Informationen zu den verwendeten Beispielen: wo Sie diese finden und welche Systemvoraussetzungen notwendig sind, um sie nachzuvollziehen.

Für wen ist dieses Buch gedacht?

Um den meisten Nutzen aus diesem Buch zu ziehen, sollten Sie über eine gute Portion an C#-Know-how und über etwas Wissen rund um .NET verfügen. Fragen zu Events, Delegates, anonymen Methoden, Lambda Expressions, Auto-implemented Properties, `async` und `await` sollten Sie nicht gleich in Verlegenheit bringen. Sind Sie beim Thema .NET oder C# etwas unsicher, empfehle ich Ihnen, vorab ein C#-Buch zu lesen. Grundkenntnisse in XML sind für das vorliegende Buch von Vorteil, allerdings nicht zwingend erforderlich.

Das Buch richtet sich an .NET-Entwickler, die Desktop-Anwendungen für Windows mit der Windows Presentation Foundation umsetzen möchten. Es eignet sich sowohl für WPF-Einsteiger als auch für leicht fortgeschrittene WPF-Entwickler.

Seit Windows 10 lassen sich neben der klassischen Desktop-Anwendung auch die Universal Windows Apps entwickeln. WPF-Anwendungen funktionieren im Gegensatz zu den Universal Windows Apps auch unter älteren Windows-Versionen. Allerdings lassen sich WPF-Anwendungen nur auf dem Desktop von Windows ausführen. Die Universal Windows Apps laufen dagegen neben dem Desktop auf verschiedensten Umgebungen: Xbox, HoloLens oder sogar auf dem Raspberry Pi.

Doch wenn Universal Windows Apps auch auf dem Desktop laufen, warum sollten Sie dann die WPF einsetzen? Die WPF ist den Universal Windows Apps nach wie vor in einigen Teilen überlegen, da das Framework deutlich mehr Möglichkeiten bietet. Somit werden auch heute mächtige Geschäftsanwendungen, die als lokale Applikation unter Windows laufen sollen, mit der WPF umgesetzt. Beispielsweise ist Visual Studio 2019 eine WPF-Anwendung, und die typische datenzentrische Geschäftsanwendung wird es auch sein.

Wenn Sie also eine Anwendung für den Desktop bauen möchten, sind Sie hier genau richtig. Möchten Sie stattdessen lieber eine Universal Windows App entwickeln, in der Sie wie auch bei der WPF XAML und C# einsetzen können, sind Sie hier auch nicht falsch, da viele Konzepte gleich sind. Zudem finden Sie im letzten Kapitel dieses Buchs eine Einführung in die Entwicklung von Universal Windows Apps.

Im vorliegenden Buch erhalten Sie neben dem Wissen, das Sie zum Entwickeln einer WPF-Anwendung benötigen, reichlich Hintergrundinformationen über die Konzepte der WPF: Dazu gehören unter anderem die Extensible Application Markup Language (XAML), Dependency Properties, Routed Events, Commands, Ressourcen oder Logical und Visual Trees. Für einfache Applikationen ist die Kenntnis dieser Konzepte nicht immer erforderlich, für die Entwicklung Ihrer umfangreichen Wunschanwendung ist sie allerdings eine wichtige Voraussetzung. Erst mit diesen Konzepten im Hinterkopf werden Sie in der Lage sein, mit der WPF erfolgreich komplexe Anwendungen zu entwickeln.

Das vorliegende Buch richtet sich also an Entwickler und nicht an grafische Designer. Ein Designer kann mithilfe dieses Buchs zwar den XAML-Code nachvollziehen, den seine Tools wie Blend generieren, aber eben nur dann, wenn er auch die Grundlagen der .NET-Programmierung und C# beherrscht. Haben Sie Webseiten entwickelt und dazu Tools wie FrontPage oder Dreamweaver eingesetzt, werden Sie bestimmt die Erfahrung gemacht haben, dass für komplexere Fälle das Programm nicht den gewünschten Output liefert und Sie das HTML-Dokument manuell editieren müssen. Ebenso verhält es sich mit XAML. Obwohl es viele Tools gibt, die XAML generieren, werden Sie bestimmte Dinge in XAML weiterhin »von Hand« erstellen oder zumindest anpassen müssen, um zu Ihrem Ziel zu kommen.

