Kapitel 2

Visual Studio 2010 und Visual C# 2010

In diesem Kapitel:

Dirk Louis, Shinja Strasser, Thorsten Kansy: Microsoft Visual C# - Das Entwicklerbuch, Microsoft Press 2010 (ISBN 978-3-86645-529-0)

Visual Studio

Entwickler, die für die Microsoft Windows-Plattform programmieren, schätzen seit jeher die Annehmlichkeiten, wie sie nur integrierte Entwicklungsumgebungen, kurz IDEs,[1] bieten. Eine der leistungsfähigsten IDEs, die von vielen Windows-Entwicklern favorisiert wird, ist Microsoft Visual Studio.

Der Name *Visual Studio* steht aber nicht nur für eine integrierte Entwicklungsoberfläche, sondern gleichermaßen für ein ganzes Entwicklungssystem und ein mächtiges Softwarepaket. Die Gründe hierfür liegen in der Geschichte der Microsoft-Entwicklungsumgebungen.

Geschichte

Die erste integrierte Entwicklungsumgebung wurde Anfang der achtziger Jahre von keinem geringeren als Anders Hejlsberg, dem Chef-Designer der Sprache C#, entwickelt. Damals arbeitete Hejlsberg allerdings noch nicht für Microsoft, und die von ihm konzipierte Entwicklungsumgebung diente der Pascal-Programmierung, lief noch unter MS-DOS und wurde als Turbo Pascal von Borland vermarktet. Sie vereinigte in sich Editor, Compiler und Linker – ein integrierter Debugger kam später hinzu.

Die Idee der integrierten Entwicklungsumgebung wurde schnell auch von anderen Softwareherstellern aufgegriffen und auf andere Sprachen ausgedehnt. Ein weiterer Quantensprung waren dann die ersten grafischen Entwicklungsumgebungen für 16-Bit-Windows – allen voran die Microsoft-Produkte für Basic und C++, die fortan die Bezeichnung *Visual* im Namen trugen.

HINWEIS Ein Wort zur Terminologie: *Visual C++* ist der Name eines Microsoft-Programmpakets, das alle Komponenten enthält, die zur Erstellung von Anwendungen mit C++ erforderlich sind. Traditionell sind dies Build-Werkzeuge, eine integrierte Entwicklungsumgebung, Bibliotheken und anderes mehr. *Visual C++* wird aber auch als Synonym für das gesamte Entwicklungssystem wie auch für die zugrunde liegende Programmiersprache verwendet.

In den nächsten Jahren wurden die verschiedenen Visual-Produkte relativ unabhängig voneinander weiterentwickelt. Während die Visual Basic-Entwicklungsumgebung beispielsweise von Beginn an die visuelle Erstellung von GUI-Anwendungen unterstützte, blieb Visual C++ der traditionellen Quelltextbearbeitung verbunden und bot lediglich für die grafische Erstellung von Dialogfeldern und Menüs spezielle Editoren an. Nicht wenige Softwareunternehmen gingen daher dazu über, sich gezielt der jeweiligen Stärken der verschiedenen Sprachen und ihrer Entwicklungssysteme zu bedienen und je nach Aufgabengebiet wahlweise das eine oder andere Visual-Produkt einzusetzen.

Den Bedarf erkennend begann Microsoft die beliebtesten Visual-Produkte zu einem Paket zu schnüren – Visual Studio war geboren. Gleichzeitig mühte sich Microsoft, die Arbeit von Programmierern, die verschiedene Produkte parallel einsetzten, durch Angleichung der IDEs zu vereinfachen. Dieses Unterfangen erwies sich jedoch als überaus schwierig, und so wurde Visual Studio noch in der Version 6 mit drei verschiedenen IDEs ausgeliefert (eine für Visual J++ und Visual InterDev, eine für Visual Basic und eine für Visual C++).

Erst 2002, im Fahrwasser der .NET-Initiative, gelang die angestrebte Integration der einzelnen Visual-Produkte – mittlerweile ergänzt um Visual C# – unter einer echten, gemeinsamen Visual Studio-IDE für alle damals im Paket enthaltenen Sprachen (Visual Basic, Visual C#, Visual C++ und Visual J#).

[1] Abkürzung für *Integrated Development Environment*.

Dirk Louis, Shinja Strasser, Thorsten Kansy: Microsoft Visual C# - Das Entwicklerbuch, Microsoft Press 2010 (ISBN 978-3-86645-529-0)

Visual Studio 2010, .NET Framework 4.0 und C# 4.0

Die aktuelle Version, Microsoft Visual Studio 2010,[2] gibt es in vier Editionen (Professional, Premium, Ultimate und Test Professional). Zudem sind die im Visual Studio-Paket enthaltenen Sprachen Visual Basic, Visual C#, Visual C++ und Visual Web Developer auch als kostenlose Einzelpakete, den so genannten Express-Editionen mit leicht reduziertem Funktionsumfang erhältlich.

Die wichtigsten Neuerungen in Visual Studio 2010, .NET Framework 4.0 und C# 4.0 sind:

- Einbettung von Typeninformationen in Assemblys
- zahlreiche neue IDE-Funktionen wie Aufrufhierarchie, verbessertes Docking, konfigurierbares IntelliSense, Zoom, neue Navigations- und Suchoptionen (siehe Kapitel 2)
- Unterstützung mehrerer Monitore (siehe Kapitel 2, Abschnitt zur Andock-Hilfe)
- Ko- und Kontravarianz für generische Lösungen (siehe Kapitel 20)
- zahlreiche, kleinere Erweiterungen in den Klassen des .NET Framework wie z. B.
  - eine neue Methode `String.IsNullOrWhitespace()` zum Erkennen leerer Strings (siehe Kapitel 22),
  - `String.Concat()` und `String.Join()` zum leichteren Umwandeln von Auflistungen und Arrays in Strings (siehe Kapitel 22),
  - den Methoden `EnumerateDirectories()` und `EnumerateFiles()` der Klassen `Directory` und `DirectoryInfo` zum Auflisten von Unterverzeichnissen und Dateien (siehe Kapitel 23)
- einer neuen Auflistungsklasse `SortedSet<T>` (siehe Kapitel 24)
- drei neue Utility-Klassen `BigInteger`, `Complex` und `Tupel` (für `BigInteger` und `Complex` siehe Kapitel 25)
- verbessertes Debugging (siehe Kapitel 50)
- Unterstützung für die Test-First-Entwicklung, Komponententests jetzt auch in der Professional-Version
- »Install Shield 2010 Limited Edition«-Projektvorlage für die Softwarebereitstellung
- diverse Neuerungen in C# 4.0 (siehe Kapitel 48)

Arbeiten in der Entwicklungsumgebung

Bevor wir in den nächsten Abschnitten detaillierter auf die wichtigsten Komponenten der Visual Studio/C#-Entwicklungsumgebung eingehen, erlauben Sie uns einige allgemeine Anmerkungen, damit sich auch Leser, für die dies die erste Begegnung mit der Visual-IDE ist, orientieren können.

Die Startseite

Jedes Mal, wenn Sie Visual Studio aufrufen, werden Sie mit der Startseite begrüßt (siehe Abbildung 2.1).

Vorausgesetzt, Ihr Rechner ist online, können Sie sich jetzt über aktuelle Neuigkeiten rund um Visual Studio, C# und andere Microsoft-Technologien informieren (Bereich *Aktuelle Nachrichten*).

[2] Falls Sie sich fragen, was aus .NET geworden ist: Das Suffix ist zwar aus dem Namen verschwunden, die entsprechende Funktionalität ist aber natürlich geblieben. Es wird nur nicht mehr so offensiv damit geworben.

Entwickler, die mit Visual Studio noch nicht vertraut sind, finden unter *Erste Schritte* eine Sammlung von Links zu Infoseiten, die ihnen weiterhelfen. Die Linksammlung ist thematisch geordnet, die einzelnen Themen können über die Menülinks *Willkommen*, *Windows*, *Web*, *Wolke* etc. ausgewählt werden. Fortgeschrittene Entwickler mögen die Links im Bereich *Leitfaden und Ressourcen* ganz hilfreich finden.

Und für die tägliche Arbeit gibt es auch noch etwas: Im linken Bereich finden Sie Links zum Erstellen neuer oder zum Öffnen bestehender und zuletzt bearbeiteter Projekte, die Ihnen den Umweg über die entsprechenden Befehle im *Datei*-Menü ersparen.

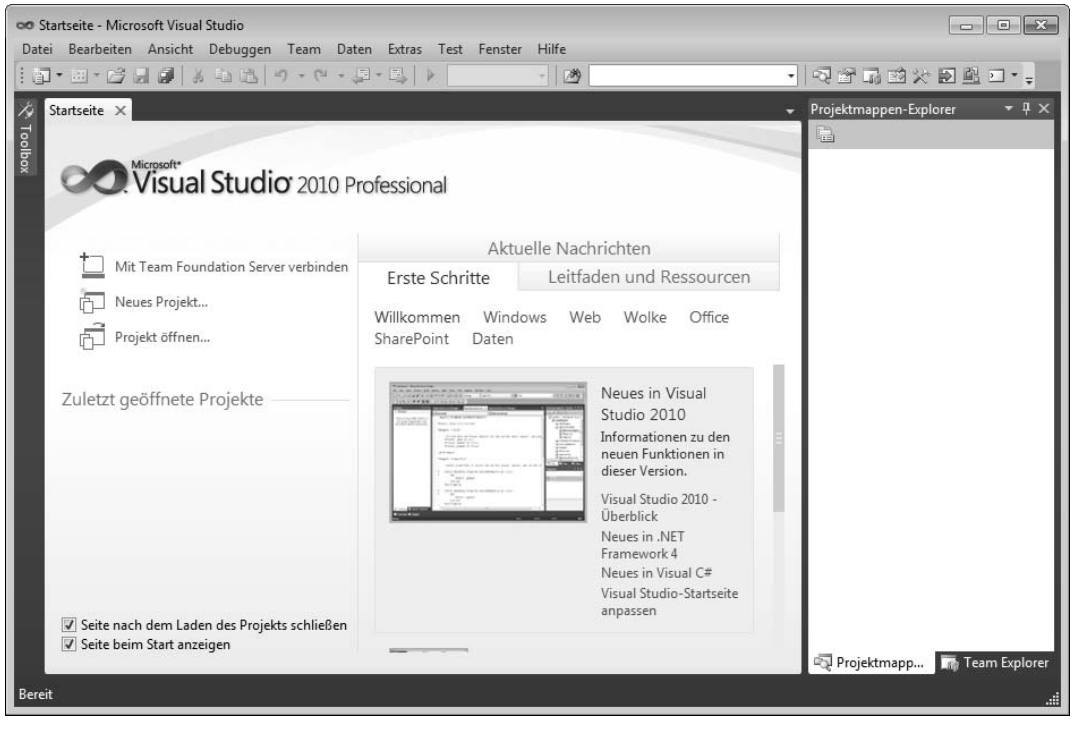

Abbildung 2.1 Die Startseite von Visual Studio 2010

HINWEIS Wenn Sie mit Visual Studio arbeiten, werden Sie beim ersten Start der IDE aufgefordert, eine der »vordefinierten Umgebungseinstellungen« auszuwählen. Als C#-Entwickler werden Sie sich höchstwahrscheinlich für die C#-Einstellungen entscheiden, auf die sich auch die nachfolgenden Erläuterungen und Abbildungen beziehen.

Sollten Sie sich für eine andere Option entschieden haben – Entwickler, die neben C# auch des Öfteren andere Programmiersprachen wie Visual Basic oder Visual C++ einsetzen, werden vielleicht die allgemeinen Umgebungseinstellungen vorziehen –, werden Sie gelegentlich leichte Abweichungen feststellen. So hängt z.B. die Auswahl der im rechten Fenster angezeigten Nachrichten von den ausgewählten Einstellungen ab.

Über den Menübefehl *Extras/Einstellungen importieren und exportieren* können Sie die getroffene Auswahl jederzeit rückgängig machen und sich für andere Umgebungseinstellungen entscheiden.

Softwareerstellung

Visual Studio wurde dazu konzipiert, Ihnen – dem Entwickler – die tägliche Arbeit mit Quelltexten, Anwendungen und Benutzeroberflächen zu erleichtern. Voraussetzung ist allerdings, dass Sie mit den für IDEs typischen Arbeitsschritten vertraut sind:

1. Schritt: Projekt anlegen

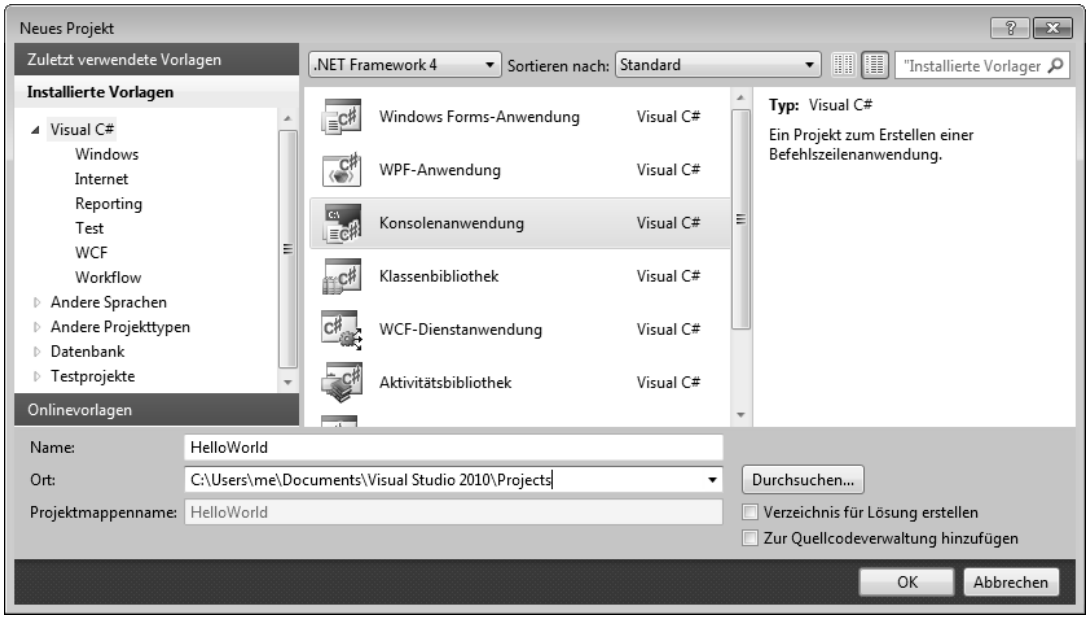

Abbildung 2.2 Ein neues Projekt beginnen

Anwendungen werden in Visual Studio in Form von Projekten verwaltet. Der erste Schritt bei der Anwendungserstellung besteht daher darin, ein neues Projekt anzulegen (Befehl *Datei/Neu/Projekt*). Als Ausgangspunkt wählen Sie eine Projektvorlage, die dem anvisierten Ziel entspricht (Konsolenanwendung, Windows Forms-Anwendung, Klassenbibliothek etc.).

Das Spektrum der angebotenen Vorlagen hängt davon ab, welche .NET Framework-Version Sie in dem Dropdown-Listenfeld oben ausgewählt haben. Im Allgemeinen werden Sie die aktuelle Version 4 beibehalten. Zur besseren Abwärtskompatibilität können Sie aber auch Version 2.0 oder 3.5 auswählen.

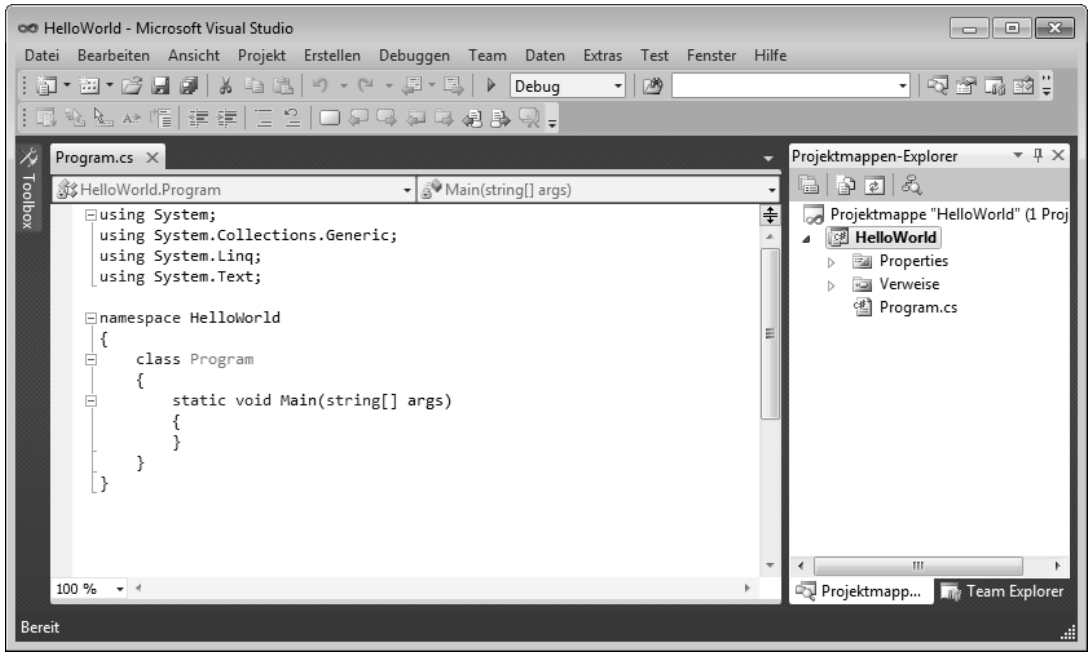

Abbildung 2.3 Das neue Projekt in der IDE

Das neu angelegte Projekt wird automatisch in die IDE geladen und das Erscheinungsbild der IDE wandelt sich:

- Das Menüsystem wird um die Menüs *Projekt* und *Erstellen* erweitert.
- Im Projektmappen-Explorer erscheint eine hierarchische Visualisierung des Projekts.

