Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Richard Kaiser

C++ mit Microsoft Visual C++ 2008

Einführung in Standard-C++, C++/CLI
und die objektorientierte
Windows .NET-Programmierung

Springer

Prof. Richard Kaiser
Schwärzlocher Str. 53
72070 Tübingen
Deutschland
http://www.rkaiser.de/

Xpert.press ISSN 1439-5428
ISBN 978-3-540-23869-0 e-ISBN 978-3-540-68844-0
DOI 10.1007/978-3-540-68844-0
Springer Heidelberg Dordrecht London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandentwurf: KuenkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Für Daniel, Alex und Kathy

Geleitwort

Wenn man heute nach Literatur über Programmiersprachen sucht, so findet man viel Neues über Sprachen wie C# und Java, die von einer virtuellen Maschine ausgeführt werden, aber auch über die dynamischen Sprachen wie Python, Ruby und PHP. Es könnte der Eindruck entstehen, dass Softwareentwicklung im Wesentlichen mit diesen Sprachen stattfindet. Dies ist aber nicht der Fall. Ein nicht zu unterschätzender Teil der professionellen Softwareentwicklung wird auf der Basis der Programmiersprache C++ durchgeführt. In meiner beruflichen Praxis, in der ich Unternehmen betreue, die mit den Softwareentwicklungswerkzeugen von Microsoft arbeiten, begegne ich häufig der Programmiersprache C++.

C++ ist auch heute immer noch die erste Wahl, wenn es darum geht, hocheffiziente Software für den technisch-wissenschaftlichen Bereich oder hardwarenahe Aufgaben zu entwickeln. Man darf auch nicht vergessen, dass für C++ umfangreiche, qualitativ hochwertige und plattformunabhängige Klassenbibliotheken für viele Anwendungsdomänen existieren. Ein anderer Vorteil von C++ ist meiner Ansicht nach, dass die Sprache sowohl das prozedurale als auch das objektorientierte Entwicklungsparadigma unterstützt. Für Microsoft spielt C++ weiterhin eine wichtige Rolle bei der Softwareentwicklung im eigenen Haus, aber auch in der Weiterentwicklung der Werkzeuge für die Programmierung mit C++, um diese noch produktiver und sicherer zu machen. Microsoft arbeitet aktiv im ISO Komitee zur Standardisierung von C++ mit und hat mit dem Standard ECMA-372 (C++/CLI) C++ für den Einsatz auf der .NET Laufzeitumgebung erweitert. Außerdem ist die einfache Integration von nativen C++ Klassenbibliotheken mit Programmen, die für Microsoft .NET entwickelt werden bzw. wurden, eine nicht zu unterschätzende Eigenschaft der .NET Laufzeitumgebung, einmal unter dem Gesichtspunkt des Investitionsschutzes, als auch unter dem Aspekt der Laufzeiteffizienz.

Das vorliegende Buch von Richard Kaiser, der selbst aktiv im DIN an der Standardisierung von C++ mitarbeitet, gibt eine umfassende Einführung in alle Aspekte von Standard C++, sowie in die Spezialitäten von C++ auf der Microsoft Windows Plattform und C++/CLI für Microsoft .NET. Die Kapitel 1 und 2 bieten eine sehr gute Einführung in das Arbeiten mit Microsoft Visual Studio 2008. Sehr positiv ist noch, dass die prozeduralen und objektorientierten Eigenschaften von C++ explizit dargestellt werden, und dass in Kapitel 3.7 eine Einführung in die Techniken und Möglichkeiten der Programmverifikation im Kontext von C++

gegeben wird. Diese Techniken werden an Bedeutung gewinnen, um die Qualität und Sicherheit von Software besser zu gewährleisten.

Ich wünsche dem Buch viel Erfolg, denn es bietet eine fundierte Einführung in alle Aspekte der Softwareentwicklung mit C++ auf der Microsoft Windows Plattform, einschließlich dem .NET Framework, und es trägt dazu bei, den Einstieg in diese mächtige Programmiersprache zu erleichtern.