Der Aufbau des Buches

Das Buch besteht aus 21 Kapiteln, die sich in vier Teile gruppieren lassen:

I. **WPF-Grundlagen und Konzepte**
In Teil I lernen Sie die Grundlagen der WPF kennen. Dazu gehören die wichtigsten Klassen, XAML, Controls, Layout und Konzepte der WPF, wie Dependency Properties, Routed Events oder Commands.
   - Kapitel 1: Einführung in die WPF
   - Kapitel 2: Das Programmiermodell
   - Kapitel 3: XAML
   - Kapitel 4: Der Logical und der Visual Tree
   - Kapitel 5: Controls
   - Kapitel 6: Layout
   - Kapitel 7: Dependency Properties
   - Kapitel 8: Routed Events
   - Kapitel 9: Commands

II. **Fortgeschrittene Techniken**
Fortgeschrittene Techniken werden in Kapitel 10 bis Kapitel 12 betrachtet. Dazu gehören neben Ressourcen die in der WPF existierenden Styles, Trigger und Templates. Mit Letzteren lässt sich das Aussehen von Controls neu definieren. In dieser Gruppe erfahren Sie auch, wie die WPF mit Daten umgeht. In diesem Zusammenhang gehe ich speziell auf das Data Binding ein.
   - Kapitel 10: Ressourcen
   - Kapitel 11: Styles, Trigger und Templates
   - Kapitel 12: Daten

III. **Reichhaltige Medien und eigene Controls**
In Teil III lernen Sie, wie Sie mit der WPF 2D- und 3D-Grafiken darstellen und auch dynamisch erzeugen. Sie lernen hier alles, was Sie über das in die WPF integrierte Animationssystem, über die Audio/Video-Unterstützung und über Texte und Dokumente wissen müssen. Die Dokumente in der WPF sind gleichzeitig auch der Schlüssel zum Drucken. Sie lernen aber nicht nur die Medien kennen, sondern erfahren auch, wie Sie eigene Controls entwickeln.
   - Kapitel 13: 2D-Grafik
   - Kapitel 14: 3D-Grafik
   - Kapitel 15: Animationen
   - Kapitel 16: Audio und Video
   - Kapitel 17: Eigene Controls
   - Kapitel 18: Text und Dokumente

IV. **Interoperabilität und Apps**
In Teil IV erfahren Sie, wie Sie in Ihrer WPF-Anwendung einen SaveFileDialog nutzen, Ihre Anwendung automatisieren und verteilen und wie Sie die Möglichkeiten der Windows Taskbar in Ihrer Applikation nutzen. Sie lernen auch, wie Sie in Ihrer WPF-Anwendung alte Technologien mittels Interoperabilität einbinden. Zudem erhalten Sie in Teil IV eine Einführung in die Entwicklung von Universal Windows Apps in Windows 10.
   - Kapitel 19: Standard-Dialoge, Windows Taskbar und mehr
   - Kapitel 20: Interoperabilität
   - Kapitel 21: Universal Windows Apps und WinUI

Der Inhalt der einzelnen Kapitel

- **Kapitel 1, »Einführung in die WPF«**
Wir beginnen mit einem Überblick über die WPF. Sie erfahren, wie sich die WPF ins .NET Framework eingliedert. Nach einem Blick auf die Geschichte der Windows-Programmierung lernen Sie die technische Architektur der WPF kennen und bekommen einen ersten Einblick in WPF-Konzepte wie XAML, Dependency Properties, Routed Events und Commands.