  Die aufgeführten Projektelemente können per Doppelklick in passende Editoren geladen oder mithilfe der Kontextmenü-Befehle bearbeitet werden.
- Enthält die Projektvorlage eine Standardquelldatei wird diese automatisch zur Bearbeitung in den zugehörigen Editor geladen. Für Konsolenanwendungen ist dies beispielsweise eine *.cs*-Quelldatei mit Anwendungsgerüst; für Windows Forms-Anwendungen ist es die *.cs*-Quelldatei des Formulars, die in den Designer geladen wird.

Bestehende Projekte können Sie jederzeit um zusätzliche Elemente wie Formulare, Klassen, Readme-Dateien, Verweise etc. erweitern. Entsprechende Befehle finden Sie im Kontextmenü des Projektknotens sowie im Menü *Projekt*.

HINWEIS Der Projektmappen-Explorer ist standardmäßig rechtsseitig an den Rahmen der IDE angedockt. Sollte er nicht zu sehen sein, können Sie ihn mit dem Befehl *Ansicht/Projektmappen-Explorer* aufrufen.

2. Schritt: Quelltexte bearbeiten

Über den Projektmappen-Explorer können Sie die Quelldateien des Projekts zur Bearbeitung in die zugehörigen Editoren und Designer laden. Dateien, die nicht Teil des aktuellen Projekts sind, können über den Menübefehl *Datei/Öffnen/Datei* geladen werden.

Visual Studio verfügt über eine Vielzahl integrierter Editoren, beispielsweise den Text-Editor für einfache Textdateien, den CSS-Editor für CSS-Stylesheets, den XML-Editor für XML-Dokumente, den Code-Editor für Quelltextdateien.

Für manche Projektelemente gibt es sogar mehrere Ansichten oder Editoren. So können Formulardateien sowohl im Code-Editor als auch im Windows Forms-Designer bearbeitet werden.

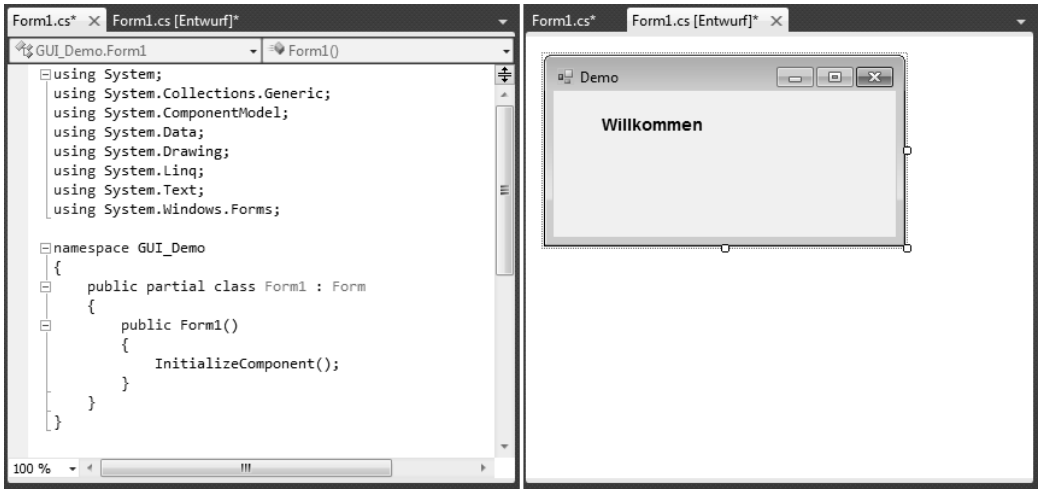

Abbildung 2.4 Ein Formular im Code-Editor und im Windows Forms-Designer

Formulardateien sind *.cs*-Quelldateien, in denen eine von `Form` abgeleitete Klasse für ein GUI-Fenster definiert wird.[3] Wenn Sie eine solche Datei über den Projektmappen-Explorer öffnen, wird sie automatisch in den Windows Forms-Designer geladen, wo sie grafisch bearbeitet werden kann.

Um den *.cs*-Code einer Formulardatei zur Bearbeitung in den Code-Editor zu laden, gehen Sie so vor, dass Sie

- entweder im Kontextmenü des Windows Forms-Designers den Befehl *Code anzeigen* aufrufen
- oder im Projektmappen-Explorer den Befehl *Code anzeigen* aus dem Kontextmenü des Dateiknotens aufrufen
- oder im Windows Forms-Designer eine Behandlungsmethode für eines der Ereignisse des Formulars oder einer eingebetteten Komponente einrichten, woraufhin die IDE automatisch in den Quelltext wechselt.

Der Code-Editor mit seinen vielfältigen Besonderheiten wird im Abschnitt »Der Code-Editor« in diesem Kapitel näher vorgestellt. Dem Windows Forms-Designer ist ein eigener Abschnitt in Kapitel 26 gewidmet.

[3] Damit die Quelldatei von der IDE als Formular-Datei erkannt wird, muss sie zusätzlich den Namespace `System.Windows.Forms` einbinden.

3. Schritt: Projekt erstellen

Um die Quelldateien des Projekts in ausführbaren IL-Code zu kompilieren, verwenden Sie die Befehle im Menü *Erstellen*. Die oberste Gruppe von Menübefehlen dient der Erstellung der kompletten Projektmappe mit allen enthaltenen Projekten, die zweite Befehlgruppe bezieht sich auf das aktuelle Projekt.

Ob Projektmappe oder aktuelles Projekt, in beiden Fällen können Sie zwischen einer Aktualisierung und einer kompletten Neuerstellung wählen. Im ersten Fall werden nur die Quelltextdateien kompiliert, die seit dem letzten Build (Erstellen) bearbeitet wurden (*Erstellen*-Befehle), im zweiten Fall werden alle Quelltextdateien neu kompiliert (*Neu erstellen*-Befehle).

Stößt der Compiler bei der Kompilierung der Quelltextdateien auf fehlerhaften oder zumindest verdächtigen Code, gibt er im Fenster der Fehlerliste Fehlermeldungen oder Warnungen aus. Sofern die Fehlerliste nicht bereits geöffnet ist, wird sie automatisch am unteren Rand der IDE eingeblendet.

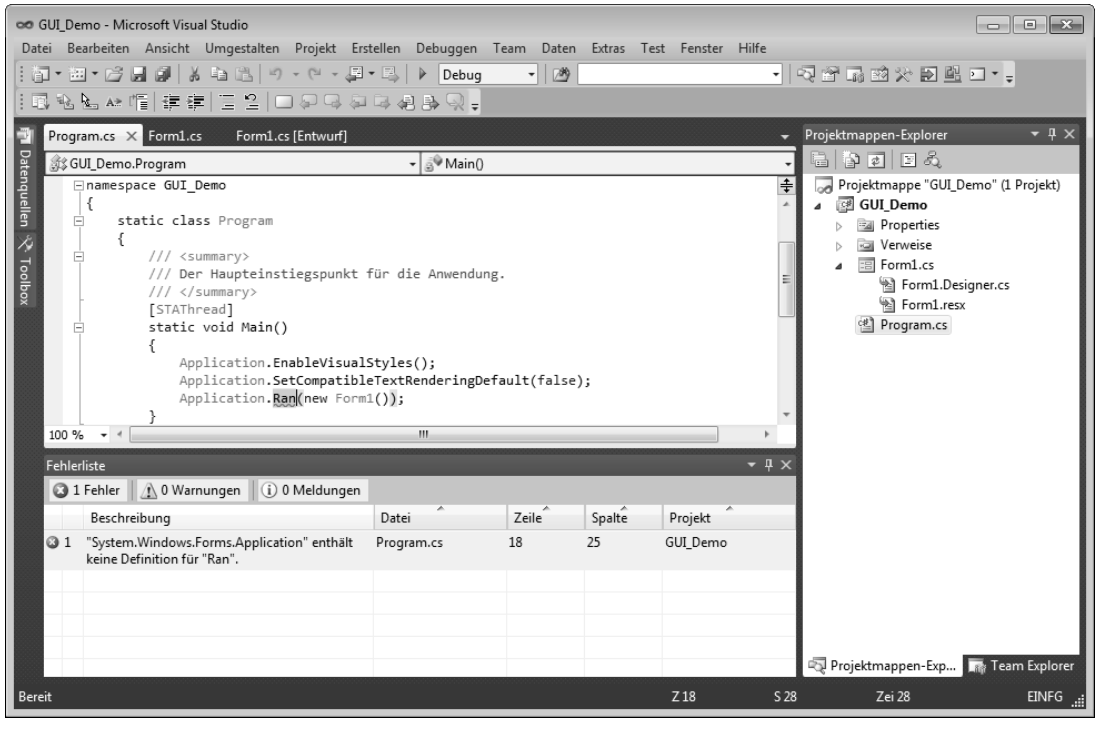

Abbildung 2.5 Fehlerliste und Markierung fehlerhaften Codes im Editor nach einem Doppelklick auf die Fehlermeldung

- Wenn Sie in der Fehlerliste auf eine Fehlermeldung doppelklicken, springt der Code-Editor zu der Stelle im Quelltext, die die Meldung auslöste, und markiert sie
- Über den Befehl *Hilfe zu Fehlern anzeigen* im Kontextmenü der Fehlerliste können Sie weiterführende Erläuterungen zu einem Fehler abrufen

4. Schritt: Projekt testen

Zu guter Letzt sollten Sie Ihren Code testen. Handelt es sich um ein Anwendungsprojekt, brauchen Sie dazu nur den Menübefehl *Debuggen/Starten ohne Debugging* aufzurufen. Zum Testen von einzelnen Modulen (beispielsweise Klassen oder Steuerelementen einer Bibliothek) benötigen Sie ein passendes Testprogramm, welches Sie am einfachsten als zusätzliches Projekt in der Projektmappe Ihres Moduls anlegen.

Sollten Sie beim Testen feststellen, dass Ihr Code nicht wunschgemäß arbeitet, können Sie ihn direkt in der IDE debuggen (siehe auch Kapitel 47).

HINWEIS In Kapitel 3 finden Sie Schritt-für-Schritt-Anweisungen zur Erstellung von Konsolen- und Windows-Anwendungen.

RAD-Entwicklung

Viele der in Visual Studio 2010 integrierten Werkzeuge und Features sind der »schnellen« Anwendungsentwicklung (RAD = *rapid application development*) verpflichtet, mit dem Ziel, Produktivität und Effizienz zu steigern und die Fehlerrate zu senken.

An erster Stelle sind hier natürlich der Windows Forms- und der WPF-Designer zu nennen, mit denen grafische Benutzeroberflächen visuell erstellt und bearbeitet werden können (siehe Kapitel 26 und 32).

Den Code-Editor wird man wohl kaum als RAD-Werkzeug bezeichnen, wohl aber die in den Editor integrierte Microsoft IntelliSense-Technologie zur Codevervollständigung oder die Codeausschnitte (siehe den Abschnitt »Der Code-Editor« in diesem Kapitel).

Aber auch der Klassen-Designer, dem wir uns in Kapitel 12 in angemessener Ausführlichkeit widmen werden, sollte in diesem Zusammenhang nicht vergessen werden.

Testen, analysieren, debuggen

Das Paradies hat einen Fehler: den Bug. Seit es Programme gibt, ringen Programmierer weltweit darum, ihre Software möglichst bug- und fehlerfrei zu halten. Visual Studio 2010 unterstützt sie dabei mit leistungsfähigen Analysewerkzeugen wie z.B.

- dem integrierten Debugger zum Aufspüren von Laufzeitfehlern (siehe Kapitel 47)
- den Test-Tools der Professional- und Team System-Editionen
- dem externen Tool Spy++ zur Überwachung von Fenstern, Threads und Meldungen von GUI-Anwendungen

Veröffentlichung und Installation

Wie findet die fertige Anwendung den Weg zum Kunden? Visual Studio unterstützt Sie selbst bei der Erzeugung passender Installationsdateien, die Sie über eine Webseite, ein Netzwerk oder eine CD veröffentlichen können. Die beiden grundlegenden Installationstechnologien sind:

- Windows Installer – der traditionelle Weg, Konsolen- und Windows-Anwendungen zu vertreiben, bei dem mithilfe eines Setup-Projekts eine *setup.exe*-Datei erstellt wird. (Die zugehörigen Projektvorlagen finden sie unter *Datei/Neu/Projekt/Andere Projekttypen/Setup und Bereitstellung* und dann wahlweise *InstallShield LE* oder *Visual Studio Installer*).
- ClickOnce – eine Technologie, die vor allem dem Endnutzer die Installation und Aktualisierung der Anwendung vereinfacht. Sie bietet dem Entwickler allerdings weniger Konfigurationsmöglichkeiten als der Windows Installer.

HINWEIS Windows Installer und ClickOnce können nicht alle möglichen Installationsszenarien abdecken. Für besondere Ansprüche muss daher womöglich eine Bereitstellungstechnologie eines Drittanbieters herangezogen werden.

Anpassbarkeit

Die Visual Studio-IDE kann in vielfältiger Weise an die Bedürfnisse und Gewohnheiten des Entwicklers angepasst werden. Dies beginnt mit der aus vielen Microsoft-Anwendungen bekannten frei konfigurierbaren Zusammenstellung der Symbolleisten (Menübefehl *Extras/Anpassen*), führt über das Setzen einzelner Optionen bis hin zur Arbeit mit ganzen Konfigurationssätzen und endet irgendwann mit der Programmierung von Projektvorlagen, Makros und Add-Ins.

Einige Highlights zur Konfiguration von Visual Studio werden wir uns im Abschnitt »Konfiguration« weiter hinten in diesem Kapitel ansehen.

Die Projektverwaltung

In Visual C# beginnt die Arbeit an einem Programm stets mit dem Anlegen eines zugehörigen Projekts. Ein Projekt ist im Grunde ein Bausatz für die Erstellung eines Programms. Es besteht aus:

- Quelldateien des Programms (*.cs*-Quelltextdateien, Ressourcendateien wie Bilddateien, Sounddateien, Readme-Dateien usw.),
- Verweise auf externe Assemblys,
- Informationen, wie die Quelltextdateien zu übersetzen sind,
- einem Verzeichnis auf Ihrer Festplatte, in dem die Dateien des Projekts abgelegt sind.

Vom Nutzen der Projektverwaltung

Solange man nur kleine Programme erstellt, die nur aus einer einzigen Quelltextdatei bestehen, kommt man auch ganz gut ohne Projektverwaltung aus.

Größere Programme sind dagegen zumeist in mehrere Quelltextdateien aufgeteilt, die jeweils für sich kompiliert werden und dann erst vom Linker zu einer ausführbaren Datei zusammengebunden werden. Müssen noch externe Ressourcen (Bilder, Sound etc.) oder zusätzliche Assemblys eingebunden werden, kann die Erstellung eines solchen Programms sehr aufwändig werden.

Hier setzt die Projektverwaltung an. Sie verwaltet die zu einem Programm gehörenden Dateien und stellt Ihnen Befehle zum Aufnehmen weiterer Quelldateien bzw. zum Entfernen bestehender Quelldateien zur Verfügung, sie organisiert die zu dem Programm gehörenden Dateien auf der Festplatte und speichert alle Informationen, die nötig sind, um aus den Programmdateien das fertige Programm zu erstellen.

Übersicht

Die integrierte Entwicklungsumgebung von Visual Studio stellt eine Reihe von Menübefehlen, Dialogfeldern und Fenstern zur Verfügung, mit deren Hilfe Projekte erzeugt, erweitert, überwacht und konfiguriert werden können.

Funktion	Menübefehl/Fenster	Beschreibung
Neues Projekt anlegen	*Datei/Neu/Projekt*	Im Dialogfeld *Neues Projekt* werden Sie aufgefordert, alle erforderlichen Angaben zu dem neuen Projekt zu machen (Projektvorlage, Name, Speicherort). Wenn Sie das Dialogfeld bestätigen, wird das neue Projekt angelegt.
Quelldateien hinzufügen	*Projekt/Neues Element hinzufügen* *Projekt/Vorhandenes Element hinzufügen* Kontextmenü des Projektknotens im Projektmappen-Explorer	Wie Sie sehen, gibt es mehrere Wege, Quelldateien in ein Projekt aufzunehmen. Teilweise führen die Befehle zu dem gleichen Ergebnis (sprich zu dem gleichen Dialogfeld). Unterscheiden muss man allerdings, ob man eine bereits bestehende Quelltextdatei in das Projekt aufnehmen will (*Projekt/Vorhandenes Element hinzufügen*) oder ob man eine ganz neue Datei anlegen und im Projekt abspeichern möchte (*Projekt/Neues Element hinzufügen*).
Quelldateien löschen	Projektmappen-Explorer	Markieren Sie im Projektmappen-Explorer die zu entfernende Datei und drücken Sie die Entf-Taste oder rufen Sie den Menübefehl *Bearbeiten/Löschen* auf. Wenn Sie im Kontextmenü der Datei den Befehl *Aus Projekt ausschließen* aufrufen, wird die Datei lediglich aus dem Projekt entfernt, nicht aber von der Festplatte gelöscht.
Projekte konfigurieren	Projektmappen-Explorer, Befehl *Eigenschaften*	Über die Eigenschaftenseiten des Projektknotens können Sie das Projekt konfigurieren (Ausgabeverzeichnis festlegen, Compiler-Optionen für Debuggen oder Optimierung einschalten etc.).
Projekte überwachen	Projektmappen-Explorer Klassenansicht Befehl *Bearbeiten/Suchen und Ersetzen/In Dateien suchen*	Dass die IDE Ihre Quelldateien in Form von Projekten überwacht, hat auch über die bequeme Programmerstellung hinaus noch seine Vorteile. So eignet sich der Projektmappen-Explorer bestens als zentrale Schaltstelle, von der aus Sie Dateien schnell per Doppelklick zur Bearbeitung in den Editor laden bzw. geladene Dateien ansteuern können. Die Klassenansicht (Aufruf über *Ansicht/Klassenansicht*) informiert Sie über die in Ihrem Programm verwendeten Klassen, Klassenmember, Schnittstellen etc. Mit dem Befehl zum Suchen in Dateien können Sie bequem in allen oder ausgewählten Dateien Ihres Projekts nach bestimmten Textstellen suchen.