Klaus Rohe, Platform Strategy Manager, Developer Platform & Strategy Group, Microsoft Deutschland GmbH

Vorwort

Dieses Buch entstand ursprünglich aus dem Wunsch, in meinen Vorlesungen über C++ nicht nur Textfensterprogramme (Konsolenanwendungen), sondern von Anfang an Windows-Programme zu entwickeln. Dafür ist Visual Studio 2008 sehr gut geeignet. Es ist so einfach zu bedienen, dass man es auch in Anfängervorlesungen einsetzen kann, ohne dabei Gefahr zu laufen, dass die Studenten nur noch mit dem Entwicklungssystem kämpfen und gar nicht mehr zum Programmieren kommen.

Dieses Buch richtet sich aber nicht nur an Anfänger, sondern ebenso an professionelle Software-Entwickler. In den letzten 10 Jahren habe ich zahlreiche C++-Kurse für Entwickler aus der Industrie gehalten. Dabei wurde ich mit einer Fülle von Anregungen aus ihrer täglichen Arbeit konfrontiert, die dem Buch viele praxisorientierte Impulse gaben. Für diese Leser wird C++ und C++/CLI umfassend dargestellt.

Dieses Buch besteht aus drei Teilen:

- Teil 1 stellt die Entwicklungsumgebung Visual Studio 2008 vor und zeigt, wie man Windows-Programme mit den wichtigsten Steuerelementen entwickelt.
- Teil 2 stellt den gesamten Sprachumfang des C++-Standards umfassend vor. Dazu gehören nicht nur die Sprachelemente von C sowie Klassen, Templates und Exception-Handling, sondern auch die C++-Standardbibliothek.
- Teil 3 behandelt den gesamten C++/CLI-Standard und gibt einen Einblick in die .NET Klassenbibliothek. Die C++/CLI-Erweiterungen bieten die Möglichkeit, mit C++ Windows .NET-Programme zu schreiben und die .NET Klassenbibliothek zu nutzen. Insbesondere kann man bestehende C++-Quelltexte in Windows .NET-Anwendungen aufzunehmen und so vorhandenen Code nutzen.

Die Programmiersprache C++ wurde als Obermenge der Programmiersprache C entworfen. Dieser Entscheidung verdankt C++ sicher seine weite Verbreitung. Sie hat aber auch dazu geführt, dass oft weiterhin wie in C programmiert wird und lediglich ein C++-Compiler anstelle eines C-Compilers verwendet wird. Dabei werden viele Vorteile von C++ verschenkt. Um nur einige zu nennen:

- In C++ werden die fehleranfälligen Zeiger viel seltener als in C benötigt.
- Die Stringklassen lassen sich wesentlich einfacher und risikoloser als die nullterminierten Strings von C verwenden.
- Die Containerklassen der C++-Standardbibliothek haben viele Vorteile gegenüber Arrays, selbstdefinierten verketteten Listen oder Bäumen.
- Exception-Handling bietet eine einfache Möglichkeit, auf Fehler zu reagieren.
- Objektorientierte Programmierung ermöglicht übersichtlichere Programme.
- Templates sind die Basis für eine außerordentlich vielseitige Standardbibliothek.

Ich habe versucht, bei allen Konzepten nicht nur die Sprachelemente und ihre Syntax zu beschreiben, sondern auch Kriterien dafür anzugeben, wann und wie man sie sinnvoll einsetzen kann. Deshalb wurde z.B. mit der objektorientierten Programmierung eine Einführung in die objektorientierte Analyse und das objektorientierte Design verbunden. Ohne die Beachtung von Design-Regeln schreibt man leicht Klassen, die der Compiler zwar übersetzen kann, die aber kaum hilfreich sind.

Man hört immer wieder die Meinung, dass C++ viel zu schwierig ist, um es als einführende Programmiersprache einzusetzen. Dieses Buch soll ein in vielen Jahren erprobtes Gegenargument zu dieser Meinung sein. Damit will ich aber die Komplexität von C++ überhaupt nicht abstreiten.

Zahlreiche Übungsaufgaben geben dem Leser die Möglichkeit, die Inhalte praktisch anzuwenden und so zu vertiefen. Da man Programmieren nur lernt, indem man es tut, möchte ich ausdrücklich dazu ermuntern, zumindest einen Teil der Aufgaben zu lösen und sich dann selbst neue Aufgaben zu stellen. Der Schwierigkeitsgrad der Aufgaben reicht von einfachen Wiederholungen des Textes bis zu kleinen Projektchen, die ein gewisses Maß an selbständiger Arbeit erfordern. Die Lösungen der meisten Aufgaben findet man auf der beiliegenden CD und auf meiner Internetseite *http://www.rkaiser.de*.