- **Kapitel 2, »Das Programmiermodell«**
  Die WPF besitzt eine tief verschachtelte Klassenhierarchie. Hier lernen Sie die zentralen Klassen kennen. Sie erhalten in diesem Kapitel eine Übersicht über die Projektvorlagen in Visual Studio 2019, bevor wir die ersten WPF-Anwendungen mit speziellem Fokus auf den Klassen `Application`, `Dispatcher` und `Window` entwickeln.
- **Kapitel 3, »XAML«**
  Kernpunkt dieses Kapitels ist die in der WPF zum Beschreiben von Benutzeroberflächen eingesetzte XML-basierte Beschreibungssprache *Extensible Application Markup Language* (XAML). Hier lernen Sie die Syntax von XAML sowie zahlreiche Tipps und Tricks kennen.
- **Kapitel 4, »Der Logical und der Visual Tree«**
  Wenn Sie eine Benutzeroberfläche in der WPF entwickeln, bauen Sie im Grunde eine Hierarchie von Objekten auf. Ein Window enthält einen Button, ein Button enthält einen String usw. Die WPF kennt zur Laufzeit zwei Hierarchien, die Voraussetzung für viele Funktionen der WPF sind, wie Routed Events und Ressourcen. Dieses Kapitel liefert reichlich Informationen über die Funktionsweise der beiden Hierarchien und zeigt mit einer Subklasse von `FrameworkElement`, wie die WPF die Hierarchien aufbaut.
- **Kapitel 5, »Controls«**
  Wie für ein UI-Framework üblich, enthält auch die WPF eine Vielzahl von bereits vordefinierten Controls. Dieses Kapitel zeigt Ihnen die wichtigsten dieser Controls, wie `TextBox`, `Menu`, `Button`, `TreeView`, `ComboBox` und `ListView`.
- **Kapitel 6, »Layout«**
  Das Anordnen und Positionieren von Elementen auf der Benutzeroberfläche unterliegt bei der WPF dem sogenannten *Layoutprozess*. Dieses Kapitel verrät, was sich hinter dem Layoutprozess verbirgt, und geht speziell auf die Größe von Elementen, den Rand, die Ausrichtung, Transformationen und Layout-Panels ein. Mit dem in diesem Kapitel vermittelten Wissen sind Sie in der Lage, ein pinnbares, animiertes Fenster ähnlich dem Projektmappen-Explorer in Visual Studio zu implementieren.
- **Kapitel 7, »Dependency Properties«**
  Dependency Properties erweitern die klassischen .NET-Properties um WPF-spezifische Logik. Sie sind die Grundlage für Animationen, Styles oder Data Bindings. Was Dependency Properties genau sind, wozu sie benötigt werden und wie Sie Dependency Properties nutzen und implementieren, ist Thema dieses Kapitels.
- **Kapitel 8, »Routed Events«**
  Die Events der WPF treten meist als sogenannte Routed Events auf. In diesem Kapitel wird gezeigt, wie Routed Events genau funktionieren und wie Sie eigene Routed Events implementieren. Darüber hinaus gehe ich auf Maus-, Tastatur-, Stift- und Touch-Events ein.
- **Kapitel 9, »Commands«**
  Commands sind insbesondere dann sinnvoll, wenn Sie mit mehreren Elementen (`Button`, `MenuItem` etc.) einen Befehl auslösen möchten. Neben dem Implementieren von eigenen Commands lernen Sie in diesem Kapitel die in der WPF bereits vorhandenen Commands

kennen. Außerdem betrachten wir hier das Model-View-ViewModel-Pattern (MVVM), das auf der Logik von Commands basiert.

- **Kapitel 10, »Ressourcen«**
  Dieses Kapitel zeigt Ihnen die Funktionsweise der WPF-spezifischen logischen Ressourcen. In ihm gehe ich auch auf die bereits aus älteren .NET-Versionen bekannten binären Ressourcen ein. Zudem lernen Sie hier, wie Sie auf einfache Weise einen Splashscreen erstellen.
- **Kapitel 11, »Styles, Trigger und Templates«**
  Styles werden in der WPF eingesetzt, um Werte für mehrere Properties zu definieren. Diese »Wertesammlung« lässt sich dann auf mehreren Elementen setzen. Templates dienen dazu, das Aussehen für ein Control oder für Daten zu definieren. In einem Style wie auch in einem Template lassen sich Trigger erstellen, die sich beispielsweise für Mouse-Over-Effekte verwenden lassen. Dieses Kapitel gibt Ihnen einen gründlichen Einblick in die Möglichkeiten mit Styles, Triggern und Templates. Dabei gehe ich auch auf den Visual-StateManager ein, der in .NET 4.0 eingeführt wurde.
- **Kapitel 12, »Daten«**
  Alles Wissenswerte zum Data Binding lesen Sie in diesem Kapitel. Darüber hinaus geht dieses Kapitel auf CollectionViews ein und zeigt, wie Sie bei der WPF durch Ihre Daten navigieren. Sie lernen, Daten zu gruppieren, zu sortieren, zu filtern und zu validieren. Ein Überblick über das `DataGrid` bringt Ihnen die zentralen Funktionen dieses Controls zum Darstellen von Listen näher.
- **Kapitel 13, »2D-Grafik«**
  In diesem Kapitel erfahren Sie alles über das »Zeichnen« mit der WPF. Sie erhalten Informationen über `Brushes`, `Shapes` und `Drawings`. Zudem zeige ich hier, wie Sie Elemente mit Effekten ausstatten – die sogenannten Pixelshader –, damit diese beispielsweise einen Schatten werfen.
- **Kapitel 14, »3D-Grafik«**
  Dieses Kapitel führt Sie in die 3D-Programmierung mit der WPF ein und bringt Ihnen die dazu notwendigen Grundlagen näher, wie das 3D-Koordinatensystem, das `Viewport3D`-Element, Kameras und 3D-Modelle. Sie lernen auch, wie Sie Ihre 3D-Inhalte interaktiv gestalten.
- **Kapitel 15, »Animationen«**
  Die WPF besitzt eine integrierte Unterstützung für Animationen. In diesem Kapitel erfahren Sie alles Notwendige über Timelines, Storyboards, einfache Animationen, Keyframe- und Pfad-Animationen und über die Animation Easing Functions.
- **Kapitel 16, »Audio und Video«**
  Integrierte Audio- und Video-Unterstützung ist eine der Stärken der WPF. Dieses Kapitel zeigt, wie Sie unter anderem Videos in Ihre Anwendung einbinden und sie zudem noch steuern.