Tabelle 2.1 Die wichtigsten Elemente der Projektverwaltung

Projektmappen

In Visual Studio werden Projekte in Projektmappen verwaltet. Eine Projektmappe ist nichts anderes als eine höhere Organisationsebene, die es erlaubt, mehrere Projekte zusammen zu verwalten. Auf diese Weise können Sie beispielsweise verschiedene Versionen eines Programms in einer Projektmappe verwalten, oder Sie legen eine Klassenbibliothek und ein Testprogramm oder eine Anwendung und eine DLL, die von dem Programm benötigt wird, in einer gemeinsamen Projektmappe ab.

Projekte und Projektmappen

Wenn es Ihnen lediglich darum geht, ein Projekt für ein einzelnes Programm zu erstellen, ist die Verwaltung in Projektmappen im Grunde nur unnötiger Ballast für Sie. Die IDE tut daher ihr Möglichstes, um Sie von diesem Ballast zu befreien.

So ist es zur Einrichtung eines neuen Projekts nicht nötig, zuerst eine Projektmappe zu erstellen und im nächsten Schritt ein Projekt in die Projektmappe aufzunehmen. Stattdessen legt man einfach ein neues Projekt an (Menübefehl *Datei/Neu/Projekt*) und lässt die Projektmappe für das Projekt automatisch erstellen – wobei man die Option *Verzeichnis für Lösung* zur Erstellung eines eigenen Projektmappenverzeichnisses im Dialogfeld *Neues Projekt* meist deaktiviert.

Dirk Louis, Shinja Strasser, Thorsten Kansy: Microsoft Visual C# - Das Entwicklerbuch, Microsoft Press 2010 (ISBN 978-3-86645-529-0)

Wenn Sie den Befehl zum Anlegen eines neuen Projekts aufrufen, während noch eine Projektmappe geöffnet ist, können Sie im Dialogfeld *Neues Projekt* auswählen, ob das Projekt der bestehenden Projektmappe hinzugefügt oder ob eine neue Projektmappe erstellt werden soll.

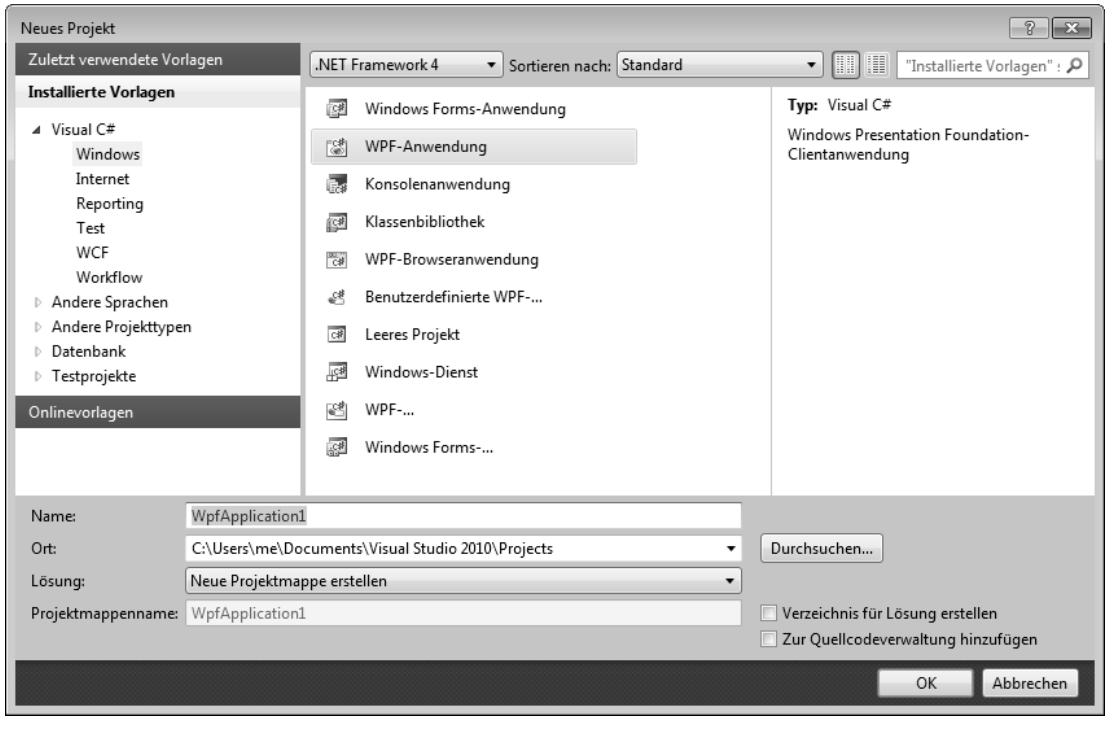

Abbildung 2.6 Dialogfeld zum Anlegen neuer Projekte und Projektmappen

HINWEIS Wenn Sie eine leere Projektmappe anlegen wollen, in die Sie nachträglich neue oder bestehende Projekte einfügen können, wählen Sie im Dialogfeld *Neues Projekt* den Typ *Andere Projekttypen/Visual Studio-Projektmappen/Leere Projektmappe* aus.

Der Projektmappen-Explorer

Der Projektmappen-Explorer ist die Schaltzentrale der Projektverwaltung.

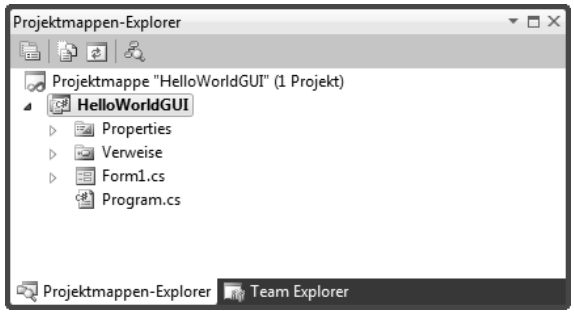

Abbildung 2.7 Der Projektmappen-Explorer

Die hierarchische Ansicht gibt Ihnen einen Überblick über den Aufbau Ihrer Projektmappen und Projekte:

- Über das Kontextmenü des Projektmappenknotens können Sie alle Projekte in der Projektmappe erstellen, den Erstellungsprozess konfigurieren oder der Projektmappe sonstige Dateien (Text, XML, Ressourcenvorlage etc.) hinzufügen
- Über die Kontextmenüs der Projektknoten können Sie das Projekt erstellen, debuggen, entfernen oder um weitere Elemente bereichern
- Über die Kontextmenüs der einzelnen Quelldateien können Sie diese entfernen, umbenennen, öffnen

TIPP Wenn Sie sich erst einmal ein wenig an die Arbeit mit dem Projektmappen-Explorer gewöhnt haben, werden Sie ihn gar nicht mehr missen wollen. Sollten Sie ihn aber einmal aus Versehen geschlossen haben, können Sie ihn jederzeit über den Befehl *Ansicht/Projektmappen-Explorer* oder mit der Tastenkombination Strg + W + S wieder anzeigen lassen.

Das Festplatten-Abbild

Visual Studio legt für jedes Projekt auf der Festplatte ein eigenes Projektverzeichnis an, das den Namen des Projekts trägt. Name und Speicherort des Projektverzeichnisses geben Sie im Dialogfeld *Neues Projekt* an.

HINWEIS Wenn Sie im Dialogfeld *Neues Projekt* das Kontrollkästchen *Verzeichnis für Lösung erstellen* aktivieren, wird unter dem angegebenen Speicherort zunächst ein Verzeichnis für die Projektmappe angelegt und erst darunter das Projektverzeichnis. Den Namen für das Projektmappenverzeichnis können Sie selbst wählen, ansonsten wird der Name des Projekts verwendet.

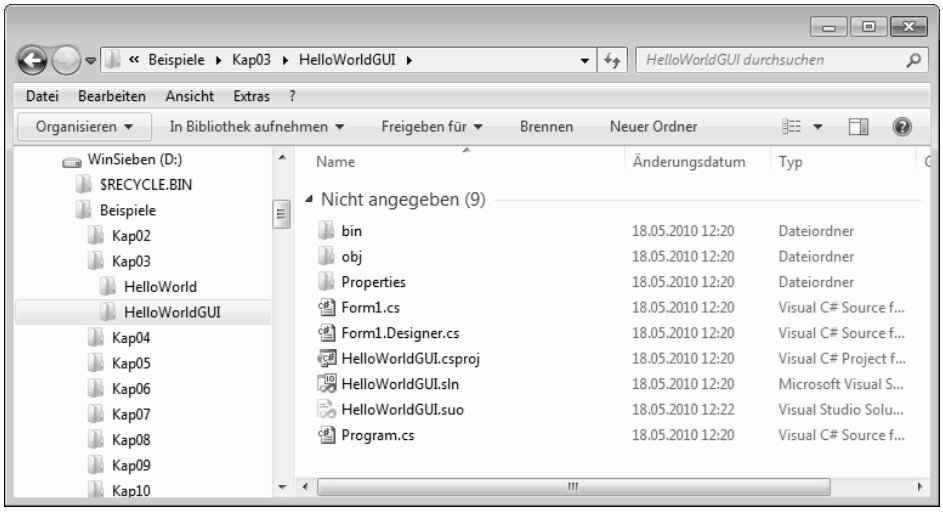

Abbildung 2.8 Ein Projekt auf der Festplatte (ohne Projektmappenverzeichnis)

Die wichtigsten Projektdateien

Neben den Quell- und Ressourcendateien, die Sie dem Projekt während der Arbeit an Ihrem Programm oder Ihrer Bibliothek selbst hinzufügen (C#-Quelltextdateien, Bilddateien, Sounddateien, Datenbanken, Readme-Dateien etc.), gehören zu jedem Projekt noch verschiedene weitere Dateien, die von der Projektverwaltung, dem Windows Forms-Designer, der Build-Engine oder anderen Tools der IDE erzeugt werden.

Dateityp	Beschreibung
Projektdateien	
.sln	Projektmappendatei Textdatei, in der allgemeine Projektmappen-Informationen – die Namen der in der Projektmappe enthaltenen Projekte, die verfügbaren Build-Konfigurationen, sonstige Projektmappendateien – gespeichert werden
.suo	Konfigurationsdatei der Projektmappe Binärdatei, in der benutzerspezifische Projektmappeneinstellungen (wie zum Beispiel Pfade) gespeichert werden. Bei Portierung der Projektmappe auf ein anderes System werden die meisten Informationen der *.suo*-Datei unbrauchbar.
.csproj	Projektdatei XML-Datei mit allen wichtigen Projekteigenschaften, etwa die Liste der Verweise oder die Build-Einstellungen
.csproj.user	Benutzerspezifische Projektdatei XML-Datei mit benutzerspezifischen Projekteigenschaften
Designer-Dateien für Windows Forms	
Formular.Designer.cs	Designer-Codedatei für Windows Forms-Formulare Wenn Sie im Windows Forms-Designer Formulare konfigurieren und mit Steuerelementen und anderen Komponenten bestücken, speichert der Designer den zugehörigen Quelltext in dieser *.cs*-Quelldatei
Formular.resx	Designer-Ressourcendatei für Formulare Ressourcendatei, in der der Designer komponentenspezifische Ressourcen (beispielsweise das Symbol einer Schaltfläche) abspeichert
Designer-Dateien für WPF	
Window.xaml	Designer-Daten zur Beschreibung eines WPF-Fensters
Window.xaml.cs	Designer-Codedatei für WPF-Fenster Wenn Sie im WPF-Designer Fenster konfigurieren und mit Steuerelementen und anderen Komponenten bestücken, speichert der Designer den zugehörigen Quelltext in dieser *.cs*-Quelldatei
Formular.resx	Designer-Ressourcendatei für Formulare Ressourcendatei, in der der Designer komponentenspezifische Ressourcen (beispielsweise das Symbol einer Schaltfläche) abspeichert
Ausgabedateien	
.exe	Anwendung
.dll	Bibliothek
.pdb	Debugger-Informationen

Tabelle 2.2 Die wichtigsten Projektdateien

Dirk Louis, Shinja Strasser, Thorsten Kansy: Microsoft Visual C# - Das Entwicklerbuch, Microsoft Press 2010 (ISBN 978-3-86645-529-0)

Startprojekte, Verweise und Projektabhängigkeiten

Richtig interessant wird die Arbeit mit Projektmappen erst dann, wenn in einer Projektmappe mehrere Projekte verwaltet werden. Dies soll nicht heißen, dass es erstrebenswert wäre, immer mehrere Projekte in einer Projektmappe zusammenzufassen. Im Gegenteil: Lieber eine glückliche 1:1-Beziehung als ein Konglomerat aus Projekten, die nicht zusammenpassen. Wenn es sich aber ergibt, dass mehrere Projekte – beispielsweise eine Anwendung und eine zugehörige Bibliothek – sinnvoll in einer Projektmappe vereint werden, so bedingt dies nicht selten weitere Entscheidungen.

Who's first?

Vielleicht möchten Sie festlegen, welches der Projekte beim Erstellen der kompletten Projektmappe als Erstes kompiliert und ausgeführt werden soll? Dazu müssen Sie das Projekt einfach nur als Startprojekt deklarieren. Den entsprechenden Menübefehl *Als Startprojekt festlegen* finden Sie im Kontextmenü des Projektknotens.

HINWEIS Das Startprojekt wird im Projektmappen-Explorer durch Fettschrift hervorgehoben.

Vielleicht möchten Sie aber auch einzelne Projekte ganz von der Erstellung ausschließen. Dann empfiehlt sich der Aufruf des Konfigurations-Managers über den gleichnamigen Befehl im Kontextmenü des Projektmappenknotens.

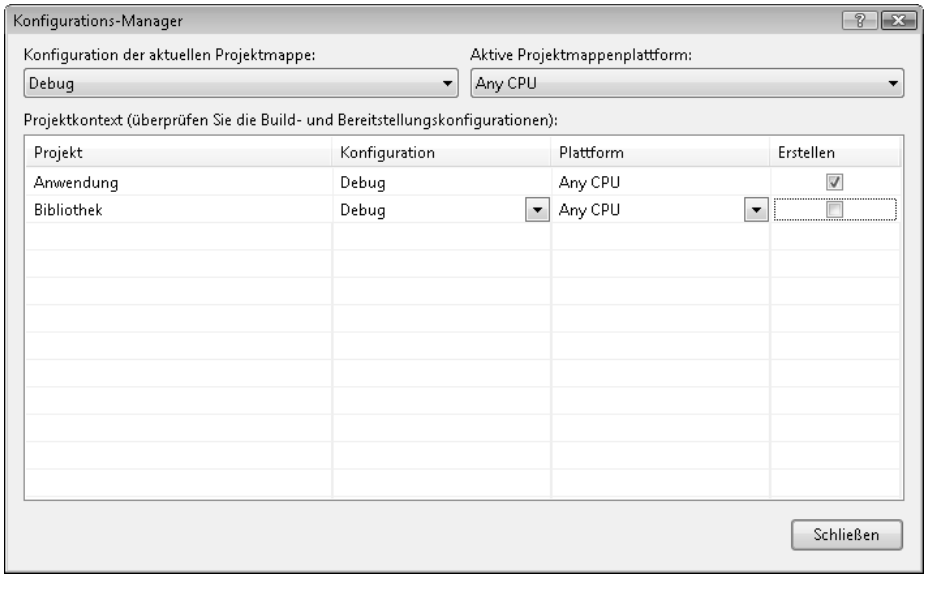

Abbildung 2.9 Im Konfigurations-Manager können Sie bequem festlegen, welche Projekte der Projektmappe erstellt werden bzw. von der Erstellung auszuschließen sind

Verweise

Verweise sind nicht nur für Projekte wichtig, die zu mehreren in einer Projektmappe zusammengefasst sind. Grundsätzlich gilt: Damit Sie in einem Projekt Klassen und ähnliche Elemente aus anderen Assemblys (seien dies nun Assemblys zu Projekten der gleichen Projektmappe, anderer Projektmappen oder Assemblys von Drittanbietern) verwenden können, müssen Sie in Ihrem Projekt einen Verweis auf die Assembly einrichten.