Anregungen, Korrekturhinweise und Verbesserungsvorschläge sind willkommen. Bitte senden Sie diese an die Mail-Adresse auf meiner Internetseite.

Bei meinen Schulungskunden und Studenten bedanke ich mich für die zahlreichen Anregungen und anregenden Fragen. Herrn Engesser und seinem Team vom Springer-Verlag danke ich für die Unterstützung und Geduld.

Tübingen, im Juni 2009 Richard Kaiser

Inhalt

Teil 1: Windows .NET Programme mit Visual Studio

Teil 2: Standard-C++

Teil 3: C++/CLI und die .NET Bibliothek

ϴ Angesichts des Umfangs dieses Buches habe ich einige Abschnitte mit dem Zeichen ϴ in der Überschrift als „weniger wichtig“ gekennzeichnet. Damit will ich dem Anfänger eine kleine Orientierung durch die Fülle des Stoffes geben. Diese Kennzeichnung bedeutet aber keineswegs, dass dieser Teil unwichtig ist – vielleicht sind gerade diese Inhalte für Sie besonders relevant.

Teil 1

Windows .NET Programme

mit Visual Studio

1 Die Entwicklungsumgebung

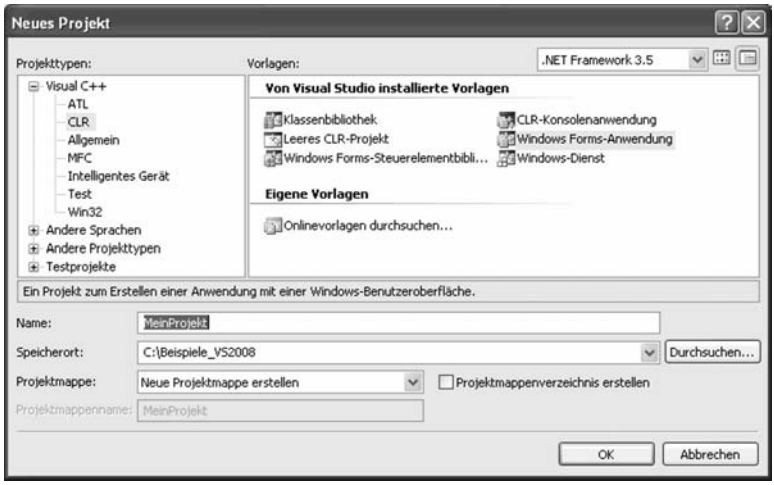
Visual Studio besteht aus verschiedenen Werkzeugen (Tools), die einen Programmierer bei der Entwicklung von Software unterstützen. Eine solche Zusammenstellung von Werkzeugen zur Softwareentwicklung bezeichnet man auch als Programmier- oder **Entwicklungsumgebung**.

Einfache Entwicklungsumgebungen bestehen nur aus einem Editor und einem Compiler. Für eine effiziente Entwicklung von komplexeren Anwendungen (dazu gehören viele Windows-Anwendungen) sind aber oft weitere Werkzeuge notwendig. Wenn diese wie in Visual Studio in einem einzigen Programm integriert sind, spricht man auch von einer **integrierten Entwicklungsumgebung** (engl.: „integrated development environment“, **IDE**).

