Einleitung

Herzlich willkommen beim Einsteiger-Kurs »Visual C#-Programmierung«! Freuen Sie sich auf eine ebenso unterhaltsame wie fundierte Einführung in die Programmierung und auf:

- **interessante Programmierbeispiele** – wie z.B. eine eigene Version der Windows-Fotoanzeige, ein Rechentrainer, ein Einarmiger Bandit, ein Reaktionstester, ein Texteditor, ein Bildschirmschoner etc.
- **pures Windows** – von Anfang an werden Sie mit Windows-Anwendungen, also Programmen mit grafischer Benutzeroberfläche arbeiten
- **viele Übungen** – am Ende jedes Kapitels gibt es Übungen zum Wiederholen und Einüben des gelernten Stoffes sowie zum Sammeln eigener Erfahrungen. Die Lösungen zu diesen Übungen finden Sie am Ende des Buches.
- **Schritt-für-Schritt-Anleitungen und Fallstudien** – neben den Übungen am Kapitelende gibt es noch diverse in die Kapiteltexte eingestreute Schritt-für-Schritt-Anleitungen und Studien, die Sie theoretisch – oder besser noch ganz real – nachvollziehen können. Diese Übungen sollen Ihnen helfen, typische Arbeitsgänge zu verinnerlichen, Praxis zu schnuppern oder auch – wie im Falle der Designstudie »Reaktionstester« aus *Kapitel 9* – einmal näherungsweise nachzuempfinden, mit welchen Problemen und Entscheidungen man als Programmierer so zu kämpfen hat (also quasi einen Blick über die Schulter zu werfen).
- zahlreiche **Hinweise**, **Praxistipps** und **Hintergrundinformationen**
- **große Themenvielfalt** – Hauptaugenmerk dieses Buches ist natürlich das Erlernen der Programmiersprache C#, die Erstellung von Windows-Anwendungen und die Arbeit mit der Entwicklungsumgebung Visual Studio. Darüber hinaus können Sie sich mit diesem Buch aber noch in viele weitere Programmiergebiete einarbeiten, namentlich:
  - die Erstellung von Datenbankanwendungen (*Kapitel 14*),
  - das Schreiben von Bildschirmschonern (*Kapitel 15*)
  - das Schreiben von Konsolenanwendungen (*Anhang E*)
  - die Weitergabe von Programmen (*Anhang C*)
  - das Schreiben von Windows Store-Apps (als herunterladbarer PDF-Auszug aus dem C#-Entwicklerbuch)

1.1 Aufbau dieses Buches

Dieses Buch ist ein Lehr-, Lern- und Arbeitsbuch.

- Ein Lehrbuch, weil es die Elemente der C#-Syntax, die Konzepte der objektorientierten Programmierung und die wichtigsten Techniken zur Erstellung von Windows-Anwendungen beschreibt
- Ein Lernbuch für das Selbststudium, weil es den Leser an die Hand nimmt, weil es erklärt, Ideen und Konzepte plausibel macht, Hintergrundinformationen liefert und Schritt-für-Schritt-Anweisungen zum Nachvollziehen enthält
- Ein Arbeitsbuch, weil auch von Ihnen etwas verlangt wird. Folgen Sie den Erklärungen und Ausführungen, vollziehen Sie die Beispiele nach, versuchen Sie sich an den Übungen und entwickeln Sie Ideen für eigene Programme (und sei es nur zum Testen des soeben gelernten Stoffes).

Das Buch soll Sie mit Lust und ohne Verdruss in die C#-Programmierung einführen.

Für den Spaß am Programmieren sorgen hoffentlich Ihre Neugier, sich einstellende Erfolgserlebnisse und das eine oder andere Beispiel (wie der Kaninchenzähler, der Reaktionstester oder der eigene Bildschirmschoner). Verdruss stellt sich dagegen meist dann ein, wenn zu wenig erklärt wird – mit dem Effekt, dass der Leser das Gefühl hat, überhaupt nicht zu verstehen, was er da treibt – oder wenn der Leser den Erklärungen nicht mehr folgen kann und irgendwann frustriert aufgibt. Damit dies nicht passiert, legt dieses Buch großen Wert auf verständliche Erklärungen und nimmt sich, wo notwendig, die Zeit, Hintergrundinformationen zu vermitteln, Parallelen zu ziehen, Anwendungen aus der Praxis aufzuzeigen oder Negativbeispiele anzuführen.