Angenommen, Ihre Projektmappe enthält ein Projekt *Bibliothek* und ein Projekt *Anwendung*, in dem Sie eine Klasse aus *Bibliothek* verwenden wollen. Dazu ist es erforderlich, dass Sie in *Anwendung* einen Verweis auf *Bibliothek* einrichten:

1. Klicken Sie mit der rechten Maustaste auf den *Verweise*-Knoten des Projekts, dem Sie einen Verweis hinzufügen wollen (hier *Anwendung*).
2. Führen Sie den Befehl *Verweis hinzufügen* aus.
3. Wechseln Sie zur Registerkarte *Projekte* und wählen Sie das Projekt aus der Projektmappe aus, auf das verwiesen werden soll (hier *Bibliothek*).

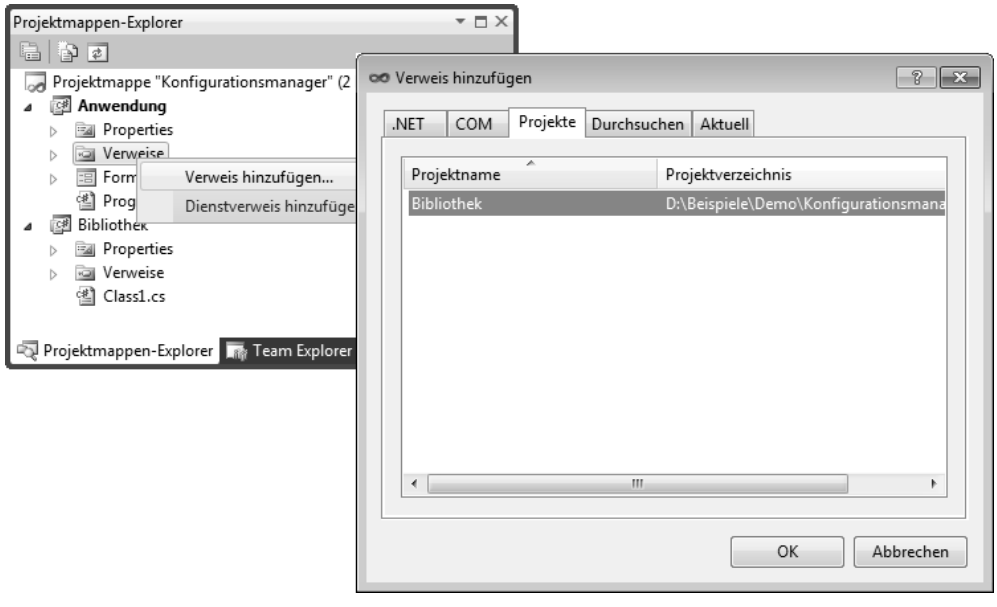

Abbildung 2.10 Verweise einrichten

HINWEIS Wie Sie dem Dialogfeld in Abbildung 2.10 entnehmen können, ist es auch möglich, Verweise auf Assemblys von .NET Framework oder auf Assemblys an beliebigen Speicherorten einzurichten.

Abhängigkeiten

Sobald es Verweise zwischen den Projekten einer Projektmappe gibt, wird die Erstellungsreihenfolge der Projekte wichtig. Diese können Sie im Dialogfeld *Projektabhängigkeiten* kontrollieren und ändern. Sie können dieses Dialogfeld über den gleichnamigen Eintrag im Kontextmenü des Projektknotens aufrufen. Allerdings ist der Eintrag nur vorhanden, wenn in der Projektmappe mindestens zwei Projekte enthalten sind.

Abbildung 2.11 Erstellungsreihenfolge von Projekten

Die Erstellungsreihenfolge berechnet Visual Studio automatisch aus den Abhängigkeiten zwischen den Projekten. Sie können die Abhängigkeiten auf der gleichnamigen Registerkarte angeben. Im Falle unserer Bibliothek/Anwendung-Abhängigkeit aus dem vorangehenden Abschnitt ist dies jedoch nicht nötig. Visual Studio hat die Abhängigkeit, die sich durch den Verweis ergibt, selbstständig erkannt und eingetragen.

Eigene Projektvorlagen

Wem die von Visual Studio mitgelieferten Projektvorlagen auf die Dauer nicht genügen, der kann ohne viel Aufwand eigene Projektvorlagen erstellen.

1. Legen Sie ein neues Projekt an und bearbeiten Sie es so weit, bis es den Stand erreicht hat, den Sie als neue Projektvorlage speichern wollen.
2. Rufen Sie den Menübefehl *Datei/Vorlage exportieren* auf.
3. Folgen Sie den Anweisungen des Assistenten.

Die neue Vorlage ist fortan unter dem Projekttyp *Visual C#* im Dialogfeld *Neues Projekt* verfügbar.

HINWEIS Um eine selbst erstellte Projektvorlage wieder zu entfernen, löschen Sie die zugehörigen ZIP-Dateien aus den Verzeichnissen *Visual Studio 2010\My Exported Templates* und *Visual Studio 2010\ Templates\Project Templates* unter Ihrem Dokument-Ordner (*Eigene Dateien* für Windows XP und *Dokumente* für Windows Vista).

Der Code-Editor

Der Code-Editor von Visual Studio bietet Syntaxhervorhebung, automatische Einrückung, Formatierung, Codegliederung, Zeilennummerierung, Aufspüren von Klammernpaaren, Zusammenarbeit mit dem integrierten Debugger sowie

- Anweisungsvervollständigung und Kontexthilfe (IntelliSense)
- `using`-Ergänzung und Umschließung
- Codeausschnitte
- Umgestaltung

Kleine Annehmlichkeiten

Sie wundern sich, warum die versprochenen Zeilennummern nicht zu sehen sind? Dann rufen Sie den Befehl *Extras/Optionen* auf, öffnen Sie die Seite *Text-Editor/Alle Sprachen* und aktivieren Sie die Option *Zeilennummern*.

Sie sind nicht zufrieden mit der Art und Weise, wie die verschiedenen syntaktischen Codeelemente im Code-Editor hervorgehoben werden? Dann rufen Sie den Befehl *Extras/Optionen* auf und öffnen Sie die Seite *Umgebung/Schriftarten und Farben*. Achten Sie darauf, dass im Dropdown-Listenfeld *Einstellungen anzeigen für* der *Text-Editor* ausgewählt ist und verändern Sie die Darstellung der Sie interessierenden Elemente. Sollten Ihnen die vorgenommenen Änderungen später doch nicht zusagen, drücken Sie auf die Schaltfläche *Standard verwenden*.

Sie sind nicht zufrieden mit der Art und Weise, wie der Code-Editor Leerzeichen und Einzüge setzt? Dann rufen Sie den Befehl *Extras/Optionen* auf und bearbeiten Sie die Optionen auf den Seiten *Abstand* und *Einzug* unter *Text-Editor/C#/Formatierung*.

Sie würden gerne einen größeren Codeabschnitt im Editorfenster einsehen können, oder umgekehrt die Anzeige vergrößern, um den Code besser lesen zu können? Dann halten Sie die [Strg]-Taste gedrückt und drehen Sie das Mausrad nach vorne (vergrößern) oder zurück (verkleinern). Oder wählen Sie den gewünschten Zoomfaktor direkt in dem Dropdown-Listenfeld am linken unteren Rand des Editorfensters aus.

Äußerst hilfreich ist auch die Codegliederung, die es Ihnen erlaubt, die Definitionen von Funktionen und Klassen mit einem Klick ein- und auszublenden. Nutzen Sie diese Option, um abgeschlossene Elemente auszublenden und konzentrieren Sie sich ganz auf die Elemente, an denen Sie gerade arbeiten. Übrigens: Sie können nicht nur Funktionen und Klassen ein- und ausblenden, sondern grundsätzlich beliebige Codeabschnitte. Markieren Sie den auszublendenden Abschnitt einfach mit der Maus und rufen Sie im Kontextmenü den Befehl *Gliederung/Aktuelles Element umschalten* auf.

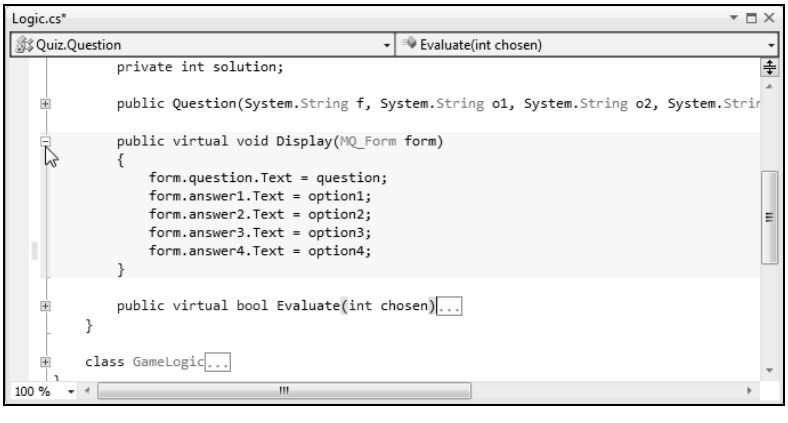

Abbildung 2.12 Ein Klick auf das Kästchen lässt das Element kollabieren

Seit längerer Zeit ist es schon möglich, im Code-Editor durch Drücken von Alt und Ziehen mit der Maus einen beliebigen rechteckigen Textbereich auszuwählen, der dann kopiert oder gelöscht werden konnte. Seit der Version 2010 ist es nun auch möglich, in den ausgewählten Bereich einzufügen. Beispielsweise können Sie auf diese Weise die Zugriffsmodifizierer mehrerer untereinander definierter Felder markieren und austauschen. Lassen Sie nach dem Markieren Maus- und Alt-Taste einfach los und geben Sie den neuen Modifizierer ein.

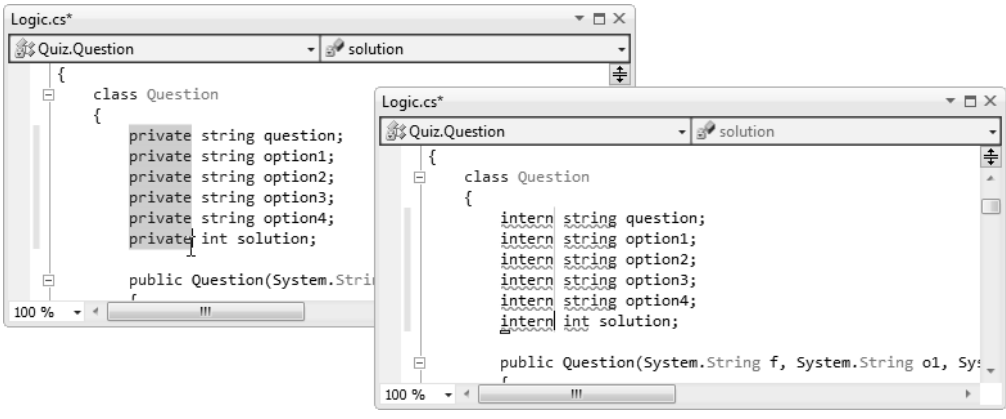

Abbildung 2.13 Ändern mit der Feldauswahl

TIPP Wenn Sie ein ganz schmales Feld markieren, erhalten Sie quasi eine Einfügemarke für mehrere Zeilen.

IntelliSense

IntelliSense ist eine Technologie, die während der Arbeit im Editor jeden Ihrer Schritte verfolgt und wo immer möglich, eine Liste mit Vorschlägen zur Codevervollständigung einblendet.

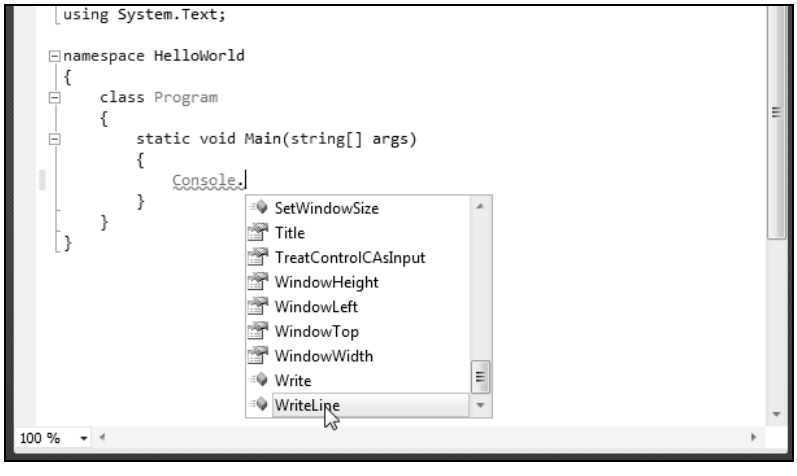

Abbildung 2.14 IntelliSense mit Member-Liste

- Wenn Sie ein neues Wort beginnen, blendet IntelliSense je nach Kontext eine Liste gleich beginnender Schlüsselwörter und/oder Typen ein
- Wenn Sie über den Punktoperator auf ein Member eines Objekts oder einer Klasse zugreifen, öffnet sich ein Listenfeld, in dem die verschiedenen Member der Klasse aufgeführt werden
- Wenn Sie weitertippen, wird das Listenfeld zu dem Eintrag gescrollt, der Ihrer bisherigen Buchstabenfolge am besten entspricht. Durch Drücken der ⇥-Taste (aber auch durch Drücken der ↵-Taste oder der Eingabe bestimmter Signalzeichen wie Leerzeichen, öffnende Klammer oder Punktoperator) können Sie das aktuell ausgewählte Listenelement in den Quelltext einfügen lassen, wobei etwaige Tippfehler in Ihrer Buchstabenfolge korrigiert werden.
- Alternativ können Sie den gewünschten Member auch mit der Maus oder den Pfeiltasten auswählen
- Wenn Sie nach einem Methodennamen eine öffnende Klammer eingeben, öffnet sich ein Listenfeld, in dem Ihnen die zu der Methode gehörenden Parameter angezeigt werden – eine Option, die Ihnen ab und an das Nachschauen in der Online-Hilfe ersparen kann.

  Die Parameteranzeige unterstützt auch überladene Methoden, die Sie mithilfe der Pfeiltasten auswählen können.

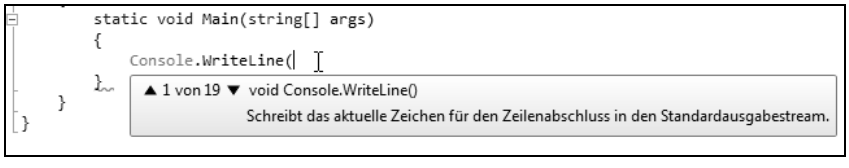

Abbildung 2.15 IntelliSense mit Parameterinformationen

ACHTUNG IntelliSense funktioniert nur für Quelldateien, die Teil eines Projekts sind! Einer einzelnen mit Visual Studio geöffneten Quellcodedatei steht IntelliSense nicht zur Verfügung.

QuickInfo

Wenn Sie in der Member-Liste einen Eintrag auswählen und ein paar Sekunden warten, wird neben dem Eintrag ein kurzer Hilfetext, das so genannte QuickInfo, eingeblendet.

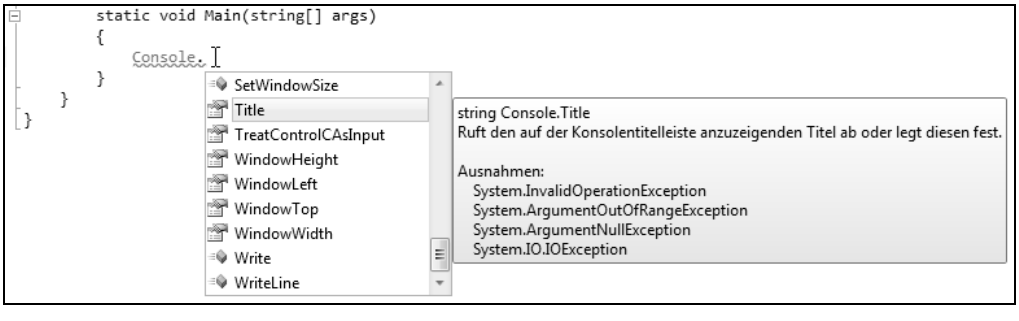

Abbildung 2.16 QuickInfo

Moduswechsel, Deaktivierung und expliziter Aufruf

IntelliSense gehört sicherlich zur Kategorie derjenigen dienstbaren Geister, auf die man nach kurzer Eingewöhnung nicht mehr verzichten kann. Der eine oder andere Programmierer mag in IntelliSense aber auch nur einen aufdringlichen Quälgeist sehen. Für diesen Fall gibt es zwei Möglichkeiten.

Wenn Sie IntelliSense an sich schätzen, Sie sich aber darüber ärgern, dass es Ihnen häufig Ihre eingetippten Variablennamen durch IntelliSense-Vorschläge ersetzt, rufen Sie den Befehl *Bearbeiten/IntelliSense/Beendigungsmodus umschalten* auf. IntelliSense ersetzt Ihre Vorgaben dann nur noch bei Drücken der ⇥-Taste. Tippen Sie dagegen zum Abschluss ein Leerzeichen, eine schließende Klammer oder ein ähnliches Zeichen, nimmt IntelliSense keine Ersetzung vor.

Wenn Sie IntelliSense gar nicht schätzen, schalten Sie es aus:

1. Rufen Sie den Menübefehl *Extras/Optionen* auf.
2. Wechseln Sie zur Seite *Text-Editor/Alle Sprachen/Allgemein*.
3. Deaktivieren Sie die Kontrollkästchen *Member automatisch auflisten* und *Parameterinformationen*.

Selbstverständlich können Sie auch nur die Memberauflistung, nur die Parameterinformationen oder nur die Wortvervollständigung (Option *Vervollständigungsliste nach Eingabe eines Zeichens anzeigen* unter *Text-Editor/C#/IntelliSense*) deaktivieren.

