Der Inhalt (im Überblick)

Der Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und C#. *Sie* versuchen, etwas zu *lernen*, und Ihr *Hirn* tut sein Bestes, damit das Gelernte nicht *hängen bleibt*. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z.B. für das Wissen darüber, welche Tiere einem gefährlich werden könnten oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, *wie* schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, etwas über C# zu wissen?

Bibliografische Informationen
http://d-nb.info/1045853968

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

1 Erste Schritte mit C# Apps im Handumdrehn

Sie wollen tolle Apps in Windeseile aufbauen?

Mit C# steht Ihnen eine **ausgezeichnete Programmiersprache** und ein **wertvolles Werkzeug** zur Verfügung. Dank der **Visual Studio IDE** werden Sie nie wieder Stunden damit verbringen müssen, obskuren Code zu schreiben, um einen simplen Button funktionsfähig zu machen. Noch besser ist, dass Sie **richtig coole Programme aufbauen können,** ohne dass Sie sich einprägen müssen, welche Teile Ihres Codes den *Namen* eines Buttons repräsentieren und welche für sein *Label* stehen. Klingt das interessant? Blättern Sie um, und legen Sie mit dem Programmieren los.

Es ist alles bloß Code

2 Unter der Motorhaube

Sie sind Programmierer, nicht bloß IDE-Benutzer.

Mit der IDE können Sie eine Menge Dinge erledigen, aber dennoch kann sie Sie nur ein bestimmtes Stück Ihres Wegs begleiten. Sicher gibt es eine Menge **sich wiederholender Aufgaben**, die Sie erledigen müssen, wenn Sie eine Anwendung aufbauen. Diese Aufgaben kann Ihnen die IDE wunderbar abnehmen. Aber die Arbeit mit der IDE ist *erst der Anfang*. Sie können Ihre Programme dazu bringen, noch viel mehr zu machen – und das geht nur, indem Sie **C#-Code schreiben**. Haben Sie das Programmieren einmal im Griff, gibt es *nichts mehr*, was Ihre Programme nicht tun könnten.

3 Objekte: Orientieren Sie sich!

Aufgabenspezifischer Code

Jedes Programm, das Sie schreiben, löst ein Problem.

Bevor Sie ein Programm erstellen, überlegen Sie sich am besten, welches *Problem* Ihr Programm lösen soll. Deswegen sind ***Objekte*** so ungemein hilfreich. Mit diesen können Sie Ihren Code auf Basis des Problems strukturieren, das er lösen soll. Und dann können Sie Ihre Zeit damit verbringen, *über das Problem nachzudenken*, an dem Sie arbeiten müssen, anstatt gleich beim Schreiben des Codes im Sumpf der Implementierungsanforderungen zu versinken. Wenn Sie Objekte richtig einsetzen, erhalten Sie Code, der *intuitiv* zu schreiben sowie leicht zu lesen und zu ändern ist.

4 Typen und Referenzen

Es ist 10:00. Wissen Sie, wo Ihre Daten sind?

Datentyp, Datenbank, Lieutenant Commander Data ... all das ist wichtiges Zeug. Ohne Daten sind Ihre Programme nutzlos. Sie brauchen **Informationen** von Ihren Anwendern und nutzen diese, um andere Informationen nachzuschlagen oder neue Informationen zu produzieren, die Sie dann wieder an Ihre User zurückliefern. Eigentlich schließt fast alles, was Sie beim Programmieren tun, auf die eine oder andere Weise **Arbeit mit Daten** ein. In diesem Kapitel werden Sie alles über die **Datentypen** von C# lernen, erfahren, wie Sie in Ihrem Programm mit Daten arbeiten, und Sie werden sogar auf einige schmutzige Geheimnisse bei **Objekten** stoßen (*psst ... auch Objekte sind Daten*).

5 Kapselung

Alles zu zeigen, ist nicht immer richtig

Denken Sie auch, dass man manches nicht gleich allen offenbaren muss?

Manchmal geht es Ihren Objekten ebenfalls so. Genau so, wie Sie nicht jeden Ihr Tagebuch oder Ihre Kontoauszüge lesen lassen, lassen gute Objekte ***andere*** Objekte nicht in ihre Felder blicken. In diesem Kapitel werden Sie die Macht der **Kapselung** kennenlernen. Sie **werden die Daten Ihrer Objekte privat machen** und Methoden hinzufügen, **die schützen, wie auf diese Daten zugegriffen wird.**

6 Vererbung

Der Stammbaum Ihres Objekts

Es gibt Momente, da *WILL man wie die eigenen Eltern sein.*

Ist Ihnen schon einmal ein Objekt über den Weg gelaufen, das ***fast*** genau das macht, was ***Ihr*** Objekt machen soll? Und haben Sie sich gewünscht, Sie ***bräuchten einfach nur ein paar Dinge zu ändern*** und das Objekt wäre perfekt? Genau das ist der Grund dafür, dass **Vererbung** eins der mächtigsten und wichtigsten Konzepte in C# ist. Bevor Sie mit diesem Kapitel durch sind, werden Sie gelernt haben, wie Sie von einer Klasse **ableiten**, um ihr Verhalten zu bekommen, dabei aber die **Flexibilität** bewahren, Änderungen an diesem Verhalten vorzunehmen. Sie **vermeiden doppelten Code**, **modellieren die wahre Welt** präziser nach und erhalten Code, der **leichter zu warten ist.**

7 Schnittstellen und abstrakte Klassen

Klassen, die Versprechen halten

Taten sagen mehr als Worte.