- **Kapitel 17, »Eigene Controls«**
  Visual Studio bietet Ihnen zum Erstellen von Controls zwei Projektvorlagen: eine für User Controls und eine für Custom Controls. Wie Sie ein User Control und ein Custom Control entwickeln, erfahren Sie hier.
- **Kapitel 18, »Text und Dokumente«**
  Reichhaltige Funktionalität bietet die WPF im Zusammenhang mit Text und Dokumenten. Was Flow-, Fix- und XPS-Dokumente sind und wie Sie diese Dokumente in Ihrer Anwendung erstellen oder darstellen, ist der Inhalt dieses Kapitels. Dokumente sind auch der Schlüssel zum Drucken. Sie erfahren hier also auch, wie Sie aus Ihrer WPF-Anwendung etwas ausdrucken können.
- **Kapitel 19, »Standard-Dialoge, Windows Taskbar und mehr«**
  Sie lernen hier die Standard-Dialoge zum Speichern und Öffnen von Dateien kennen. Ebenso erfahren Sie, wie Sie Ihre Windows-Anwendung in die Taskbar von Windows 10 integrieren. In diesem Kapitel erkläre ich auch, welche Möglichkeiten Sie haben, um Ihre WPF-Anwendung zu verteilen.
- **Kapitel 20, »Interoperabilität«**
  In diesem Kapitel erfahren Sie, wie Sie die WPF mit Windows Forms, ActiveX, Win32 oder Direct3D kombinieren. Sie erhalten einen Überblick über mögliche Migrationsstrategien Ihrer Altanwendungen und erfahren anhand zahlreicher Beispiele, wie Sie verschiedene Interoperabilitätsszenarien implementieren.
- **Kapitel 21, »Universal Windows Apps und WinUI«**
  Unter Windows 10 lassen sich die sogenannten Universal Windows Apps entwickeln. Das sind native Apps, die C# und XAML und damit viele Konzepte aus der WPF nutzen. In diesem Kapitel erhalten Sie eine Einführung in die Entwicklung von Universal Windows Apps und in das WinUI Framework.

Wie Sie das Buch lesen

Dieses Buch ist so aufgebaut, dass Sie bei Kapitel 1 beginnen und sich von dort Schritt für Schritt zu Kapitel 21 vorarbeiten können. Am Ende jedes Kapitels folgt eine Zusammenfassung der wichtigsten Punkte. An diesem Überblick können Sie kontrollieren, ob Sie die wichtigsten Inhalte eines Kapitels aufgenommen haben.

Falls Sie nicht zu den Lesern gehören, die Bücher von vorn nach hinten durcharbeiten, können Sie sich natürlich auch einzelne Kapitel herauspicken und diese als Nachschlagelektüre verwenden. Beim Schreiben habe ich darauf geachtet, die einzelnen Kapitel möglichst unabhängig voneinander zu gestalten. Allerdings sollten einzelne Kapitel auch nicht immer alles wiederholen müssen, was in vorherigen Kapiteln bereits erläutert wurde. Somit benötigen Sie für spätere Kapitel dieses Buches meist ein Basiswissen, das Sie entweder bereits haben oder eben erlangen, indem Sie die vorherigen Kapitel durcharbeiten.

Hinweis

Das Buch verwendet aus didaktischen Gründen an manchen Stellen bewusst Wiederholungen. So erhalten Sie beispielsweise im ersten Kapitel einen Überblick über die WPF-Konzepte. In späteren Kapiteln werden diese Konzepte wiederholt und vertieft.