Deaktiviert wird allerdings nur die im Hintergrund laufende Codeüberwachung; die Vervollständigungslisten sind weiter vorhanden und können auf Wunsch jederzeit eingeblendet werden – nur eben, dass Sie selbst, und nicht IntelliSense, entscheiden, wann die Listen angezeigt werden.

- Die Member- und die Wortliste rufen Sie mit der Tastenkombination Strg+J auf
- Die Parameterliste rufen Sie mit der Tastenkombination Strg+⇧+␣ auf
- Die QuickInfo rufen Sie mit der Tastenkombination Strg+K+I auf
- Alle Listen und die QuickInfo können alternativ mit den Menübefehlen unter *Bearbeiten/IntelliSense* aufgerufen werden

using-Smarttag

Um eine Klasse oder einen sonstigen Typ aus einer anderen Assembly verwenden zu können, müssen Sie den voll qualifizierten Klassennamen angeben (inklusive Namespace) oder eine `using`-Anweisung für den Namespace einfügen. Falls Sie nicht sicher sind, aus welchem Namespace die Klasse stammt oder wegen der Schreibweise im Zweifel sind, lassen Sie sich am Besten von dem `using`-Smarttag helfen. Dieses wird als kleines (!) rotes Rechteck unterhalb des ungebundenen Bezeichners eingeblendet. Wenn Sie die Maus über das Rechteck bewegen, erscheint das Symbol eines Menüs mit Befehlen zur Erweiterung des Bezeichners oder der `using`-Anweisungen.

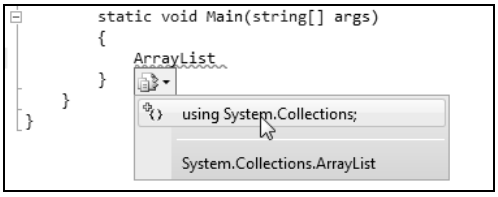

Abbildung 2.17 *using*-Smarttag

ACHTUNG Elemente anderer Assemblys sind nur dann verfügbar, wenn das Projekt Verweise auf die betreffenden Assemblys enthält (siehe den Abschnitt »Projektmappen« in diesem Kapitel). Dies ist gleichzeitig die Informationsquelle für das `using`-Smarttag.

Umschließung

Ein ebenfalls nettes kleines Feature ist der Umschließungsbefehl, mit dem Anweisungsblöcke angelegt oder bestehender Code nachträglich in Anweisungsblöcke zusammengefasst werden kann:

1. Markieren Sie die Anweisungen, die Sie zu einem Block zusammenfassen möchten.

   Soll ein leeres Blockgerüst angelegt werden, markieren Sie nichts, sondern platzieren Sie einfach die Einfügemarke.
2. Rufen Sie im Kontextmenü des Editors den Befehl *Umschließen mit* auf.
3. Wählen Sie in der daraufhin geöffneten Liste den gewünschten Block aus.

TIPP Die Einrückung des Blocks können Sie gegebenenfalls nachträglich mit den Symbolleistenschaltflächen *Einzug vergrößern* respektive *Einzug verkleinern* anpassen.

Codeausschnitte

Codeausschnitte sind kurze benannte Codefragmente, wie sie bei der Programmierung häufig benötigt werden, etwa `for`-Schleifen, `if`-Blöcke, `WriteLine()`-Aufrufe etc.

Ausschnitt	Beschreibung	Expandierter Code

class	Klassendeklaration	`class MyClass`
		`{`
		
		`}`

ctor	Konstruktor	`public ClassNamePlaceholder ()`
		`{`
		
		`}`
		Wenn Sie den Ausschnitt innerhalb einer Klassendeklaration einfügen, wird `ClassNamePlaceholder` durch den Namen der Klasse ersetzt.

cw	Konsolenausgabe	`Console.WriteLine();` ▶

Ausschnitt	Beschreibung	Expandierter Code

equals	Überschreibung von `Object.Equals()`	`// override object.Equals`
		`public override bool Equals (object obj)`
		`{`
		`  //`
		`  // See the full list of guidelines at ...`
		`  //`
		
		`  if (obj == null || GetType() != obj.GetType())`
		`  {`
		`  return false;`
		`  }`
		
		`  // TODO: write your impl. of Equals() here.`
		`  throw new NotImplementedException();`
		`  return base.Equals (obj);`
		`}`
		
		`// override object.GetHashCode`
		`public override int GetHashCode()`
		`{`
		`  // TODO: write your impl. of GetHashCode() here`
		`  throw new NotImplementedException();`
		`  return base.GetHashCode();`
		`}`

for	for-Schleife, inkrementierend	`for (int i = 0; i < length; i++)`
		`{`
		
		`}`
		Nach dem Einfügen ist die Schleifenvariable markiert. Wenn Sie einen anderen Bezeichner eingeben, werden die nachfolgenden Vorkommen automatisch ersetzt.

forr	for-Schleife, dekrementierend	`for (int i = length - 1; i >= 0 ; i--)`
		`{`
		`}`
		Nach dem Einfügen ist die Schleifenvariable markiert. Wenn Sie einen anderen Bezeichner eingeben, werden die nachfolgenden Vorkommen automatisch ersetzt.

if	if-Bedingung	`if (true)`
		`{`
		
		`}`

try	try...catch-Block	`try`
		`{`
		
		`}`
		`catch (Exception)`
		`{`
		`  throw;`
		`}`

unsafe	Block für unsicheren Code	`unsafe`
		`{`
		
		`}`

Tabelle 2.3 Eine Auswahl vordefinierter Codeausschnitte

Um einen Codeausschnitt einzufügen, setzen Sie die Einfügemarke an die Stelle, wo der Codeausschnitt beginnen soll, rufen im Kontextmenü des Code-Editors den Befehl *Ausschnitt einfügen* auf und wählen in der angezeigten Liste den gewünschten Codeausschnitt aus.

Handelt es sich bei dem Codeausschnitt um einen Block, mit dem eine bestehende Anweisungsfolge nachträglich eingefasst werden soll, rufen Sie im Kontextmenü den Befehl *Umschließen mit* auf.

TIPP Hinter den Codeausschnitten stehen XML-Dateien, die nach dem CodeSnippet-Schema von Microsoft aufgebaut sind. Entwickler sind angehalten, eigene Codeausschnitte zu definieren und auszutauschen. Wenn Sie selbst weitere Codeausschnitte aus dem Internet herunterladen oder importieren möchten, verwenden Sie dazu den Codeausschnitt-Manager (Aufruf über das *Extras*-Menü).

Sollten Sie gar interessiert sein, eigene Codeausschnitte zu generieren, müssen Sie eine schemakonforme XML-Datei mit der Extension *.snippet* anlegen, die Ihren Codeausschnitt beschreibt. Für eine Beschreibung dieses Formats reicht der Platz hier leider nicht aus, wenn Sie aber über etwas Erfahrung mit XML verfügen und sich als Ausgangspunkt eine nicht zu komplizierte bestehende Snippet-Datei kopieren, die Sie nur anzupassen brauchen (beispielsweise *cw.snippet* oder *class.snippet*), werden Ihre Bemühungen sicherlich von Erfolg gekrönt sein. Ihre fertige Snippet-Datei müssen Sie dann nur noch im Verzeichnis für benutzerdefinierte Codeausschnitte speichern (den genauen Pfad zu diesem Verzeichnis können Sie im Codeausschnitt-Manager nachschlagen).

Umgestaltung

Eine ebenfalls sehr nützliche Technik ist das Refactoring, zu Deutsch *Umgestaltung*. Dahinter verbirgt sich eine ausgefeilte Suchen-und-Ersetzen-Technologie, die Ihnen bei der Überarbeitung, Wartung und Umstrukturierung Ihres Codes hilft.

Umgestalten-Befehl	Beschreibung
Umbenennen	Ändert alle Vorkommen eines Bezeichners (Name einer Klasse, Methode, Namespaces etc.) innerhalb einer Anwendung.
Methode extrahieren	Wandelt Codeanweisungen in eine Methode um. Markieren Sie den umzuwandelnden Codeblock. Beim Aufruf des Befehls wird in der aktuellen Klasse eine neue Methode definiert, mit den markierten Anweisungen als Implementierung. Der ursprünglich markierte Code wird durch einen Methodenaufruf ersetzt.
Feld kapseln	Erzeugt aus einem Feld (also einer Klassen-Member-Variablen) eine Eigenschaft. Setzen Sie die Einfügemarke auf das umzuwandelnde Feld. Beim Aufruf des Befehls wird in der aktuellen Klasse eine Eigenschaft mit `get` und `set` zu dem Feld definiert. Zugriffe auf das Feld werden durch Zugriffe über die Eigenschaft ersetzt. (Ob dies nur außerhalb der Klasse oder auch innerhalb der Klasse geschieht, können Sie selbst in einem Dialogfeld festlegen.)
Schnittstelle extrahieren	Erzeugt eine Schnittstellendeklaration aus einer Klassendeklaration. Setzen Sie die Einfügemarke auf den Namen der Klassenvorlage. Beim Aufruf des Befehls erscheint ein Dialogfeld, in dem die `public`-Eigenschaften und -Methoden der Klasse aufgeführt werden. Sie können auswählen, welche dieser Member in die Schnittstellendeklaration übernommen werden sollen. Die Schnittstelle wird in einer eigenen Quelldatei definiert, die ursprüngliche Klasse implementiert die Schnittstelle.
Lokale Variable auf Parameter hochstufen	Verwandelt eine lokale Variable in einen Parameter. Handelt es sich um eine Methode, die eine geerbte Methode überschreibt oder eine Schnittstellenmethode implementiert, wird eine Warnung ausgegeben. ▶

Umgestalten-Befehl	Beschreibung
Parameter entfernen	Entfernt einen Parameter aus einer Methodendeklaration. Die Methodenaufrufe werden ebenfalls angepasst, der Parameter wird allerdings nicht aus dem Methodenrumpf gestrichen.
Parameter neu anordnen	Ändert die Reihenfolge der Parameter in einer Methodendeklaration. Die Methodenaufrufe werden ebenfalls angepasst.

Tabelle 2.4 Befehle im Menü *Umgestalten*

Um zum Beispiel den Namen eines Felds global in einer Anwendung zu ändern, gehen Sie wie folgt vor:

1. Setzen Sie die Einfügemarke auf den Feldbezeichner oder markieren Sie ihn.
   Sie müssen die Einfügemarke nicht notwendigerweise auf das erste Vorkommen des Bezeichners setzen.
2. Rufen Sie den Befehl *Umbenennen* im Menü *Umgestalten* aus der Hauptmenüleiste oder dem Kontextmenü des Editors auf.

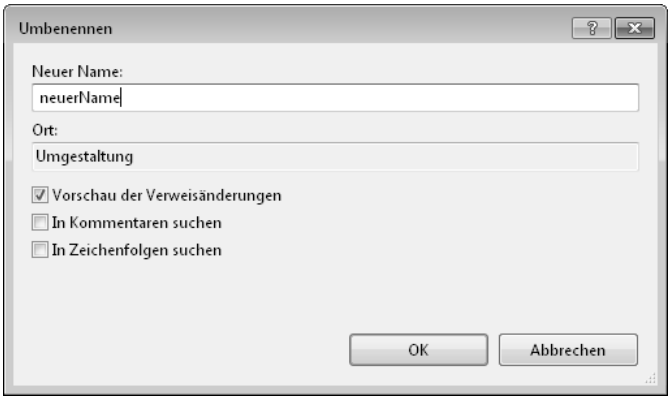

Abbildung 2.18 Umbenennen eines Felds

3. Im Dialogfeld *Umbenennen* geben Sie den neuen Feldnamen an.

   Achten Sie darauf, dass das Kontrollkästchen *Vorschau der Verweisänderungen* aktiviert ist. Ansonsten führt der Klick auf die *OK*-Schaltfläche bereits unwiderruflich zur Änderung Ihres Codes. Sie können die Änderungen zwar notfalls rückgängig machen (Strg+Z), aber es ist weit schwerer festzustellen, ob die Änderungen korrekt oder falsch sind.

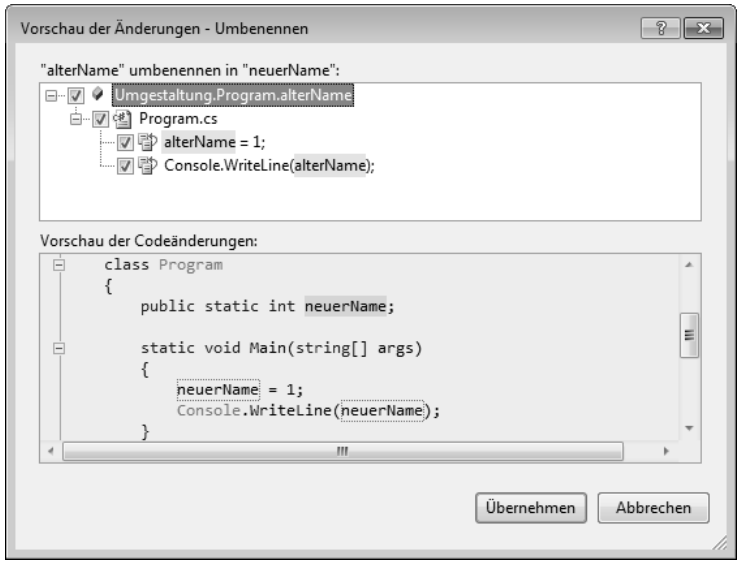

Abbildung 2.19 Letzte Kontrolle vor der Umbenennung

4. Prüfen Sie im Vorschau-Dialogfeld die geplanten Änderungen.

   Sie können den unteren Anzeigebereich durchblättern oder im oberen Teil die einzelnen Vorkommen anklicken.

5. Sind die Änderungen in Ihrem Sinne, klicken Sie auf *Übernehmen*.

Erstellung

Die Befehle zum Erstellen sind im gleichnamigen Menü der IDE zusammengefasst. Sie können wahlweise das aktuelle Projekt oder alle Projekte in der Projektmappe erstellen und Sie können das Projekt oder die Projekte

- erstellen (Kompilation noch nicht kompilierter oder überarbeiteter Module)
- neu erstellen (Kompilation aller Module)
- bereinigen (löscht alle Zwischenprodukte vorangehender Erstellungen)

MSBuild

Für die Erstellung der Projekte ist MSBuild verantwortlich. MSBuild ist ein eigenständiges Programm, das die Informationen, wie die Projekte zu erstellen sind, den XML-Projektdateien entnimmt. Obwohl es möglich ist, auf die Buildvorgänge durch direkte Bearbeitung der XML-Dateien einzuwirken, ist dies, solange Sie mit Visual C# arbeiten, in der Regel nicht nötig. Die Steuerung des Buildvorgangs erfolgt hier über die Projekteigenschaften.

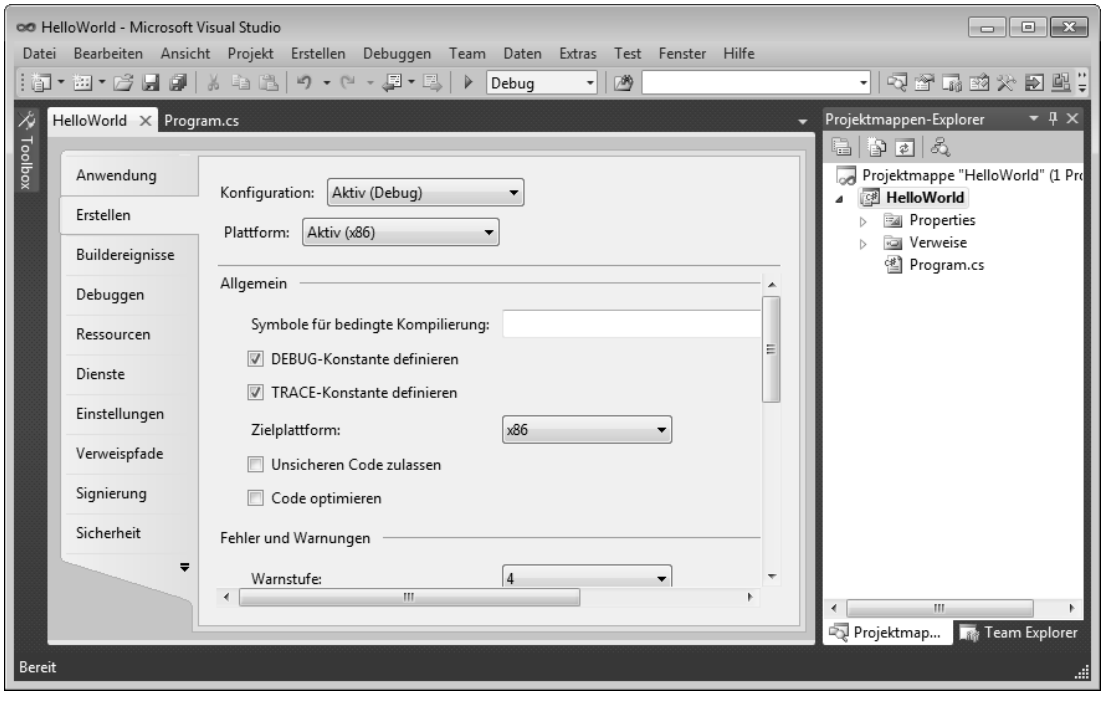

Abbildung 2.20 Projekteigenschaften

Zum Bearbeiten der Projekteigenschaften klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf den Knoten des Projekts und rufen im Kontextmenü den Befehl *Eigenschaften* auf.