Gelegentlich müssen Sie Ihre Objekte auf Basis dessen gruppieren, **was sie tun können, und** nicht auf Basis der Klassen, von denen sie erben. Dann kommen **Schnittstellen** ins Spiel – diese ermöglichen Ihnen, mit jeder Klasse zu arbeiten, die bestimmte Aufgaben erledigen kann. Aber mit **großer Macht geht eine große Verantwortung einher**: Jede Klasse, die eine Schnittstelle implementiert, muss versprechen, **all ihre Verpflichtungen zu erfüllen** ... oder der Compiler tritt Ihnen vors Schienbein, Verstanden()?

8 Enums und Auflistungen

Daten in Massen speichern

Eigentlich tritt alles immer in Massen auf.

Im wahren Leben wird man mit Daten nie in kleinen Fragmenten konfrontiert. Nein, sie begegnen Ihnen immer in **Massen, Bergen und Haufen**. Und Sie brauchen ziemlich leistungsfähige Werkzeuge, um sie zu organisieren. Genau das sind **Auflistungen**. Mit diesen können Sie alle Daten **speichern, sortieren und verwalten**, die Ihr Programm durchforsten muss. Sie können sich darauf konzentrieren, wie Ihr Programm mit den Daten arbeitet, und es den Auflistungen überlassen, sich für Sie um die Daten selbst zu kümmern.

9 Dateien lesen und schreiben

Speichere das Array, rette die Welt

Gelegentlich zahlt es sich aus, wenn man Dinge festhält. Bisher hatten alle Ihre Programme nur ein Kurzzeitgedächtnis. Sie starten, laufen eine Weile und enden dann. Manchmal reicht das nicht aus, insbesondere wenn Sie mit wichtigen Daten arbeiten. Sie müssen **Ihre Arbeit speichern können**. In diesem Kapitel werden wir uns ansehen, wie man **Daten in einer Datei speichert** und diese **Informationen dann wieder aus einer Datei einliest**. Sie werden etwas über die **.NET-Stream-Klassen** lernen und sich mit den Mysterien **Hexadezimal** und **Binär** auseinandersetzen.

10 Windows Store-Apps mit XAML

Ihre Apps auf die nächste Stufe bringen

Sie sind für ein neues Kapitel der App-Entwicklung bereit.

Mit WinForms und dem Aufbau von Windows-Desktop-Programmen kann man wichtige C#-Konzepte leicht erlernen, aber *Ihre Programme könnten noch so viel mehr leisten.* In diesem Kapitel werden Sie **XAML** nutzen, um Windows Store-Apps zu gestalten. Sie werden lernen, wie man **Seiten so aufbaut, dass sie auf alle Geräte passen**, wie man mit **Datenbindungen** Daten in Seiten **einbindet** und wie man sich mit Visual Studio einen Weg zu den innersten Geheimnissen von XAML-Seiten bahnt, indem man die Objekte untersucht, die von Ihrem XAML-Code erstellt werden.

XAML, Datei-I/O und Datenkontraktserialisierung

11 Entschuldigen Sie die Unterbrechung

Niemand mag, warten müssen, insbesondere Nutzer nicht.

Computer sind sehr geschickt, wenn es darum geht, mehrere Dinge auf einmal zu tun – es gibt also keinen Grund dafür, dass Ihre Apps das nicht auch können sollten. In diesem Kapitel werden Sie lernen, wie Sie Ihre Apps reaktionsfähig halten, indem Sie **asynchrone Methoden erstellen**. Sie werden auch erfahren, wie Sie die eingebauten **Dateiwähler- und Benachrichtigungsdialoge** sowie **asynchrone Dateieingabe und -ausgabe** nutzen, damit Ihre Apps nicht einfrieren. Kombinieren Sie das mit der **Datenkontraktserialisierung,** und Sie haben das Fundament einer absolut modernen App.