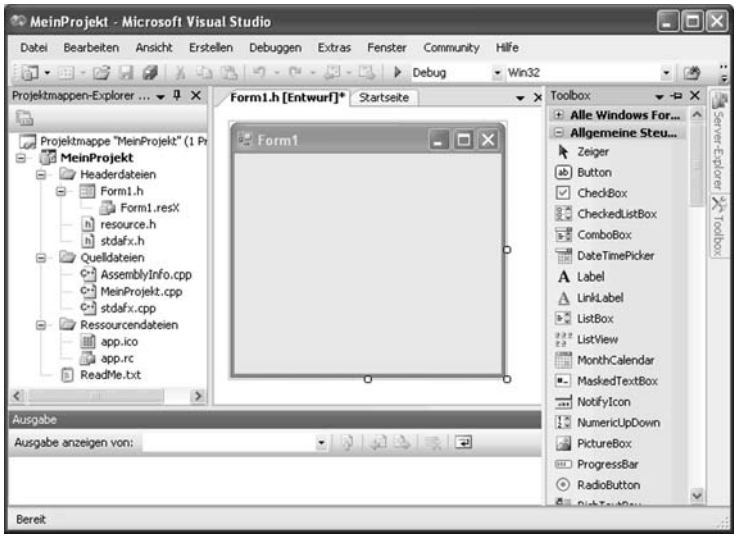
In diesem Kapitel wird zunächst an einfachen Beispielen gezeigt, wie man mit Visual Studio 2008 Windows-Programme mit einer grafischen Benutzeroberfläche entwickeln kann. Anschließend (ab Abschnitt 1.3) werden dann die wichtigsten Werkzeuge von Visual Studio ausführlicher vorgestellt. Für viele einfache Anwendungen (wie z.B. die Übungsaufgaben) reichen die Abschnitte bis 1.7. Die folgenden Abschnitte sind nur für anspruchsvollere oder spezielle Anwendungen notwendig. Sie sind deshalb mit dem Zeichen Θ (siehe Seite xxiv) gekennzeichnet und können übergangen werden. Weitere Elemente der Entwicklungsumgebung werden später beschrieben, wenn sie dann auch eingesetzt werden können.

1.1 Visuelle Programmierung: Ein erstes kleines Programm

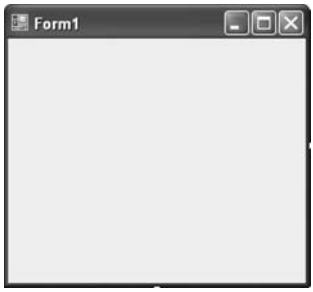
In Visual Studio 2008 findet man unter ***Datei|Neu|Projekt*** Vorlagen für verschiedene Arten von Anwendungen. Eine Windows-Anwendung mit einer graphischen Benutzeroberfläche erhält man mit dem Projekttyp ***CLR*** (Common Language Runtime) und der Vorlage ***Windows Forms-Anwendung***. Ein solches Projekt legt man dann an, indem man nach *Name* einen Namen und nach *Speicherort* ein Verzeichnis für das Projekt eingibt und dann den OK-Button anklickt:



Anschließend werden einige Tools der IDE angezeigt. Die hier rechts eingeblendete **Toolbox** wird angezeigt, wenn man mit der Maus über den Toolbox-Rider fährt, oder mit *Ansicht|Toolbox*:



Das **Formular** (hier *Form1*) ist der Ausgangspunkt für alle Windows Forms Anwendungen, die mit Visual Studio 2008 entwickelt werden. Es entspricht dem Fenster, das beim Start des Programms angezeigt wird:

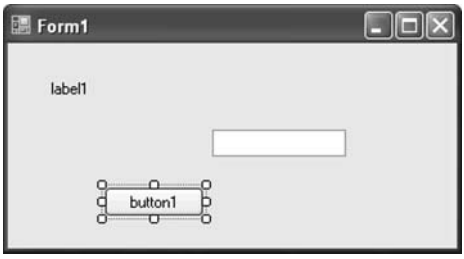


Ein Formular kann mit den in der **Toolbox** verfügbaren **Steuerelementen** (**Controls**) gestaltet werden. Die Toolbox zeigt praktisch alle der unter Windows üblichen Steuerelemente an, wenn das Formular angezeigt wird. Sie sind auf verschiedene Gruppen verteilt (z.B. *Allgemeine Steuerelemente*, *Container* usw.), die über die Icons + und – auf- und zugeklappt werden können. Ein Teil dieser Komponenten (wie z.B. ein Button) entspricht Steuerelementen, die im laufenden Programm angezeigt werden. Andere, wie ein *Timer* aus der Gruppe *Komponenten*, sind im laufenden Programm nicht sichtbar.

Falls Ihnen die Namen und die kleinen Icons nicht allzu viel sagen, lassen Sie einfach den Mauszeiger kurz auf einer Komponente stehen: Dann erscheint ein kleines gelbes Hinweisfenster mit einer kurzen Beschreibung.