Hier können Sie beispielsweise festlegen, ob der erzeugte Code optimiert, die Kompilierung bei Warnungen abgebrochen und welche Befehlszeilenargumente beim Debuggen übergeben werden sollen.

Build-Konfigurationen

Die Anpassungsmöglichkeiten, die die Projekteigenschaften bieten, sind großartig, doch wenn man die Einstellungen laufend wechseln müsste, um die Projekterstellung an die augenblicklichen Bedürfnisse anzupassen, wäre dies äußerst lästig. Visual Studio erlaubt es daher, komplette Sätze von Projekteinstellungen als *Konfigurationen* abzuspeichern und stellt gleich einige Konfigurationen zur Verfügung. Welche genau, hängt von der jeweiligen Visual Studio-Version ab.

Konfiguration	Beschreibung
Debug	In dieser Konfiguration ist die Option für die *DEBUG*-Konstante gesetzt, Optimierungen sind deaktiviert, das Ausgabeverzeichnis ist das Unterverzeichnis *\bin\Debug*.
Release	In dieser Konfiguration ist die Option für die *DEBUG*-Konstante deaktiviert, Optimierungen sind eingeschaltet, das Ausgabeverzeichnis ist das Unterverzeichnis *\bin\Release*. Diese Konfiguration ist für die abschließende Erstellung des fertigen Programms gedacht.

Tabelle 2.5 Vordefinierte Build-Konfigurationen (der Professional-Edition)

Für die Arbeit mit Konfigurationen müssen Sie Folgendes wissen:

- Wie Sie Konfigurationen anpassen ...

  Rufen Sie im Kontextmenü des Projektknotens den Befehl *Eigenschaften* auf und wählen Sie auf der Seite *Erstellen* im Listenfeld *Konfiguration* die zu bearbeitende Konfiguration aus. Überarbeiten Sie die verschiedenen Einstellungen auf den Seiten *Erstellen* und *Debuggen* nach Ihren Vorstellungen. Speichern Sie Ihre Einstellungen.

- Wie Sie Konfigurationen neu anlegen ...

  Rufen Sie den Konfigurations-Manager auf (Menübefehl *Erstellen/Konfigurations-Manager*). Wählen Sie im Listenfeld *Konfiguration der aktuellen Projektmappe* die Option *<Neu...>* aus. Im daraufhin geöffneten Dialogfeld können Sie anschließend einen Namen für die neu anzulegende Konfiguration angeben.

- Wie Sie Konfigurationen auswählen ...

  Rufen Sie den Konfigurations-Manager auf (Menübefehl *Erstellen/Konfigurations-Manager*). Wählen Sie in der Spalte *Konfiguration* die gewünschte Konfiguration für Ihr Projekt aus.

Konfiguration

Die IDE und ihre integrierten Programmierwerkzeuge können in vielfältiger Weise konfiguriert und angepasst werden.

Umgebungseinstellungen

Als Sie Visual Studio das erste Mal gestartet haben, wurden Sie aufgefordert, eine der vordefinierten Umgebungseinstellungen auszuwählen. Wenn Sie einige Zeit mit Visual Studio gearbeitet haben, werden Sie vielleicht feststellen, dass Sie lieber eine andere der vordefinierten Umgebungseinstellungen ausprobieren möchten. Oder Sie haben – ausgehend von der gewählten Umgebungseinstellung – die IDE mittlerweile weiter an Ihre Bedürfnisse angepasst und würden die aktuelle IDE-Gestaltung gerne als benutzerdefinierte Umgebungseinstellung speichern. Oder Sie möchten einfach die ursprüngliche Umgebungseinstellung wiederherstellen.

Alles ist möglich. Sie müssen nur den Menübefehl *Extras/Einstellungen importieren und exportieren* aufrufen und den Anweisungen des Assistenten folgen.

Extras/Optionen

Über den Menübefehl *Extras/Optionen* rufen Sie das mehrseitige Dialogfeld *Optionen* auf, in dem das Gros der Einstellungsmöglichkeiten zusammengefasst ist. Die einzelnen Optionen sind nach Seiten geordnet und in der Hilfe (Klick auf das Fragezeichen) dokumentiert und erläutert. Wir begnügen uns hier mit einer kleinen Auswahl.

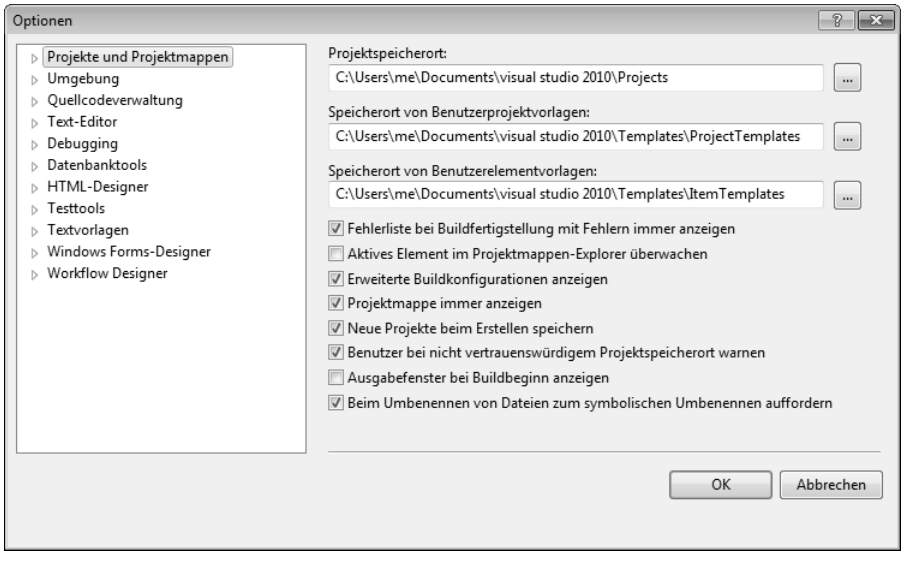

Abbildung 2.21 Die Optionen zur IDE-Konfiguration

Mit letztem Projekt statt mit Startseite starten

Nichts gegen die Startseite, aber wäre es nicht effizienter, wenn die IDE gleich mit der zuletzt bearbeiteten Projektmappe starten würde?

Was beim Start geschehen soll, legen Sie über die Optionen unter *Umgebung/Start* fest.

Anzahl der zuletzt geöffneten Projekte

Standardmäßig speichert Visual Studio in seinen History-Listen die jeweils letzten zehn Dateien und Projekte. Wenn Sie über die *Zuletzt geöffnete...*-Befehle im *Datei*-Menü Zugriff auf mehr (oder weniger) Dateien oder Projekte haben möchten, erhöhen (bzw. verringern) Sie unter *Umgebung/Allgemein* die Einstellung für die Option *Elemente in »Zuletzt geöffnet«-Listen angezeigt*.

Tastenkombinationen

Unter *Umgebung/Tastatur* können Sie auswählen, mit welchem Tastaturzuordnungsschema Sie arbeiten möchten – beispielsweise *Visual C# 2005*, *Visual Basic 6* oder auch *Emacs*.

Sie können das gewählte Tastaturzuordnungsschema auch individuell abwandeln, indem Sie zusätzliche Tastenkombinationen definieren und bestehende Tastenkombinationen löschen oder ersetzen. Die Früchte Ihrer Arbeit können Sie jederzeit als benutzerdefiniertes Tastaturzuordnungsschema speichern:

1. Rufen Sie den Menübefehl *Extras/Einstellungen importieren und exportieren* auf.
2. Wählen Sie auf der ersten Seite des Assistenten die Option *Ausgewählte Umgebungseinstellungen exportieren* und klicken Sie auf *Weiter*.
3. Deaktivieren Sie auf der zweiten Seite alle Kästchen. Expandieren Sie anschließend die Knoten *Optionen/Umgebung* und wählen Sie allein die Kategorie *Umgebung/Tastatur* aus. Klicken Sie danach auf *Weiter*.
4. Exportieren Sie Ihre Einstellungen mit einem Klick auf *Fertig stellen*.

Einzüge im Editor

Sie sind nicht zufrieden mit der Art und Weise wie der Code-Editor Leerzeichen und Einzüge setzt? Dann bearbeiten Sie die Seiten *Abstand* und *Einzug* unter *Text-Editor/C#/Formatierung*.

Fehlerunterstreichung deaktivieren

Nerven Sie die Schlangenlinien, mit denen der Text-Editor mögliche Fehler unterstreicht? Wenn ja, dann deaktivieren Sie auf der Seite *Text-Editor/C#/Erweitert* die Optionen *Fehler im Editor unterstreichen* und *Livesemantikfehler anzeigen*.

Externe Tools einrichten

Neben der Ausführung der vielen integrierten Dienstprogramme bietet Ihnen die IDE auch die Möglichkeit, beliebige externe DOS- oder Windows-Programme direkt von der IDE aus aufzurufen.

Um ein externes Dienstprogramm für den Aufruf aus der IDE einzurichten, gehen Sie wie folgt vor:

1. Rufen Sie den Befehl *Extras/Externe Tools* auf und klicken Sie auf die Schaltfläche *Hinzufügen*.
2. Geben Sie einen *Titel* für das Dienstprogramm ein. Der eingegebene Name wird später als Menübefehl zum Aufruf des Programms im Menü *Extras* eingeblendet.
3. Geben Sie optional vor dem Buchstaben für den Alt-Aufruf ein kaufmännisches Und-Zeichen (&) ein.
4. Wählen Sie die *.exe*-Datei des Dienstprogramms an. Klicken Sie auf die Schaltfläche mit den drei Punkten neben dem Eingabefeld *Befehl*, um nach der *.exe*-Datei suchen zu können.
5. Geben Sie unter *Argumente* gegebenenfalls gewünschte oder notwendige Befehlszeilenargumente für den Aufruf des Dienstprogramms an. Über die Schaltfläche mit dem schwarzen Dreieck neben dem Eingabefeld können Sie eine Liste mit verschiedenen nützlichen Makros auswählen und in die Befehlszeile aufnehmen. Diese Makros werden dann bei Aufruf des Dienstprogramms von der IDE ausgewertet.
6. Geben Sie unter *Ausgangsverzeichnis* gegebenenfalls an, wohin etwaige Ausgaben umgeleitet werden sollen.

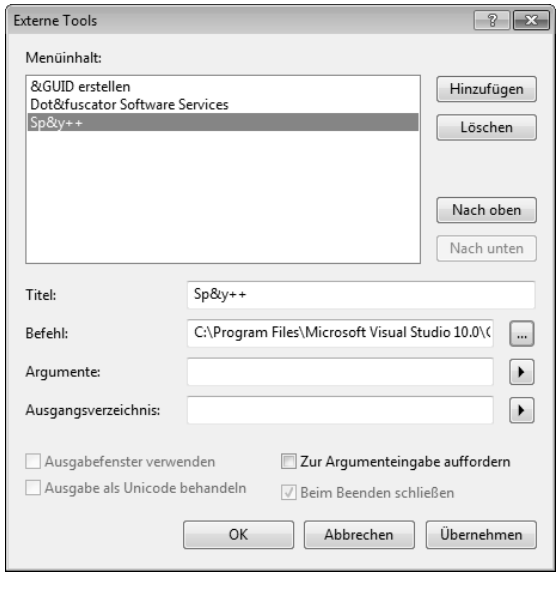

Abbildung 2.22 Einrichtung von Spy++ als externes Tool in der Visual Studio-IDE

Fensterverwaltung

Eine der größten Herausforderung bei der Arbeit mit Visual Studio ist die Verwaltung der vielen Fenster. Drei Technologien, die in den folgenden Abschnitten beschrieben sind, sollen Ihnen dabei helfen.

Die Andock-Hilfe

Das Andocken der Toolfenster in den IDE-Rahmen wurde bereits in der letzten Visual Studio-Version wesentlich verbessert. Während das Andocken eines Fensters früher den Charakter eines Trial-and-Error-Experiments hatte, mit dem Ziel, durch wildes Verschieben des Fensters die gewünschte Andockposition zu finden, blendet Visual Studio mittlerweile, sobald Sie ein Fenster aufnehmen, zwei Kränze von Andockpositionen ein:

- einen inneren Kranz mit Andockpositionen relativ zu dem Fenster, über dem sich gerade Ihre Maus befindet
- einen äußeren Kranz zum Andocken in den IDE-Rahmen (nur für Toolfenster), wobei stets die gesamte Breite bzw. Länge des Rahmens eingenommen wird

Alles, was Sie tun müssen ist, die Maus mit dem aufgenommenen Fenster über die gewünschte Andockposition zu ziehen, kontrollieren, ob die daraufhin blau hervorgehobene Andockposition Ihren Vorstellungen entspricht und dann das Fenster loszulassen.

Für Visual Studio 2010 wurde die Platzierung der Fenster sogar noch einmal verbessert:

- Sie sehen die Andockpositionen jetzt auch, wenn Sie die Registerkarte eines Editorfensters ziehen, um die Editorfenster nebeneinander statt übereinander anzuzeigen
- Sie können jetzt an den Docksymbolen besser erkennen, ob Ihr Fenster gleichberechtigt neben dem anderen Fenster angezeigt wird (bei einer Vergrößerung des Visual Studio-Fensters profitieren dann beide Fenster) oder ob Ihr Fenster angedockt wird. Letztere Dockpositionen sind an dem dreieckigen Pfeilsymbol erkennbar und werden nur für andockbare Toolfenster angeboten (angedockte Fenster behalten bei einer Vergrößerung des Visual Studio-Fensters ihre ursprüngliche Breite bzw. Länge bei).
- Sie können Fenster nicht nur aus der IDE herausziehen, Sie können sie jetzt sogar auf mehrere Monitore verteilen – vorausgesetzt natürlich, Sie haben mehrere Monitore angeschlossen

Dirk Louis, Shinja Strasser, Thorsten Kansy: Microsoft Visual C# - Das Entwicklerbuch, Microsoft Press 2010 (ISBN 978-3-86645-529-0)

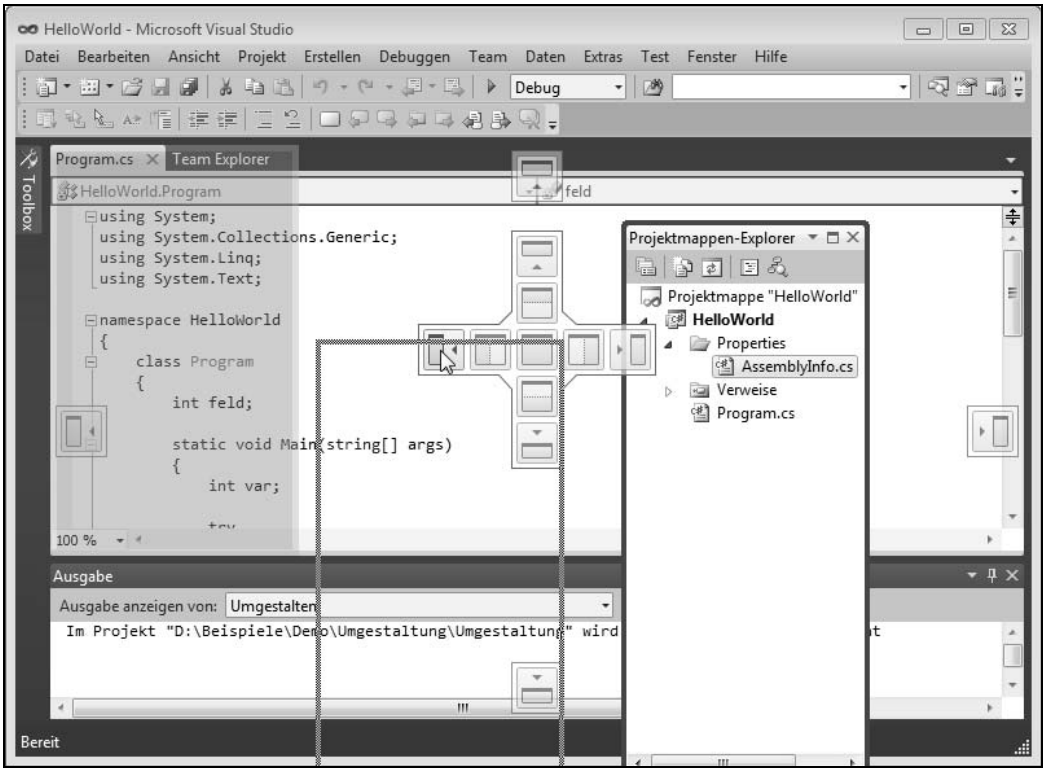

Abbildung 2.23 Das Toolfenster des Projektmappen-Explorers wurde aus dem rechten Rand herausgezogen und soll nun an den linken Rand parallel zum Editorfenster angedockt werden

Die Pin-Nadel

Andockbare Toolfenster verfügen über eine Pin-Nadel, mit der die Fenster festgesteckt oder freigegeben werden können. Nicht festgesteckte Fenster schrumpfen zu einem Platz sparenden Registerkarten-Reiter zusammen, wenn die Maus aus dem Fenster herausbewegt wird. Bewegt der Programmierer die Maus über die Registerkarte, springt das Fenster wieder auf und bleibt sichtbar, solange die Maus über dem Fenster weilt.

Fazit: Fenster, die Sie immer oder häufig benötigen, sollten Sie festpinnen. Fenster, die Sie nur gelegentlich benötigen, sollten Sie freigeben, um Platz zu sparen.

Der IDE-Navigator

Auch für Entwickler, die ihre Fenster nicht andocken, sondern lieber überlappen lassen, hat Visual Studio 2010 etwas anzubieten. Wenn Sie einmal ein benötigtes Fenster nicht gleich wieder finden, drücken Sie einfach die Tastenkombination Strg + ⇆.

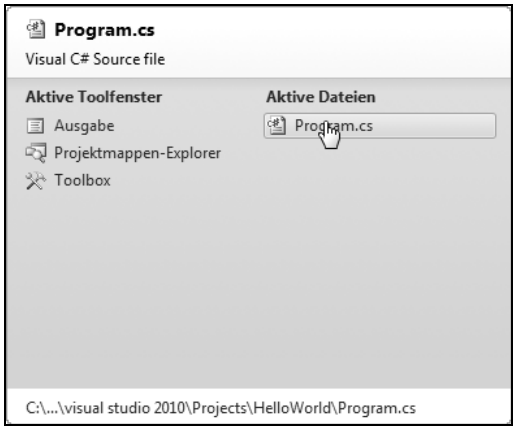

Abbildung 2.24 Der IDE-Navigator

Halten Sie die Strg-Taste gedrückt und wählen Sie mit der Maus oder den Pfeiltasten das Fenster aus, das Sie in den Vordergrund holen möchten.

Navigation

Einen nicht geringen Teil ihrer Zeit verbringen Programmierer mit der Beantwortung ganz trivialer Fragen: Wo wird ein Element verwendet? Wo ist es definiert? Wie komme ich am schnellsten zu seiner Definition? Gute IDEs wie Visual Studio bieten dem Programmierer daher passende Tools an, die ihn bei der Orientierung und der Navigation in seinem Code unterstützen.

Suchen

Die klassische Vorgehensweise zum Aufspüren der Vorkommen eines bestimmten Bezeichners ist natürlich die Suche mit der Suchfunktion aus dem *Bearbeiten*-Menü. Obwohl die Verwendung des Suchbefehls sicherlich allen Lesern bekannt ist, möchte ich dennoch auf einige Besonderheiten hinweisen, welche die Suchoption in Visual Studio aufzuweisen hätte. Dies beginnt damit, dass Visual Studio über das Menü *Bearbeiten/Suchen und Ersetzen* gleich drei verschiedene Suchbefehle anbietet:

- *Schnellsuche* Strg+F. Erlaubt die Suche nach beliebigen Textpassagen oder regulären Ausdrücken. Sie können wählen, ob Sie in dem aktuellen Block, dem aktuellen Dokument, allen geöffneten Dokumenten, dem aktuellen Projekt oder der gesamten Projektmappe suchen möchten. Beim Drücken der *Weitersuchen*-Schaltfläche bzw. der ↵-Taste wird das nächste Vorkommen im Code-Editor angezeigt und markiert.
- *In Dateien suchen* Strg+⇧+F. Erlaubt die Einschränkung der Suche auf bestimmte Dateitypen (*Suchoptionen*) und listet alle gefundenen Vorkommen im Fenster *Suchergebnisse* auf. Wenn Sie dort auf einen Eintrag doppelklicken, wird die betreffende Stelle im Code-Editor angezeigt.

- *Symbol suchen* Alt+F12. Erlaubt die Suche nach Bezeichnern (Namespaces, Klassennamen, Membernamen, Variablen etc.). Sie können wählen, ob Sie in der Projektmappe, den Bibliotheken des .NET Framework oder einem von Ihnen zusammengestellten Satz von Suchorten (*Komponenten*) suchen möchten. Letzteren können Sie im Objektkatalog (*Ansicht/Objektkatalog*) über die Option *Benutzerdefinierten Komponentensatz bearbeiten* aus dem Dropdown-Listenfeld *Durchsuchen* definieren. Alle gefundenen Vorkommen werden im Fenster *Suchergebnisse* aufgelistet. Wenn Sie dort auf einen Eintrag doppelklicken, wird die betreffende Stelle im Code-Editor angezeigt.

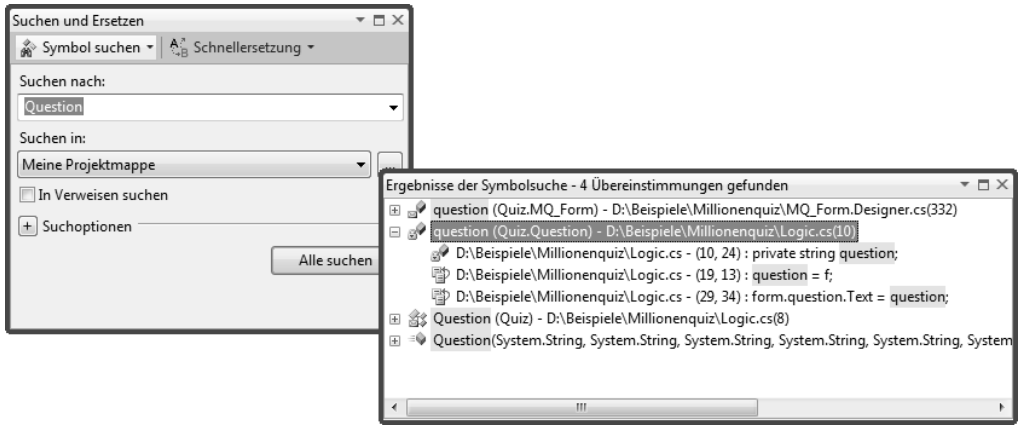

Abbildung 2.25 Die Suche nach Symbolen (neu in Visual Studio 2010)

Für die Suche in den geöffneten Dokumenten gibt es ab Visual Studio 2010 noch eine weitere Option. Wenn Sie den Bezeichner, an dessen weiteren Vorkommen Sie interessiert sind, vor sich sehen, setzen Sie einfach den Textcursor in den Bezeichner und warten Sie. Schon nach kurzer Zeit sind alle Vorkommen in den geöffneten Dokumenten grau unterlegt.

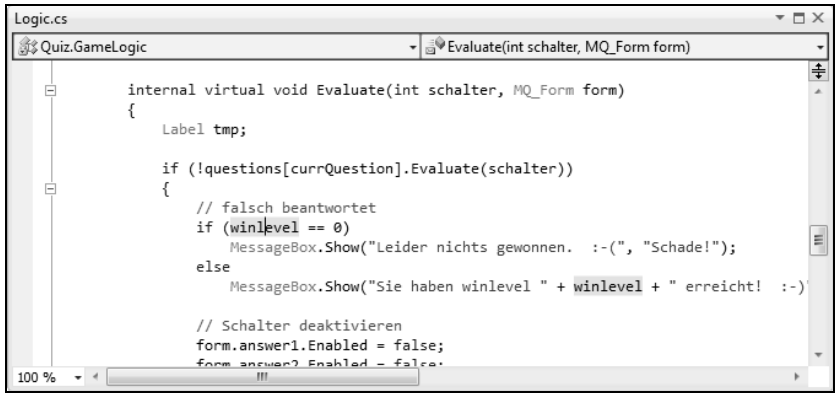

Abbildung 2.26 Automatische Markierung aller Vorkommen

HINWEIS Um die automatische Markierung der Vorkommen eines Bezeichners auszuschalten, rufen Sie den Befehl *Extras/Optionen* auf und deaktivieren Sie auf der Seite *Text-Editor/C#/Erweitert* die Option *Verweise auf Symbole unter Cursor markieren*.

Zur Definition wechseln

Visual Studio kennt unzählige Wege, wie Sie zur Definition eines Bezeichners wechseln können.

- Wenn Sie den Bezeichner vor sich im Editorfenster sehen, klicken Sie ihn mit der rechten Maustaste an und rufen Sie im Kontextmenü den Befehl *Gehe zu Definition* auf
  - Wenn Sie von der Definition zu dem Vorkommen zurückspringen möchten, setzen Sie zuvor ein Lesezeichen, siehe Befehle im Menü *Bearbeiten/Lesezeichen*
  - Wenn Sie die Definition nur einsehen möchten, öffnen Sie zuvor das Codedefinitionsfenster (Menübefehl *Ansicht/Codedefinitionsfenster*)
- Suchen Sie nach den Vorkommen des Bezeichners mittels des Befehls *Bearbeiten/Suchen und Ersetzen/In Dateien suchen* und doppelklicken Sie im Fenster für die Suchergebnisse auf den Eintrag für die Definition des Elements
- Wenn Sie gerade den Objektkatalog, die Klassenansicht oder die Aufrufhierarchie geöffnet haben, können Sie sich die Definitionen zu den aufgelisteten Elementen anzeigen lassen
  - Wenn Sie die Definition nur einsehen möchten, öffnen Sie zuvor das Codedefinitionsfenster (Menübefehl *Ansicht/Codedefinitionsfenster*), achten Sie darauf, dass Sie beide Fenster im Blick haben, und klicken Sie im Objektkatalog (der Klassenansicht oder der Aufrufhierarchie) dann auf das Element, dessen Definition angezeigt werden soll
  - Wenn Sie im Editor zu der Definition springen möchten, doppelklicken Sie im Objektkatalog (der Klassenansicht oder der Aufrufhierarchie) auf das betreffende Element

Klassenansicht und Objektkatalog

Klassenansicht und Objektkatalog sind Übersichtsfenster, die Ihnen helfen, den Überblick über die Namespaces, Klassen und Klassenmember Ihres Projekts zu bewahren. Beide Fenster verfügen über diverse Möglichkeiten zum Filtern und Sortieren der angezeigten Informationen (seit Version 2010 unterstützt durch ein eigenes Suchfeld) und beide Fenster erlauben das Laden der zugehörigen Definitionen (siehe oben).

Aufrufhierarchie

Ebenfalls neu in Visual Studio ist die Aufrufhierarchie – das Entwurfszeit-Pendant zur Aufrufliste des Debuggers (siehe Kapitel 47). Sie ist das Analyseinstrument Nummer 1, wenn es darum geht, herauszufinden, von wo aus eine Methode (inklusive Konstruktoren und Eigenschaften) aufgerufen wird oder welche anderen Methoden sie selbst aufruft.

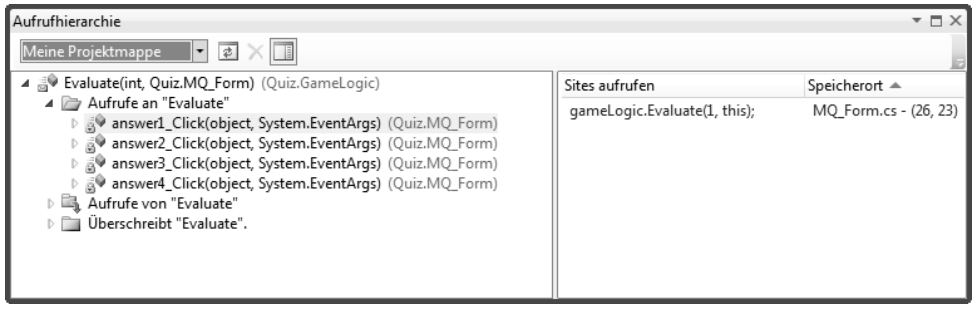

Abbildung 2.27 Die Aufrufhierarchie zur Methode *Evaluate*()

Um die Aufrufhierarchie einer Methode einzusehen, klicken Sie im Editorfenster irgendwo mit der rechten Maustaste auf den Namen der zu untersuchenden Methode und rufen Sie im Kontextmenü den Befehl *Aufrufhierarchie anzeigen* auf. In der Aufrufhierarchie erscheint sodann ein Stammknoten für die besagte Methode und darunter expandierbare Knoten *Aufrufe an* (für die Methoden, die die aktuelle Methoden aufrufen) und *Aufrufe von* (für die Methoden, die die aktuelle Methode selbst aufruft) sowie gegebenenfalls einen Knoten *Überschreibt*, wenn es sich um eine virtuelle Methode mit Überschreibungen handelt.

- Wenn Sie die Knoten expandieren und wie in der Abbildung auf eine der aufgelisteten Methoden klicken, erscheint im Detailfenster rechts der zugehörige Aufruf. (Wenn Sie gleichzeitig das Codedefinitionsfenster geöffnet haben, sehen Sie in diesem gleichzeitig die Definition der angeklickten Methode.)
- Ein Doppelklick auf eine der gelisteten Methoden lädt die Methodendefinition ins Editorfenster.

HINWEIS Wenn Sie sich die Aufrufhierarchie weiterer Methoden anzeigen lassen, wird im Fenster der Aufrufhierarchie für jede dieser Methode ein eigener Stammknoten angelegt. Sollte Ihnen dies zu unübersichtlich werden, löschen Sie einfach nicht mehr interessierende Stammknoten, indem Sie die Knoten auswählen und in der Symbolleiste auf *Stamm entfernen* klicken bzw. den gleichnamigen Befehl im Kontextmenü aufrufen.

Hilfe

Visual Studio unterstützt Sie mit einem umfangreichen, nahezu allgegenwärtigen Hilfesystem. Die Arbeit mit der Visual Studio-Hilfe ist weitgehend intuitiv verständlich und den meisten Lesern sicherlich wohl vertraut. Dennoch kann es ob der schieren Komplexität dieses Systems nicht schaden, sich einen kleinen Überblick zu verschaffen und sich mit einigen – vielleicht weniger bekannten – Punkten vertraut zu machen.

Übersicht

Das Visual Studio-Hilfesystem ist ein Konglomerat verschiedener Technologien und Zugangsmöglichkeiten:

- Das Haupteingangsportal zum Hilfesystem ist natürlich das *Hilfe*-Menü, und dort insbesondere der Befehl *Hilfe/Hilfe anzeigen*.
- Alle größeren Dialogfelder sind mit einer Hilfe-Schaltfläche versehen (Fragezeichen in der Titelleiste), über den Sie sich Informationen zu den Elementen des Dialogfelds anzeigen lassen können. Sie brauchen dazu lediglich auf das Fragezeichen zu klicken und schon wird die zu dem Dialogfeld gehörende Hilfeseite aufgerufen.
- Durch Drücken der Taste F1 rufen Sie zu einem Element der Visual Studio-Oberfläche, einem im

Quelltext markierten Wort oder einer Compiler-Meldung die Kontexthilfe auf.

- Eine wichtige Hilfe für den Programmierer sind natürlich auch die zahlreichen Beispielprogramme. Wo Sie diese finden, erfahren Sie nach einem Klick auf den Menübefehl *Hilfe/Beispielcode*.
- Wenn Sie sich mit anderen Entwicklern austauschen möchten, können Sie den Befehl *Hilfe/MSDN-Foren* als Zugang zum Eingangsportal der Community nutzen.
- Bei akuten Problemen können Sie sich mit dem technischen Support *(Hilfe/Technischer Support)* oder der Community (*Hilfe/Problem melden*) in Verbindung setzen.

Lokal oder Online

Wenn Sie die Hilfe das erste Mal über den Befehl *Hilfe/Hilfe anzeigen* aufrufen, werden Sie gefragt, ob Sie die Onlinehilfe im Internet nutzen wollen. Wenn Sie sich hier für *Ja* entscheiden, wird bei jedem Aufruf der Hilfe die zugehörige Hilfeseite aus der MSDN heruntergeladen und in Ihrem Browser angezeigt. Die besonderen Vorzüge der Onlinehilfe sind natürlich ihre schiere Mächtigkeit und ihre Aktualität.

Wenn Sie vornehmlich Hilfe zu bestimmten Themenkreisen suchen, können Sie auch die lokale Hilfe nutzen. Dazu weiter unten mehr.

Wie auch immer Sie sich entscheiden, Sie können natürlich jederzeit zwischen lokaler und Onlinehilfe wechseln. Sie müssen dazu nur den Menübefehl *Hilfe/Hilfeeinstellungen verwalten* aufrufen und – gegebenenfalls nach Festlegung des Verzeichnisses für die lokalen Hilfedateien – über den Link *Onlinehilfe oder lokale Hilfe auswählen* das Dialogfeld öffnen, in dem Sie dann schließlich Ihre Auswahl treffen.

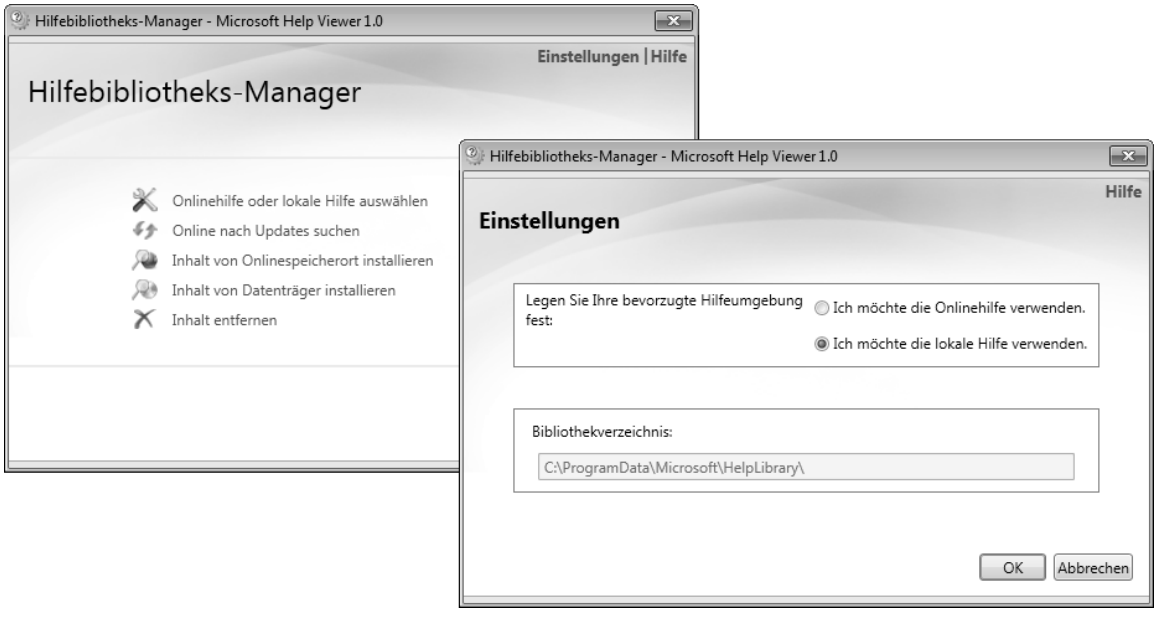

Abbildung 2.28 Die Hilfe wechseln

Die Online-Hilfe

Wenn Sie sich für die Onlinehilfe entschieden haben, lädt der Menübefehl *Hilfe/Hilfe anzeigen* die Startseite der Visual Studio-Dokumentation in Ihren Browser. Anschließend können Sie über die angebotenen Links (vorzugsweise im Bereich *Dokumentation*) oder über die Suchmaschine nach den von Ihnen benötigten Informationen forschen.