12 Exception-Handling
Fehler-Prävention

Programmierer sind keine Feuerlöscher. Sie haben sich auf die Hinterbeine gesetzt, einen Haufen trockener Handbücher und ein paar ansprechende Von Kopf bis Fuß-Bücher durchgearbeitet und haben den Gipfel Ihres Berufsstands erreicht: **Meisterprogrammierer**. Dennoch erhalten Sie immer mitten in der Nacht panische Anrufe, weil **Ihr Programm abstürzt** oder **sich nicht so verhält, wie es sich verhalten soll**. Nichts kann einem die Programmierstimmung so verhageln wie die Forderung, einen seltsamen Fehler zu beheben ... aber mit **Exception-Handling** kann sich Ihr Code **um die aufkommenden Probleme kümmern**. Und Sie können auf diese Probleme sogar reagieren und dafür sorgen, **dass das Programm weiterläuft**.

13

14 Datenabfrage und App-Bau mit LINQ

Bekommen Sie Ihre Daten in den Griff

Die Welt ist datengesteuert ... Sie sollten damit zu leben wissen.

Vorbei sind die Zeiten, in denen Sie Tage, sogar Wochen programmieren konnten, ohne sich mit **Massen von Daten** befassen zu müssen. Heute ***dreht sich alles um Daten***. Und mit **LINQ** lässt sich all das bewältigen. Mit LINQ können Sie **Ihre Daten** nicht bloß auf leichte, intuitive Weise **abfragen**, sondern auch **gruppieren** und **Daten von unterschiedlichen Quellen zusammenführen**. Und wenn Sie Ihre Daten zu Happen gebündelt haben, mit denen man arbeiten kann, bieten Ihnen Windows Store-Apps **Steuerelemente**, über die Benutzer diese Daten erforschen, durchlaufen oder auch detailliert ansehen können.

15 Events und Delegates

Was Ihr Code macht, wenn Sie nicht gucken

Ihre Objekte beginnen, für sich selbst zu denken.

Sie können nicht immer kontrollieren, was Ihre Objekte machen. Manche Dinge passieren einfach. Und wenn sie passieren, sollten Ihre Objekte schlau genug sein, **auf alles eine Antwort zu wissen**, was eintreten könnte. Darum geht es bei Events (oder Ereignissen). Ein Objekt *veröffentlicht* ein Event, andere Objekte *abonnieren* es, und alle arbeiten zusammen, damit die Dinge im Fluss bleiben. Das geht so lange gut, bis Sie steuern können wollen, wer ein Event abonnieren darf. Dann werden sich **Callbacks** als praktisch erweisen.

16 App-Entwurf mit dem MVVM-Muster

Tolle Apps, außen wie innen

Beeindruckendes Aussehen allein reicht nicht.

Woran denken Sie, wenn Sie den Begriff Design hören? An ein Beispiel hervorragender Gebäudearchitektur? Eine aufregend gestaltete Seite? Ein Produkt, das gleichermaßen ästhetisch ansprechend wie gut gebaut ist? Genau diese Prinzipien gelten auch für Apps. In diesem Kapitel werden Sie das **Model-View-ViewModel-Muster** kennenlernen und erfahren, wie Sie mit seiner Hilfe gut gebaute, locker gebundene Apps aufbauen. Dabei werden Sie etwas über **Animationen und Control-Templates** für die visuelle Gestaltung Ihrer Apps erfahren, lernen, wie Sie sich Datenbindungen mit **Konvertierern** erleichtern und wie Sie all das zusammenbringen, um ein ***solides C#-Fundament*** für alle Apps zu gestalten, die Sie erstellen wollen.

Bonusprojekt

17 Erstellen Sie eine Windows Phone-App

Sie können bereits Windows Phone-Apps schreiben.

Klassen, Objekte, XAML, Kapselung, Vererbung, Polymorphie, LINQ, MVVM ... Sie haben alle Werkzeuge, die Sie zur Erstellung beeindruckender Windows Store- und Desktop-Apps benötigen. Aber wussten Sie auch, dass Sie die **gleichen Werkzeuge nutzen können, um Apps für *Windows Phones* zu erstellen**? Sie haben richtig gehört! In diesem Bonus-Projekt werden wir Sie durch den Aufbau eines Spieles für die Windows Phone-Plattform begleiten. Und sollten Sie kein Windows Phone-Gerät haben, können Sie den **Windows Phone-Emulator** nutzen, um es zu spielen. Legen wir los!

Anhang: Was übrig bleibt

Die Top 11 der Themen, die es nicht ins Buch geschafft haben

Der Spaß fängt gerade erst an!

Wir haben Ihnen viele wunderbare Werkzeuge gezeigt, mit denen Sie mit C# richtig **mächtige Software** aufbauen können. Aber es war unmöglich, **alle Werkzeuge, Technologien und Techniken** in dieses Buch einzuschließen – so viele Seiten hat es einfach nicht. Wir mussten einige *sehr harte Entscheidungen* in Bezug darauf treffen, was wir aufnehmen und was wir weglassen. Hier sind ein paar Themen, die es nicht geschafft haben. Aber auch wenn wir zu ihnen nicht gekommen sind, denken wir trotzdem, dass sie **wichtig und nützlich** sind, und möchten Ihnen einen kleinen Ausblick auf sie bieten.