Systemvoraussetzungen

Alle Beispiele in diesem Buch wurden mit der finalen Version von *Visual Studio 2019* unter Windows 10 entwickelt. Dabei habe ich die frei erhältliche Edition namens *Visual Studio Community* genutzt, die Sie unter *www.visualstudio.com/downloads* herunterladen können.

Die Beispiele wurden mit .NET Core 3.0 entwickelt. Damit Sie also sofort loslegen und einige Beispiele ausprobieren können, sollten Sie auf Ihrem Rechner Folgendes installieren:

- als Betriebssystem mindestens **Windows 7** oder am besten **Windows 10**
- als Entwicklungsumgebung eine Edition von **Visual Studio 2019**
- **.NET Framework 4.8** oder **.NET Core 3.0**

Um die Beispiele aus Kapitel 21, »Universal Windows Apps und WinUI«, ausprobieren zu können, benötigen Sie zwingend Windows 10. Für alle anderen Kapitel reicht eine Windows-Version ab Windows 7 aus.

Codebeispiele

Die Codebeispiele zu diesem Buch können Sie in Form der *Beispiele.zip*-Datei unter folgendem Link herunterladen:

www.thomasclaudiushuber.com/wpfcode

Beispiele zu einem Kapitel

Die *Beispiele.zip*-Datei enthält eine nach Kapiteln aufgebaute Ordnerstruktur. Das bedeutet, Sie finden für jedes Kapitel einen Ordner: *K01*, *K02*, *K03* usw. Die Beispiele für Kapitel 5, »Controls«, liegen im Ordner *K05*. Unter den Codeausschnitten in diesem Buch finden Sie eine Unterschrift mit dem Pfad zu der Datei, die den dargestellten Code enthält. Das Folgende ist ein Codeausschnitt aus Kapitel 5, »Controls«; beachten Sie die Unterschrift:

```
<Label Content="_Benutzer"
       Target="{Binding ElementName=benutzer}"/>
<TextBox x:Name="benutzer" MinWidth="120" Text="Hallo"/>
```

Listing 1 Beispiele\K05\05 LabelTarget.xaml

Den Code aus diesem Listing finden Sie, wie in der Unterschrift angegeben, in der Datei *05 LabelTarget.xaml* im Ordner *KO5*.

Quellcode zu Abbildungen

In diesem Buch befinden sich auch Abbildungen, zu denen der Code nicht explizit abgebildet ist, da es beispielsweise an der entsprechenden Stelle für den dargestellten Sachverhalt nicht erforderlich ist. Dennoch finden Sie in der *Beispiele.zip*-Datei im Ordner eines Kapitels oft auch jenen Code, der diesen Abbildungen zugrunde liegt. Im Ordner *KO5* gibt es unter anderem eine Datei Abbildung 5.14 – *ScrollViewer.xaml*, die den für Abbildung 5.14 verwendeten Code enthält, der im Buch nicht explizit dargestellt ist.

Kapitelübergreifende Beispiele

Neben den Kapitelordnern wie *KO4* und *KO5* finden Sie im *Beispiele*-Ordner auch kapitelübergreifende Beispiele. Kapitelübergreifend ist unter anderem der Quellcode der in diesem Buch oft verwendeten Anwendung *FriendStorage*. Die kapitelübergreifenden Beispiele sind auf gleicher Ebene wie die Kapitelordner zu finden. Ein Ausschnitt des *Beispiele*-Ordners enthält somit die folgenden Ordner:

- *FriendStorage*
- *KO1*
- *KO2*
- *KO3*
- ...

Darstellungskonventionen

Ihre zukünftigen WPF-Anwendungen sollten ein durchgängiges Designkonzept verwenden, und so macht es auch dieses Buch. Beachten Sie die folgenden Hinweise zur Darstellung von Text, Code und Bemerkungen.

Textdarstellung

Pfade zu Dateien werden im Fließtext als *C:\In\Der\PfadFormatierung* geschrieben. Tritt ein wichtiger Begriff zum ersten Mal auf, wird er in der *Begriff-Formatierung* dargestellt. Dialognamen, Menüfunktionen und Ähnliches werden in KAPITÄLCHEN notiert.