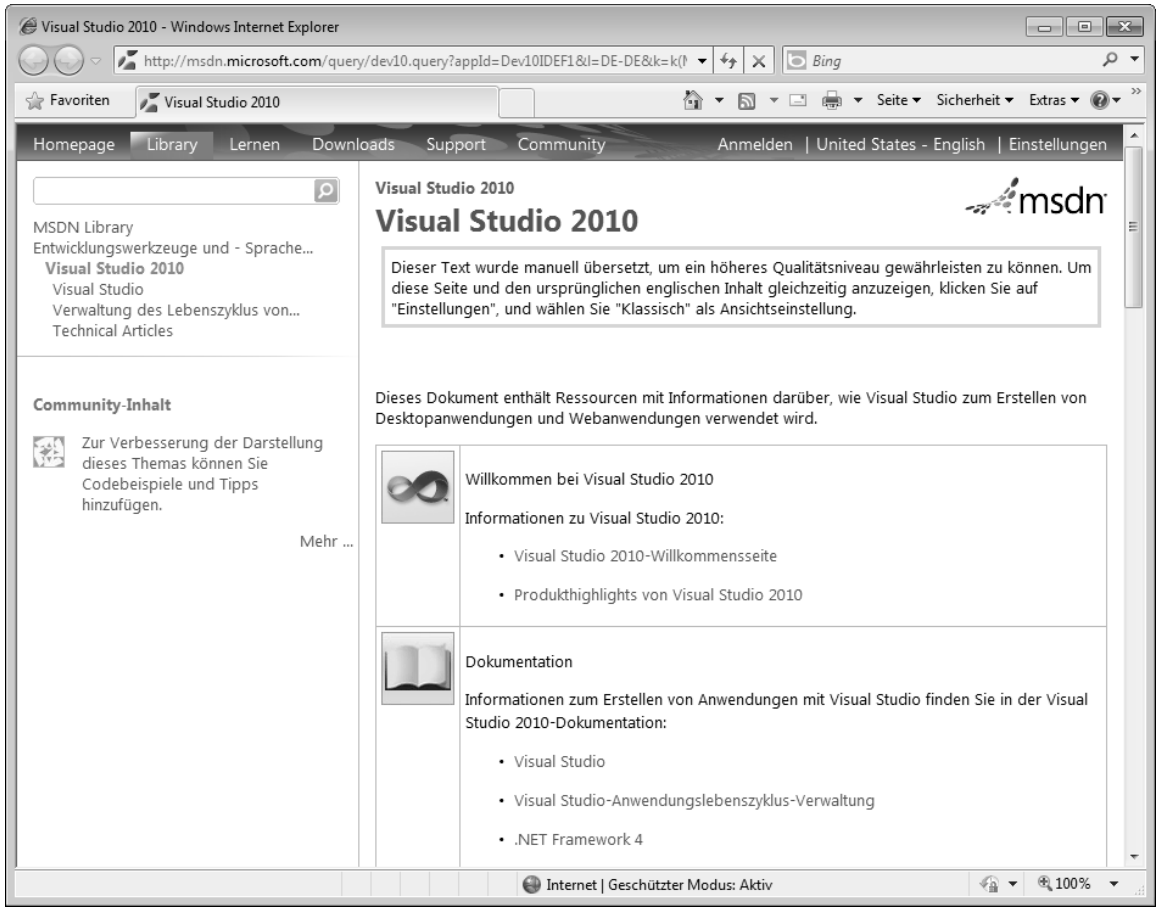

Abbildung 2.29 Die Hilfe im Windows Explorer (Online-Version)

Design kontra Funktionalität

Das Standarddesign der Onlinehilfe ist zwar grafisch sehr ansprechend, zum effizienten Nachschlagen von Informationen wird sich der eine oder andere Programmierer aber wohl eine etwas nüchternere, funktionalere Darstellung wünschen. Aus diesem Grund bietet die MSDN neben der standardmäßigen Kompakt-Darstellung noch zwei weitere Darstellungen an: *Ohne Skript* und *Klassisch*.

Um aus der Kompakt-Darstellung zu einer anderen Darstellung zu wechseln, klicken Sie in der Menüleiste oben rechts auf den Link *Einstellungen*. Auf der daraufhin erscheinenden Seite können Sie dann eine andere Darstellung wählen.

- Die *Ohne Skript*-Ansicht hat den Vorzug, dass sie etwas übersichtlicher und schneller ist.
- Die *klassische* Darstellung besticht durch ein Inhaltsverzeichnis und die teils zweisprachige Präsentation der Hilfeseiten.

  Besonders Programmierer, die hauptsächlich die deutschen Seiten lesen, werden es begrüßen, dass sie sich nun mit einem Klick auf *Inhalt anzeigen* am oberen Seitenrand den Originaltext anzeigen lassen können. So lassen sich Übersetzungsfehler wie z.B »Der Lambda-Ausdruck x => x * x bedeutet "x wechselt x Mal zu x".« schnell entlarven (der Originaltext heißt in diesem Fall erwartungsgemäß: »is read "x goes to x times x."«).

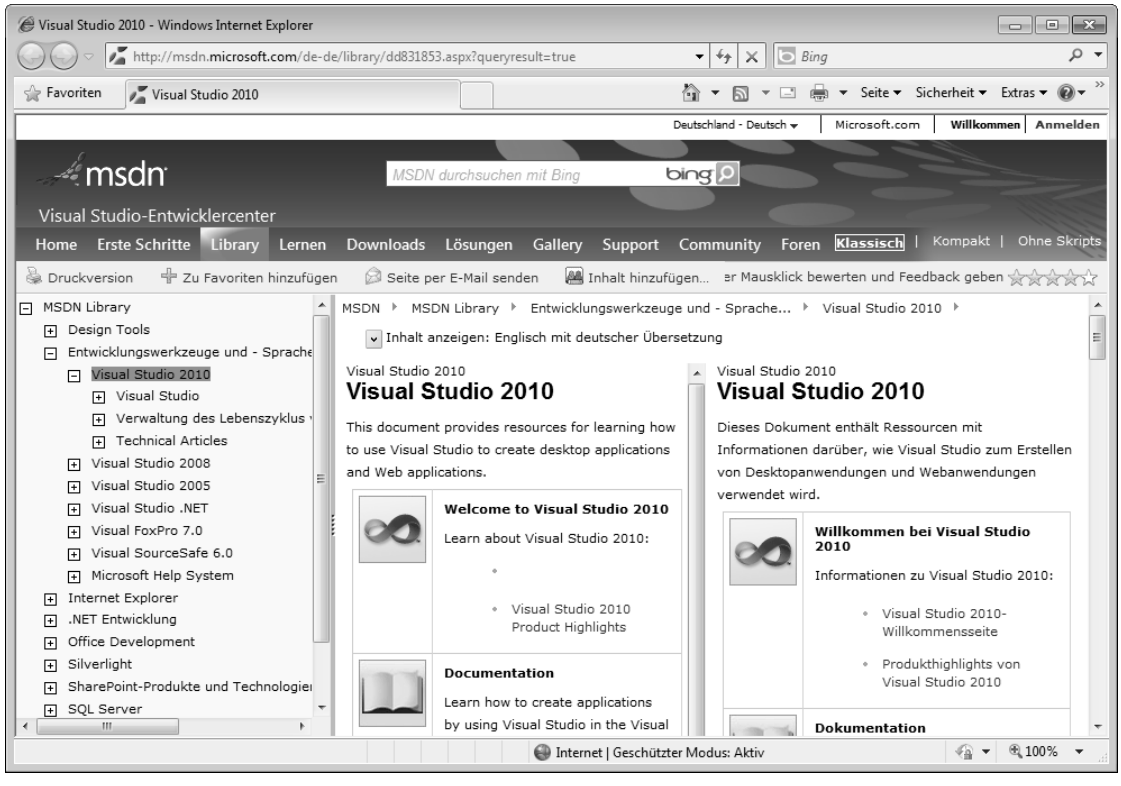

Abbildung 2.30 Die Onlinehilfe in der klassischen Ansicht

In den Darstellungen *Ohne Skript* und *Klassisch* sind die Links zum Wechseln der Darstellung direkt in die Menüleiste oben rechts integriert.

Die C#-Referenz und die Bibliotheksdokumentation

Als Programmierer werden Sie naturgemäß viel in der C#-Referenz bzw. in der Dokumentation der .NET-Bibliotheken recherchieren.

- Die C#-Referenz finden Sie in der klassischen Ansicht unter *MSDN Library/Entwicklungswerkzeuge und -sprachen/Visual Studio 2010/Visual Studio/Visual Studio-Sprachen/Visual Basic und Visual C#/C# Reference*.

- Die Dokumentation der der .NET-Bibliotheken finden Sie in der klassischen Ansicht unter *MSDN Library/Entwicklerbibliothek/.NET-Entwicklung* sowie unter *MSDN Library/.NET-Entwicklung/.NET-Framework/.NET-Framework-Klassenbibliothek*.

Der schnellste Weg zur Dokumentation eines Schlüsselwortes der Sprache, einer Klasse oder eines Klassenmembers führt aber zweifelsohne über die Kontexthilfe. Setzen Sie dazu einfach den Textcursor in das betreffende Wort und drücken Sie die Taste F1.

Wenn Sie mittels des Suchfelds der Hilfeseiten nach Klassen, Enumerationen, Namespaces, Eigenschaften oder Methoden suchen, geben Sie hinter den Suchbegriffen immer als Suffix -Klasse, -Enumeration, -Namespace, -Eigenschaft oder -Methode an. Sie erhalten dann weniger, aber dafür bessere Treffer.

Die lokale Hilfe

Wenn Sie sich für die lokale Hilfe entschieden haben, müssen Sie – sofern dies nicht schon geschehen ist – zunächst die Sie interessierenden Inhalte auf Ihren Rechner laden.

Rufen Sie dazu den Menübefehl *Hilfe/Hilfeeinstellungen verwalten* auf. Im Dialogfeld des Hilfebibliotheks-Managers können Sie sich dann entscheiden, ob Sie Inhalte aus dem Internet (Onlinespeicherort) oder von einem Datenträger installieren möchten.

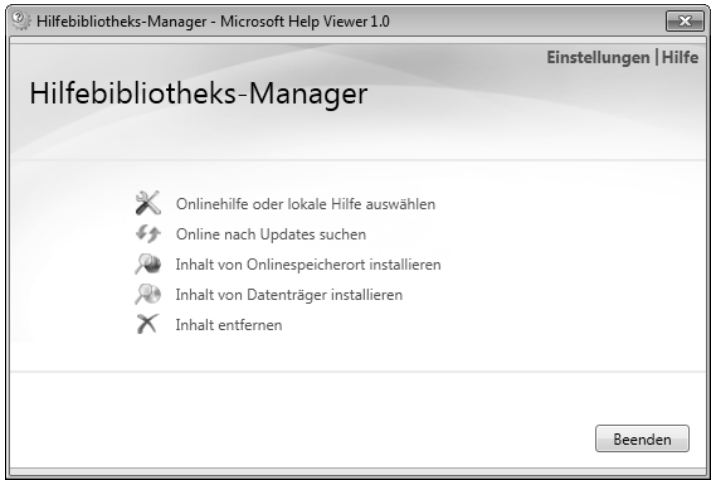

Abbildung 2.31 Das Dialogfeld des Hilfebibliotheks-Managers

Bei einer Installation von einem Datenträger müssen Sie das Verzeichnis der *HelpContentSetup.msha* angeben. Auf der Installations-DVD von Visual Studio ist dies das Verzeichnis *ProductDocumentation*.

Bei einer Installation aus dem Internet müssen Sie ein wenig warten, bis die Liste der verfügbaren Dokumentationen aufgebaut ist. Danach können Sie die Sie interessierenden Dokumentationen über die *Hinzufügen*-Links auf Ihren Rechner für den Download vormerken. Angestoßen wird der Download dann durch Drücken der *Aktualisieren*-Schaltfläche.

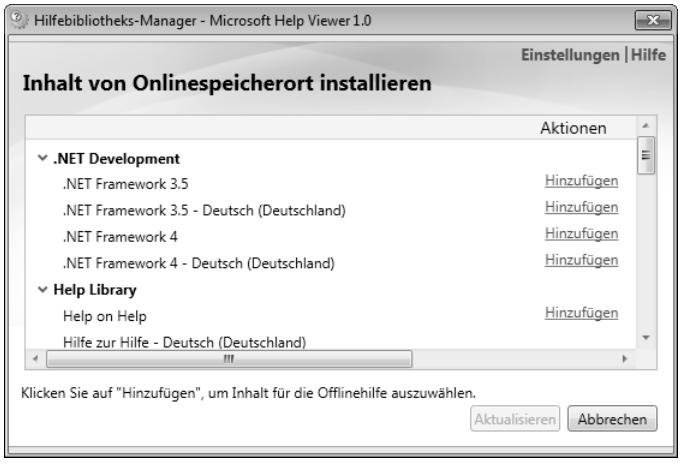

Abbildung 2.32 Hier wählen Sie die Dokumentationen aus, die Sie lokal installieren möchten

Nach der Installation können Sie die Internetverbindung trennen und die Hilfe nutzen. Die Darstellung der Hilfeseiten entspricht im Wesentlichen der Kompakt-Darstellung der Online-Hilfe. Aufruf und Nutzung der Hilfe sind zur Online-Hilfe identisch.

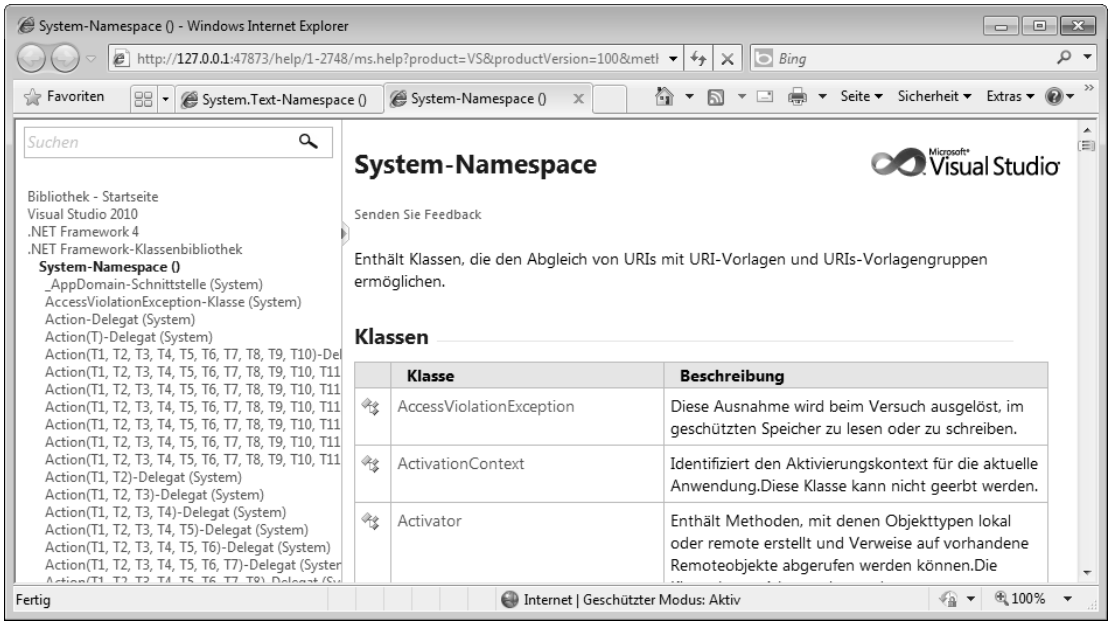

Abbildung 2.33 Die lokale Hilfe nach Aufruf der Kontexthilfe zum Namespace System (Markieren in Code-Editor, Drücken von F1).

Online-Aktualisierung

Über den Link *Online nach Updates suchen* im Hilfebibliotheks-Manager (Aufruf über *Hilfe/Hilfeeinstellungen verwalten*) können Sie die lokal installierten Hilfedateien jederzeit auf dem aktuellen Stand halten.

Dirk Louis, Shinja Strasser, Thorsten Kansy: Microsoft Visual C# - Das Entwicklerbuch, Microsoft Press 2010 (ISBN 978-3-86645-529-0)