Ein Spaziergang wird das Lernen mit diesem Buch deswegen aber nicht. Zum einem ist dieses Buch gespickt mit Informationen, Hinweisen und Syntaxbeispielen. Zum anderen ist es in C# schlichtweg unmöglich, auch nur das kleinste Programm zu schreiben, ohne zugleich auf zahlreiche, teils weit fortgeschrittene Programmierkonzepte vorzugreifen. So wird der Anfänger gleich zu Beginn mit einer Fülle von neuen Begriffen, Konzepten und Syntaxformen konfrontiert. Dabei nicht den Überblick zu verlieren und trotzdem halbwegs zu verstehen, was man tut, ist gar nicht so leicht.

Das vorliegende Buch verfolgt daher einen dreistufigen Ansatz:

- *Kapitel 2 und 3* bilden den sprichwörtlichen Sprung ins kalte Wasser. Viele zentrale Konzepte, sowohl der Sprache C# als auch der Entwicklungsumgebung Visual Studio, werden hier bereits eingeführt und erläutert – mit dem Ziel, dass Sie einfach schon einmal von den betreffenden Konzepten gehört und eine erste Vorstellung von ihnen gewonnen haben.

- In den nachfolgenden Kapiteln werden die verschiedenen Konzepte dann noch einmal aufgegriffen und vertieft. *Kapitel 4 bis 5* und *7 bis 9* widmen sich dabei vornehmlich der Sprache C#, *Kapitel 6* der Visual Studio-Umgebung und die *Kapitel 10 bis 13* dem Aufbau grafischer Benutzeroberflächen.
- Und damit Sie sich auf jedem Wissensstand schnell über einen bestimmten Fachbegriff oder eine bestimmte Syntaxform informieren können, finden Sie am Ende des Buches ein ausführliches Stichwortverzeichnis

Zuletzt ein Rat: Sollten Sie auf Schwierigkeiten stoßen – sei es, dass Sie etwas nicht ganz verstanden haben oder ein Programm nicht zum Laufen bekommen –, versuchen Sie, sich nicht zu sehr in das Problem zu verbohren. Legen Sie eine kleine Pause ein oder lesen Sie erst einmal ein wenig weiter – oftmals klärt sich das Problem danach von selbst.

Konventionen in diesem Buch 1.2

Dieses Buch verwendet einige wenige Konventionen. Wenn Sie mit ihnen vertraut sind, können Sie einfacher mit dem Buch arbeiten:

- **Fette Schrift** – Text, den Sie in Eingabefelder eintippen müssen, wird in fett gedruckten Buchstaben dargestellt. In größeren Listings wird die Fettschrift gelegentlich dazu verwendet, die wichtigeren Codepassagen hervorzuheben.
- *Kursive Schrift* – Die kursive Schrift wird für Dateinamen verwendet oder kennzeichnet Elemente der Benutzeroberfläche (also die Benennungen der Menüs, Registerkarten, Schaltflächen oder Eigenschaften).
- `Listingschrift` – Die Listingschrift wird für die abgedruckten C#-Codebeispiele verwendet, sowie für C#-Code, der im Fließtext auftaucht (also beispielsweise Namen von Variablen, Methoden oder Ähnliches).

Zusätzlich finden Sie im ganzen Buch verteilt Absätze, die mit den folgenden Icons versehen sind und die auf bestimmte, hilfreiche Elemente hinweisen:

Wichtig Absätze mit diesem Icon enthalten wichtige Informationen, auf die Sie unbedingt achten sollten.

Tipp In den Tipp-Absätzen finden Sie Informationen dazu, wie Sie Dinge besonders einfach und zeitsparend erledigen können.