Um eine **Komponente** aus der Toolbox **auf das Formular** zu **setzen**, zieht man sie einfach von der Toolbox auf das Formular. Oder man klickt mit der Maus zuerst auf die Komponente (sie wird dann als markiert dargestellt) und dann auf die Stelle im Formular, an die ihre linke obere Ecke kommen soll.

Beispiel: Nachdem man ein Label (Zeile sieben in *Allgemeine Steuerelemente*, mit dem großen *A*), eine TextBox (vierte Zeile von unten, Aufschrift *ab*) und einen Button (zweite Zeile mit der Aufschrift *ab*) auf das Formular gesetzt hat, sieht es etwa folgendermaßen aus:



Durch diese Spielereien haben Sie **schon ein richtiges Windows-Programm** erstellt – zwar kein besonders nützliches, aber immerhin. Sie können es folgendermaßen starten:

– mit *Debuggen|Debuggen starten* von der Menüleiste, oder
– mit *F5* von einem beliebigen Fenster in Visual Studio oder
– durch den Aufruf der vom Compiler erzeugten Exe-Datei.

Dieses Programm hat schon viele Eigenschaften, die man von einem Windows-Programm erwartet: Man kann es mit der Maus verschieben, vergrößern, verkleinern und schließen.

Bemerkenswert an diesem Programm ist vor allem der im Vergleich zu einem nichtvisuellen Entwicklungssystem **geringe Aufwand**, mit dem es erstellt wurde. So braucht Petzold in seinem Klassiker „Programmierung unter Windows“ (Petzold 1992, S. 33) ca. 80 Zeilen nichttriviale C-Anweisungen, um den Text „Hello Windows“ wie in einem Label in ein Fenster zu schreiben. Und in jeder dieser 80 Zeilen kann man einiges falsch machen.

Vergessen Sie nicht, Ihr **Programm** zu **beenden**, bevor Sie es weiterbearbeiten. Sie können den Compiler nicht erneut starten, solange das Programm noch läuft.

Diese Art der Programmierung bezeichnet man als **visuelle Programmierung**. Während man bei der konventionellen Programmierung ein Programm ausschließlich durch das Schreiben von Anweisungen (Text) in einer Programmiersprache entwickelt, wird es bei der visuellen Programmierung ganz oder teilweise aus vorgefertigten grafischen Komponenten zusammengesetzt.

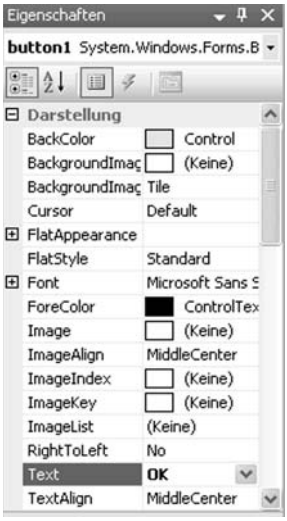
Mit Visual Studio kann die Benutzeroberfläche eines Programms visuell gestaltet werden. Damit sieht man bereits beim Entwurf des Programms, wie es später zur Laufzeit aussehen wird. Die Anweisungen, die als Reaktionen auf Benutzereingaben (Mausklicks usw.) erfolgen sollen, werden dagegen konventionell in einer Programmiersprache (z.B. C++) geschrieben.

Wenn die Toolbox oft verwendet wird, ist es am einfachsten, sie in der Entwicklungsumgebung zu fixieren, indem man in ihrem Kontextmenü (rechte Maustaste) „Automatisch im Hintergrund“ deaktiviert.

Die zuletzt auf einem Formular (bzw. im Pull-down-Menü des Eigenschaftenfensters) angeklickte Komponente wird als die **aktuell ausgewählte Komponente**

bezeichnet. Man erkennt sie an den kleinen Quadraten an ihrem Rand, den sogenannten **Ziehquadraten**. An ihnen kann man mit der Maus ziehen und so die Größe der Komponente verändern. Ein **Formular** wird dadurch zur aktuell ausgewählten Komponente, dass man mit der Maus eine freie Stelle im Formular anklickt.