Codedarstellung

Codebeispiele sind immer in der `Codeformatierung` dargestellt. In der Codeformatierung ist **`besonders wichtiger Code fett`** abgebildet, was natürlich nicht heißen soll, dass der restliche

Code unwichtig ist. Wie im vorherigen Abschnitt bereits erwähnt, werden die meisten Codebeispiele in diesem Buch mit einer Unterschrift versehen. Die Unterschrift enthält entweder einen Pfad zur Quelldatei oder eine kurze Beschreibung des dargestellten Codes. Sehr kurze, selbsterklärende Codebeispiele besitzen keine Unterschrift.

Einige Listings in diesem Buch besitzen Zeilennummern. Diese gehören nicht zum eigentlichen Quellcode, sondern dienen einfach als zusätzliches Beschreibungsmittel. In Kapitel 4, »Der Logical und der Visual Tree«, finden Sie ein solches Listing mit Zeilenangabe. Im Buchtext wird dann auf eine Listingzeile beispielsweise mit »Beachten Sie in Listing 2 in Zeile 8 ...« hingewiesen.

```
1  <TextBlock Height="50"
2             Padding="5,16,0,0"
3             Background="Black"
4             Foreground="White"
5             FontSize="17">
6    <Bold Foreground="LightGray">Friend</Bold><Bold
7          FontSize="20" Foreground="Red">S</Bold><Italic>
8          torage</Italic> - Info
9  </TextBlock>
```

Listing 2 Beispiele\FriendStorage\Dialogs\InfoDialog.xaml

An vielen Stellen dieses Buches ist eine vollständige Darstellung einer Quellcode-Datei nicht notwendig und würde den Sachverhalt nur komplizierter darstellen, als er tatsächlich ist. Entweder wird aus einer solchen Quellcode-Datei in einem Listing nur ein zusammenhängender Ausschnitt dargestellt oder es werden einzelne Codeabschnitte weggelassen. Auf nicht gezeigte Codeabschnitte wird im Listing mit drei Punkten (...) hingewiesen, sofern sich diese Codeabschnitte nicht am Anfang oder Ende des dargestellten Codes befinden.

Bemerkungen

Um den Fließtext etwas kompakter zu gestalten und besondere Informationen zusätzlich hervorzuheben, finden Sie einige Bemerkungen in einem Kasten. Dabei werden drei Arten von Bemerkungen verwendet: In einem Kasten befindet sich entweder ein Tipp, eine Warnung oder ein Hinweis.

Tipp

Ein Tipp gibt Ihnen etwas Nützliches mit auf den Weg. Der Tipp ist in manchen Situationen sehr hilfreich.

Achtung

Dieser Kasten weist Sie explizit auf ein eventuell auftretendes Problem, mögliche Stolperfallen oder etwas extrem Wichtiges hin.

Hinweis

Ein Hinweis-Kasten hebt wichtige Details hervor oder erläutert einen Sachverhalt noch etwas tiefgründiger.

Etwas zu Anglizismen

»Kommen Sie zu uns ins Office in unserer neuen Location in...«, warum nicht einfach *Büro* und *Standort* statt *Office* und *Location*? Ich bin wirklich kein Fan von Anglizismen. Allerdings ist es für einige Begriffe in diesem Fachbuch meines Erachtens wenig sinnvoll, sie ins Deutsche zu übertragen. Dieses Buch verwendet daher an vielen Stellen durchgängig die englischen Originalwörter und -begriffe. Ich habe mich dafür entschieden, da einerseits eine deutsche Variante eines Begriffs oft zu Missverständnissen führt und es andererseits in der WPF Begriffe wie beispielsweise *Dependency Properties* gibt, für die in der Entwicklerszene einfach kein konsistentes deutsches Pendant existiert. Deutschsprachige WPF-Entwickler sagen daher ebenfalls »Dependency Properties« und nicht »Abhängigkeitseigenschaften« (auch wenn die Dependency Properties in der deutschen MSDN-Dokumentation als Abhängigkeitseigenschaften bezeichnet werden).

Aufgrund dieser Tatsache finden Sie in diesem Buch die original englischen Begriffe wie *Dependency Properties*, *Routed Events*, *Commands*, *Markup Extensions* etc. Diese Begriffe gehen meist mit den Klassennamen einher, die für die entsprechende Funktionalität verwendet werden. Für Dependency Properties haben Sie die Klassen `DependencyObject` und `DependencyProperty`, für Routed Events die Klasse `RoutedEvent` usw.

Auch Steuerelemente werden durchgängig als »Controls« bezeichnet, die .NET-Eigenschaften eines Objekts als »Properties« oder die Behandlungsmethoden für ein Event als »Event Handler«.