Hintergrund Absätze mit diesem Icon liefern wichtige Hintergrundinformationen.

1.3 Software und Beispielsammlung

Als Entwicklungsumgebung wird in diesem Buch die Visual Studio 2013 Express Edition verwendet, die Sie – traditionell kostenlos – von der Webseite *http://www.visualstudio.com/de-de/downloads/download-visual-studio-vs* herunterladen können. Wählen Sie *Visual Studio 2013 Express für Windows Desktop* und klicken Sie auf *Jetzt Installieren*. Für nähere Erläuterungen zur Installation siehe *Anhang B*.

Die Beispielsammlung zu diesem Buch finden Sie zusammen mit einem PDF-Auszug zur Windows Store-App-Programmierung aus dem ebenfalls bei Microsoft Press erschienenen *C#-Entwicklerbuch hier*

www.microsoft-press.de/support/9783866452213

oder hier:

msp.oreilly.de/support/2341/858

Arbeiten mit der Beispielsammlung

Nachdem Sie die ZIP-Datei der Beispielsammlung heruntergeladen haben, sollten Sie die ZIP-Datei in ein Verzeichnis auf Ihrem Rechner extrahieren – beispielsweise unter *C:* oder unter Ihrem Benutzerverzeichnis. Als Windows 8- oder Windows 7-Anwender können Sie zum Entpacken den Extrahieren-Befehl des Windows Explorer (Aufruf über (⊞ + E)) verwenden. Achten Sie beim Extrahieren darauf, dass die Pfadangaben berücksichtigt werden. Sie finden die Projekte dann nach Kapiteln geordnet unter dem Verzeichnis *Beispiele*.

Beispielprojekte öffnen und ausführen

Zum Laden der Beispielprojekte gehen Sie folgendermaßen vor:

1. Starten Sie Visual Studio Express und laden Sie über *Datei/Projekt öffnen* die Projektmappendatei aus dem Projektverzeichnis des ausführenden Programms (z.B. *Beispiele\Kapo3\ErstesProjekt.sln*).

Das Projekt wird daraufhin geladen und die einzelnen Projektdateien werden rechts im Projektmappen-Explorer aufgelistet, von wo aus sie per Doppelklick zur Bearbeitung aufgerufen werden können (siehe auch Ausführungen in *Kapitel 2*).

2. Wenn Sie ein Projekt erstellen und das erstellte Programm ausführen möchten, rufen Sie den Menübefehl *Debuggen/Starten ohne Debugging* auf oder drücken Sie die Tastenkombination Strg F5.

Sollte es Schwierigkeiten mit den Projektdateien geben, dann erstellen Sie in Ihrer Visual Studio-Umgebung ein neues Projekt und kopieren Sie von den Beispielen lediglich die C#-Quelltextdateien (Endungen *.cs*).

Einige Beispielprogramme sind auf externe Ressourcen (Textdateien, Datenbanken etc.) angewiesen. In den Projektverzeichnissen finden Sie dann eine *Liesmich.txt*-Datei mit Hinweisen zu den Abhängigkeiten.

Liesmich.txt

Einstellungen aus Beispielprojekten ablesen

Die Beispiele im Buch sind üblicherweise recht ausführlich erklärt. Allerdings konzentrieren sich die Erklärungen meist auf das gerade behandelte Thema. Wenn Sie sich fragen, wie andere Teile der Beispielanwendung implementiert sind, laden Sie einfach das zugehörige Beispielprojekt. Mithilfe von Visual Studio können Sie praktisch alle Einstellungen abfragen und jeden Code einsehen.

Hinweis: Die folgenden Erläuterungen setzen voraus, dass Sie bereits *Kapitel 2 und 3* gelesen haben.

Welche Projektvorlage wurde verwendet?

1. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf den Projektknoten (dies ist der Knoten unter dem Projektmappen-Knoten, der bei Projektmappen mit nur einem Projekt immer fett hervorgehoben ist) und wählen Sie den Befehl *Eigenschaften*.
2. Wählen Sie links die Seite *Anwendung*. Danach sehen Sie rechts die Einstellung *Ausgabetyp*, die auf die verwendete Vorlage rückschließen lässt. Der Ausgabetyp *Windows-Anwendung* entspricht z.B. der Vorlage *Windows Forms-Anwendung*.

Wie wurde eine Komponente konfiguriert?

1. Laden Sie das Formular mit der Komponente in den Designer (Doppelklick im Projektmappen-Explorer auf den Knoten des Formulars).
2. Klicken Sie im Formular auf die Komponente, um sie auszuwählen, und lassen Sie, falls nicht schon geschehen, das Eigenschaftenfenster anzeigen (Befehl *Ansicht/Eigenschaftenfenster*).
3. Lassen Sie die Eigenschaften anzeigen (Klick auf die Schaltfläche *Eigenschaften* in der Symbolleiste des Eigenschaftenfensters).
4. Jetzt können Sie sich in Ruhe ansehen, wie die Eigenschaften konfiguriert wurden. Achten Sie dabei besonders auf die fett hervorgehobenen Werte – dies sind die Werte, die gegenüber der Standardeinstellung der Komponente verändert wurden.

Fett hervorgehobene Eigenschaftenwerte

Fett ist leider nicht gleichbedeutend mit »wurde vom Programmierer verändert«. Einige wenige Eigenschaften werden nämlich bereits vom Designer verändert, wenn Sie die Komponente im Formular ablegen (z.B. wird die *AutoSize*-Eigenschaft von Label-Komponenten auf **True** gesetzt). Wenn Sie oder ein anderer Programmierer eine solche Eigenschaft dann auf ihren Standardwert zurücksetzen (z.B. *AutoSize* auf **False**), verschwindet die Fettmarkierung.

Mit welchem Ereignisbehandlungscode wurde eine Komponente verbunden?

1. Laden Sie das Formular mit der Komponente in den Designer (Doppelklick im Projektmappen-Explorer auf den Knoten des Formulars).
2. Klicken Sie im Formular auf die Komponente, um sie auszuwählen und lassen Sie, falls nicht schon geschehen, das Eigenschaftenfenster anzeigen (Befehl *Ansicht/Eigenschaftenfenster*).
3. Lassen Sie die Ereignisse anzeigen (Klick auf die Schaltfläche *Ereignisse* in der Symbolleiste des Eigenschaftenfensters).
4. Für Ereignisse, die mit Code verbunden wurden, wird im Feld neben dem Ereignis die Behandlungsmethode angezeigt. Mit einem Doppelklick auf den Namen der Behandlungsmethode können Sie den Code der Methode in den Editor laden.

Wie sieht der Code des Formulars aus?

1. Laden Sie den Code des Formulars in den Editor (Klicken Sie im Projektmappen-Explorer auf den Knoten des Formulars und rufen Sie im Kontextmenü den Befehl *Code anzeigen* auf).
2. Im Code sehen Sie nicht nur alle Ereignisbehandlungsmethoden, sondern auch welche weiteren Felder und Methoden für das Formular definiert wurden.
3. Am Anfang der Datei sehen Sie die importierten Namespaces.
4. Am Anfang der Klassendefinition sehen Sie den Konstruktor des Formulars.

Wie ist das Layout eines Formulars aufgebaut?

1. Laden Sie das Formular mit der Komponente in den Designer (Doppelklick im Projektmappen-Explorer auf den Knoten des Formulars).
2. Rufen Sie den Menübefehl *Ansicht/Weitere Fenster/Dokumentgliederung* auf.
3. In dem erscheinenden Fenster sehen Sie die Hierarchie der eingebetteten Komponenten. Diese Anzeige ist besonders hilfreich, wenn Sie in das Formular eine Container-Komponente einbetten (wie z.B. Panel oder ToolStrip), in die Sie wieder andere Komponente einbetten, sodass eine mehrstufige Komponentenhierarchie entsteht (siehe *Kapitel 10.4*).
4. Am Anfang der Klassendefinition sehen Sie den Konstruktor des Formulars.