Beispiel: Im letzten Beispiel ist *button1* die aktuell ausgewählte Komponente.



Im **Eigenschaftenfenster** (Kontextmenü der Komponente auf dem Formular, oder *Ansicht|Weitere Fenster|Eigenschaftenfenster*, nicht mit *Ansicht|-Eigenschaften-Manager* verwechseln) werden die Eigenschaften (properties) der aktuell ausgewählten Komponente angezeigt. In der linken Spalte stehen die **Namen** und in der rechten die **Werte** der Eigenschaften. Mit der Taste *F1* erhält man eine Beschreibung der Eigenschaft.

Den Wert einer Eigenschaft kann man über die rechte Spalte verändern. Bei manchen Eigenschaften kann man den neuen Wert über die Tastatur eintippen. Bei anderen wird nach dem Anklicken der rechten Spalte ein kleines Dreieck für ein Pull-down-Menü angezeigt, über das ein Wert ausgewählt werden kann. Oder es wird ein Symbol mit drei Punkten „…“ angezeigt, über das man Werte eingeben kann.

Beispiel: Bei der Eigenschaft ***Text*** kann man mit der Tastatur einen Text eingeben. Bei einem Button ist dieser Text die Aufschrift auf dem Button (z.B. „OK“), und bei einem Formular die Titelzeile (z.B. „Mein erstes C++-Programm“).
Bei der Eigenschaft ***BackColor*** (z.B. bei einem Button) kann man über ein Pull-down-Menü die **Hintergrundfarbe** auswählen.
Klickt man die rechte Spalte der Eigenschaft ***Font*** und danach das Symbol „…“ an, kann man die **Schriftart** der Eigenschaft *Text* auswählen.

Eine Komponente auf dem Formular wird nicht nur an ihre Eigenschaften im Eigenschaftenfenster angepasst, sondern auch umgekehrt: Wenn man die Größe einer Komponente durch Ziehen an den Ziehquadraten verändert, werden die Werte der entsprechenden Eigenschaften (*Location* und *Size* im Abschnitt *Layout*) im Eigenschaftenfenster automatisch aktualisiert.

1.2 Erste Schritte in C++

Als nächstes soll das Programm aus dem letzten Abschnitt so erweitert werden, dass als Reaktion auf Benutzereingaben (z.B. beim Anklicken eines Buttons) Anweisungen ausgeführt werden.

Windows-Programme können Benutzereingaben in Form von Mausklicks oder Tastatureingaben entgegennehmen. Im Unterschied zu einfachen Konsolen-Programmen (z.B. DOS-Programmen) muss man in einem Windows-Programm aber keine speziellen Funktionen (wie z.B. *scanf* in C) aufrufen, die auf solche Eingaben warten. Stattdessen werden alle Eingaben von Windows zentral entgegengenommen und als sogenannte Botschaften (Messages) an das entsprechende Programm weitergegeben. Dadurch wird in diesem Programm ein sogenanntes **Ereignis** ausgelöst.

Die Ereignisse, die für die aktuell ausgewählte Komponente eintreten können, zeigt das Eigenschaftenfenster an, wenn man das Icon für die *Events* (**Ereignisse**) anklickt.

Die Abbildung rechts zeigt einige Ereignisse für einen Button. Dabei steht *Click* für das Ereignis, das beim Anklicken des Buttons eintritt. Offensichtlich kann ein Button nicht nur auf das Anklicken reagieren, sondern auch noch auf zahlreiche andere Ereignisse.

Einem solchen Ereignis kann eine Funktion zugeordnet werden, die dann aufgerufen wird, wenn das Ereignis eintritt. Diese Funktion wird auch als **Ereignisbehandlungsroutine** (engl. **event handler**) bezeichnet. Sie wird von Visual Studio durch einen Doppelklick auf die Zeile des Ereignisses erzeugt und im **Quelltexteditor** angezeigt. Der Cursor steht dann am Anfang der Funktion.

Vorläufig soll unser Programm allerdings nur auf das Anklicken eines Buttons reagieren. Die bei diesem Ereignis aufgerufene Funktion erhält man am einfachsten durch einen Doppelklick auf den Button im Formular. Dadurch erzeugt Visual Studio die folgende Funktion: