

Teil B

Die Visual Studio- Entwicklungsumgebung

13	Ein Flug über die Weiten der Visual Studio-IDE
37	Codeeditor und Formular-Designer enthüllt
109	Tipps & Tricks für das angenehme Entwickeln zuhause und unterwegs

Bei einem Buch dieses Umfangs ist es vergleichsweise schwierig, eine Struktur zu schaffen, die auf die Bedürfnisse jedes Lesers ideal einzugehen in der Lage ist. Redundanzen bei Erklärungen lassen sich dabei nicht vermeiden – eigentlich sind sie an einigen Stellen sogar recht sinnvoll. Gerade dieses erste Kapitel, das die Entwicklungsumgebung – die so genannte *IDE (Integrated Development Environment, etwa: integrierte Entwicklungsumgebung)* – vorstellt, und Ihnen Tipps und Tricks gibt, die Ihnen das Entwickeln im Büro und auch unterwegs so einfach wie möglich machen sollen, ist davon besonders betroffen: Es nimmt einige Features vorweg, auf die im Laufe des Buches natürlich noch detaillierter eingegangen wird. Und gerade für Leser, die schon einige Erfahrungen beim Entwickeln von .NET mitbringen, ist die Vorgehensweise dieses Kapitels wahrscheinlich dennoch die beste, um sich in der neuen Umgebung schnellstmöglich zurechtzufinden.

2 Ein Flug über die Weiten der Visual Studio-IDE

13	Die Startseite – der erste Ausgangspunkt für Ihre Entwicklungen
16	Die IDE auf einen Blick
20	Die wichtigsten Toolfenster
34	Die wichtigsten Tastenkombinationen auf einen Blick

Die IDE – die integrierte Entwicklungsumgebung – ist das »Modul« in Visual Studio, mit dem Sie ganz sicher die meiste Zeit verbringen werden. Je besser Sie die wichtigsten Funktionen und Eigenarten dieses unglaublich mächtigen Werkzeugs kennen, desto mehr Zeit werden Sie bei der täglichen Entwicklung Ihrer Softwareprojekte sparen.

Eben weil die Visual Studio-IDE gespickt ist mit Funktionen, Toolfenstern, Werkzeugen und anderen Hilfsmitteln kann es passieren, dass Sie den Überblick verlieren oder sich die Arbeit unnötig schwer machen, da Sie vielleicht mit einem geeigneten Werkzeug, das Sie einfach nur noch nicht kennen, schneller zum Ziel kämen, als Sie es bislang gemacht haben.

Die Startseite – der erste Ausgangspunkt für Ihre Entwicklungen

Jedes Mal, wenn Sie Visual Studio 2005 starten, begrüßt Sie die IDE mit der Startseite, etwa wie in Abbildung 2.1 zu sehen. Die Startseite zeigt Ihnen nicht nur Ihre zuletzt bearbeiteten Projekte, die Sie direkt per Mausklick auf den jeweiligen Eintrag in der IDE öffnen können, sondern sie liefert – Internetanbindung vorausgesetzt – auch aktuelle Infos rund um Visual Studio 2005, SQL Server 2005 und das Framework 2.0.

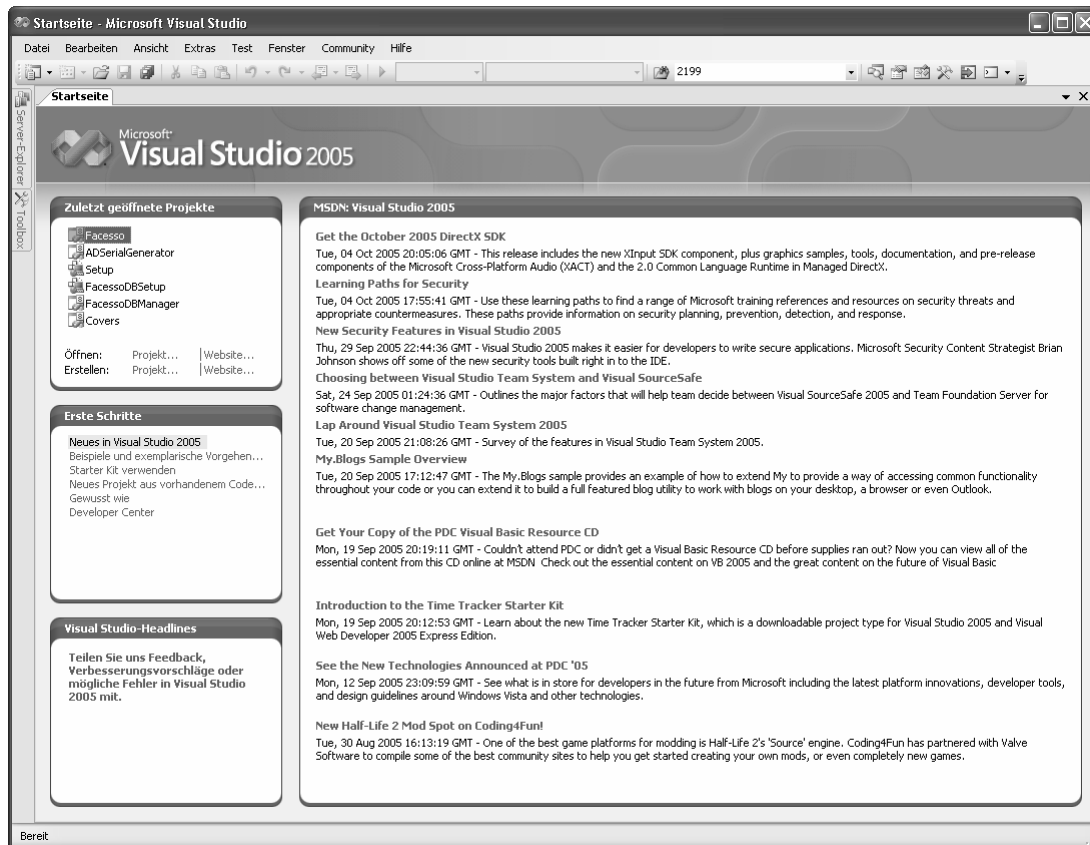


Abbildung 2.1: Nach dem Start von Visual Studio 2005 begrüßt Sie die Startseite, auf der Sie die zuletzt bearbeiteten Projekte öffnen können, und die Ihnen aktuelle Infos rund um Visual Studio liefert

In der linken, oberen Ecke der Startseite finden Sie die letzten Projekte, die Sie bearbeitet haben. Ein einfacher Mausklick genügt, um das entsprechende Projekt zu öffnen. Sie können auch direkt von hier aus neue Projekte anlegen oder Projekte öffnen, die sich nicht in der Liste der zuletzt verwendeten Projekte befinden.

Der Visual Studio-Nachrichten-Channel

Dominierend auf der Startseite ist der Visual Studio-Nachrichten-Channel, der Sie, sofern Sie über eine funktionsfähige Internetverbindung verfügen, immer mit aktuellen Informationen rund um Visual Studio 2005, SQL Server 2005 und dem Framework 2.0 versorgt. Damit verwandte Themen werden Sie hier sicherlich ebenfalls finden.

Welche Informationen hier genau angezeigt werden, richtet sich übrigens nach dem eingestellten Nachrichtenchannel, den Sie jederzeit ändern können, wenn Ihnen ein besserer bekannt ist. Dazu

rufen Sie mit *Extras | Optionen* den *Optionen*-Dialog von Visual Studio auf und wählen *Start* im Zweig *Umgebung*.¹

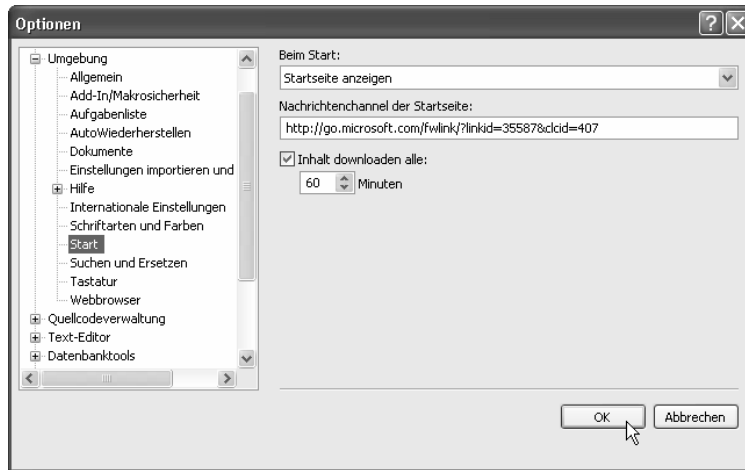


Abbildung 2.2: Mit dieser Registerkarte konfigurieren Sie den verwendeten Nachrichtenchannel der Startseite

Unter *Nachrichtenchannel der Startseite* geben Sie den URL Ihres favorisierten Nachrichtenchannels ein.

Anpassen der Liste der zuletzt bearbeiteten Projekte

In einigen Fällen können sich ältere Projekte, die Sie nicht mehr bearbeiten, als störend in der Liste erweisen. Falls Sie mit dem *Registry Editor* von Windows vertraut sind, dann – und nur dann – können Sie unter `HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\8.0\ProjectMRUList` die Liste der Projektdateien einsehen und gegebenenfalls ändern. Das Eintragsformat der Dateien gestaltet sich folgendermaßen:

```
File1:REG_SZ:C:\Pfad zur gewünschten Projektmappe\test1.sln
File3:REG_SZ:C:\Pfad zur gewünschten Projektmappe\test3.sln
File2:REG_SZ:C:\Pfad zur gewünschten Projektmappe\test2.sln
File4:REG_SZ:C:\Pfad zur gewünschten Projektmappe\test4.sln
```

Beachten Sie, dass die Einträge nicht notwendigerweise in numerischer Reihenfolge aufgelistet sind. **Und ganz wichtig:** Die Einträge müssen lückenlos nummeriert sein, damit alle Projekte angezeigt werden. Es reicht also nicht aus, einen Eintrag aus der Liste zu entfernen, Sie müssen auch dafür sorgen, dass die entstehende Lücke in der Nummerierung ausgeglichen wird. Am einfachsten ist es, die Angaben des letzten Eintrags der Liste in den zu löschenden Eintrag zu übertragen und dann den letzten Eintrag der Liste zu löschen.

Die Projekte, die hier verzeichnet sind, beziehen sich sowohl auf die Projekte der Startseite als auch auf die Projekte, die Sie sehen, wenn Sie aus dem Menü *Datei* den Menüpunkt *Zuletzt geöffnete Projekte* auswählen.

¹ Im Bedarfsfall müssen Sie bei einigen Visual Studio-Versionen erst im Dialog das Kontrollkästchen *Alle Einstellungen anzeigen*, um Zugriff auf alle Optionen nehmen zu können.

Die Liste der zuletzt geöffneten Dateien können Sie übrigens ebenfalls bearbeiten. Verfahren Sie dazu wie oben beschrieben, verwenden Sie jedoch statt des genannten Schlüssels den Schlüssel `HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\8.0\FileMRUList`.

Die IDE auf einen Blick

Auf der nächsten Seite finden Sie eine Grafik, die Ihnen die wichtigsten Elemente der IDE demonstriert. Um es ganz klar zu sagen: Sollte Ihre persönliche IDE-Konfiguration schon jetzt oder nach einer Weile so ausschauen, wie die IDE-Konfiguration in der Grafik auf der nächsten Seite – stoppen Sie sofort, was auch immer Sie machen. Rudern Sie zurück. Setzen Sie die komplette IDE (übrigens mit dem Befehl *Fenster / Fensterlayout zurücksetzen*) in ihren Ausgangszustand zurück, und denken Sie darüber nach, welche der IDE-Elemente Sie am häufigsten benötigen, und bringen Sie diese gemäß der Erklärungen der nächsten Abschnitte in den Vordergrund.

TIPP: Um sich mit allen Elementen in Visual Studio .NET vertraut machen zu können, benötigen Sie ein Projekt zum »Spielen«. Da Sie das in den folgenden Beispielen verwendete Projekt noch nie verwendet haben, steht es natürlich noch nicht in der Liste.

- Wählen Sie aus dem Menü *Datei* den Menüpunkt *Projekt/Projektmappe öffnen*.
- In der Datei-Auswahl, die jetzt erscheint, wählen Sie das Laufwerk und Verzeichnis, in dem Sie die Begleitdateien installiert haben. Wählen Sie dort den Ordner `\VB 2005 - Entwicklerbuch\B - IDE\02 - Ereignisse`.
- Doppelklicken Sie auf den Dateinamen `Ereignisse.sln`, um das Programm zu laden.
- Wählen Sie anschließend im Menü *Erstellen* den Menüpunkt *Ereignisse neu erstellen*.

Die folgende Grafik dient lediglich der Orientierungshilfe, als Ausgangspunkt sozusagen für die Abschnitte, die sich im Detail mit den wichtigsten Elementen beschäftigen. Falls Sie bemerken, dass Ihnen der Platz auf dem Bildschirm zu eng wird, ziehen Sie es wirklich in Erwägung, einen Monitor mit einer größeren Auflösung oder besser, da noch mehr Platz und obendrein billiger, einen zweiten Monitor für den Mehrmonitorbetrieb Ihres Computers anzuschaffen (Kapitel 3 liefert mehr Informationen dazu).

Die meisten Grafikkarten, die heutzutage in modernen Computern oder Notebooks verbaut werden, erlauben ohnehin den Anschluss eines zweiten Monitors und die entsprechende Erweiterung des Desktop auf diesen. Die zusätzliche Investition für einen zweiten Monitor wird sich schnell rechnen, denn glauben Sie mir: Eine Auflösung von 1024 x 768 Punkten auf nur einem verfügbaren Monitor führt fast zwangsläufig zu einer Überladung Ihres Sichtfeldes; Ihre Konzentration wird entweder durch das ständige Auf- und Zuklappen der benötigten Toolfenster oder durch zu viele zu kleine Toolfenster stark nachlassen! Außerdem verlieren Sie durch das ständige Navigieren zwischen den Elementen zu viel Zeit.

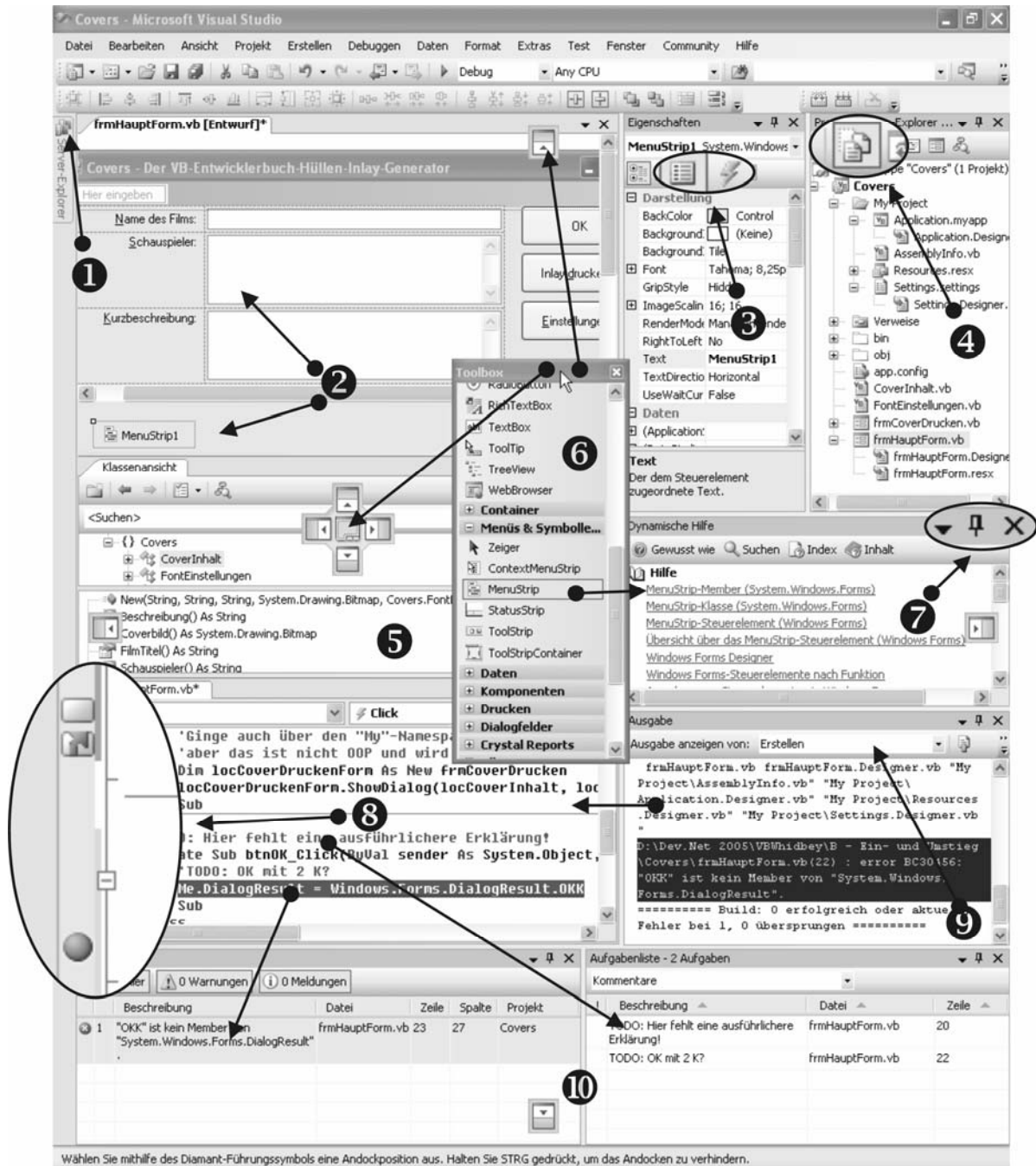


Abbildung 2.3: Die wichtigsten IDE-Elemente auf einen Blick. In den nachfolgenden Abschnitten finden Sie beschrieben, zu welchem Zweck sie dienen und wie sie untereinander interagieren und sich beeinflussen können

Genereller Umgang mit Fenstern in der Entwicklungsumgebung

Dass die Benutzeroberfläche der Entwicklungsumgebung unwahrscheinlich viele Möglichkeiten bietet, werden Sie seit Ihren ersten Experimenten mit Visual Studio bemerkt haben. Der Nachteil der Entwicklungsumgebung ist: Sie hält so viele Elemente parat, dass Sie leicht die Übersicht verlieren können. Der Vorteil: Die Entwicklungsumgebung können Sie auf Ihre eigenen Bedürfnisse zuschneiden, wie sonst kaum ein anderes Windowsprogramm. Schauen Sie sich die einzelnen Bereiche der Oberfläche ein wenig genauer an. Abbildung 2.3 zeigt Ihnen die wichtigsten Elemente der Visual Studio-IDE auf einen Blick.

Dokumentfenster

Die Fenster der Entwicklungsumgebung werden durch so genannte **Registerkartengruppen** verwaltet. Davon gibt es zwei verschiedene Arten: Zum einen die, die den eigentlichen Inhalt Ihrer Projektdateien (Code, Formulare) sowie die Startseite, Projekteigenschaften und u.U. bestimmte Werkzeuge wie die Klassenansicht eines Projektes abbilden. Diese werden **Dokumentfenster** genannt. Dokumentfenster werden dynamisch erstellt, wenn Sie Dateien oder andere Elemente öffnen oder erstellen. Die Liste der geöffneten Dokumentfenster wird im Menü *Fenster* angezeigt.

Die Möglichkeiten zur Verwaltung von Dokumentfenstern hängen weitestgehend vom Schnittstellenmodus ab, der unter *Allgemein, Umgebung* im Dialog *Optionen* ausgewählt wurde. Sie können wahlweise im Modus *Dokumente im Registerkartenformat* (Standard) oder im Modus *Multiple Document Interface* (MDI – etwa: Mehrfache Dokumentschnittstelle) arbeiten. Experimentieren Sie einfach mit diesen Einstellungen, um eine Ihren Bedürfnissen und Ihrem Geschmack entsprechende Umgebung für die Dokumentbearbeitung zu erstellen.

Toolfenster

Toolfenster werden im Menü *Ansicht* aufgelistet und sind durch die aktuelle Anwendung und deren Add-Ins definiert. Sie können in der IDE unterschiedlich konfiguriert werden:

- Automatisch ein- oder ausgeblendet (in der Abbildung mit ❶ gekennzeichnet)
- Im Registerformat und verknüpft mit anderen Toolfenstern
- Angedockt an die Ränder der IDE
- Nicht angedockt (»schwebend«) (in der Abbildung mit ❷ gekennzeichnet)
- Zur Anzeige als Dokumentfenster (in der Abbildung mit ❸ gekennzeichnet)
- Zur Anzeige auf anderen Monitoren

Zusätzlich können Sie gleichzeitig mehrere Instanzen bestimmter Toolfenster anzeigen lassen. So können Sie z. B. mehrere Fenster eines Webbrowsers anzeigen. Wählen Sie zum Erstellen einer neuen Toolfensterinstanz im Menü *Fenster* den Menübefehl *Neues Fenster* aus. Außerdem können Sie festlegen, wie sich die Betätigung der Schaltflächen *Schließen* und *Automatisch im Hintergrund* auf eine Gruppe zusammen angedockter Toolfenster auswirkt.

WICHTIG: Sie können nahezu jedes Toolfenster als Dokumentfenster behandeln (dazu öffnen Sie das Kontextmenü des jeweiligen Toolfensters durch Rechtsklick auf den Fenstertitel und wählen anschließend Dokument im Registerkartenformat); Sie können allerdings kein echtes Dokumentfenster wie den Quellcode einer Codedatei oder die Designer-Darstellung eines Formulars als Toolfenster darstellen.

Andocken von Toolfenstern

Im Modus *Dokumente im Registerkartenformat* können Sie festlegen bzw. verhindern, dass die Dokumentfenster angedockt werden können, indem Sie im Kontextmenü des Fenstertitels die Option *Andockbar* aktivieren bzw. deaktivieren. Im MDI²-Modus sind Dokumentfenster nicht andockbar.

TIPP: Bei einigen Dokumentfenstern innerhalb der IDE handelt es sich in Wirklichkeit um Toolfenster, für die die Andockfunktion deaktiviert wurde. Sie können diese Fenster andocken, indem Sie im Menü Fenster die Option *Andockbar* auswählen.

Um Fenster entweder an eine der Seiten der IDE oder in anderen Registerkartengruppen anzudocken, verfahren Sie folgendermaßen. Orientieren Sie sich dabei bitte an der Toolbox (bezeichnet mit ⑥) in Abbildung 2.3.

- Klicken Sie mit der Maus auf den Fenstertitel des Toolfensters, das Sie neu positionieren möchten, und halten Sie die Maustaste dabei fest.
- Sobald Sie beginnen, das Fenster zu verschieben, blendet die IDE Positionshilfen ein, wie Sie sie auch in der Abbildung erkennen können.
- Um ein Toolfenster an einer Seite der IDE anzudocken, ziehen Sie es auf eine der vier Positionshilfen, die an den vier Seiten der IDE zu finden sind. In der Abbildung erkennen Sie eine dieser Positionshilfen, die durch den von der Titelzeile der Toolbox nach oben laufenden Pfeil markiert ist.
- Um ein Toolfenster neben oder in einer Registerkartengruppe anzuordnen, bewegen Sie die Registerkarte in Richtung einer Registerkartengruppe (oder eines anderen Toolfensters, aus der dann nach dem Loslassen automatisch eine Registerkartengruppe wird). Die IDE bildet anschließend eine weitere Positionshilfe ein – immer in der Nähe der jeweiligen Registerkartengruppe – die aus fünf Symbolen besteht. Die äußeren Symbole erlauben das Anordnen des Toolfensters an der jeweiligen Seite der Registerkartengruppe (oder des anderen Toolfensters); das innere Symbol dient dem Zweck, das Toolfenster der Registerkartengruppe hinzuzufügen (oder ein einzelnes Toolfenster, das Sie ansteuern, zu einer Registerkartengruppe werden zu lassen).

Arbeiten mit Toolfenstern

- Um ein Toolfenster einer Registerkartengruppe zu aktivieren, klicken Sie einfach auf die entsprechende »Lasche« in der Registerkartengruppe.
- Möchten Sie, dass ein Toolfenster nur im Bedarfsfall aufgeklappt wird, wenn Sie es benötigen, und sich anschließend wieder schließt, wenn Sie seinen Umgebungsbereich verlassen, klicken Sie auf das Heftzwecken-Symbol, wie Sie es in Abbildung 2.3 mit ⑦ bezeichnet erkennen können.

² MDI: *Multi Document Interface*. Visual Studio stellt dabei ein umgebendes Hauptfenster dar, und *alle* anderen Dokumentfenster lassen sich hierin gleichberechtigt (ohne Registerkartenaufteilung) anordnen und organisieren.

- Wenn Sie ein Toolfenster versehentlich geschlossen haben, finden Sie die wichtigsten Fenster zur erneuten Aktivierung durch Menübefehle im Menü *Ansicht*. Beachten Sie dabei auch den Menüpunkt *Weitere Fenster*, der Zugriff auf die weniger wichtigen Fenster bereithält.
- Das Fensterlayout während des Debuggens eines Programmes ist unabhängig vom Fensterlayout der IDE im Entwurfsmodus. Sobald Sie Ihre Anwendung in der IDE starten, schaltet Visual Studio auf das Fensterlayout des Debug-Modus. Änderungen, die Sie an den Toolfensterkonfigurationen vornehmen, wirken sich nicht auf die Fensterkonfiguration aus, die gilt, wenn Sie sich im Entwurfsmodus befinden, und umgekehrt. Spezielle Toolfenster zum Debuggen finden Sie übrigens im Menü *Debuggen | Fenster*.

Wechseln zwischen aktiven Dokumentfenstern mit der Tastatur

Übrigens: Genau So wie Sie in Windows selbst mit der Tastenkombination **Alt+Tabulator** zwischen den verschiedenen laufenden Anwendungen wechseln können, haben Sie die Möglichkeit, in Visual Studio mit **Strg+Tabulator** zwischen den aktiven Dokumenten zu wechseln. Visual Studio blendet dabei als Orientierungshilfe ein Fenster ein, wie Sie es auch in der folgenden Abbildung sehen können.

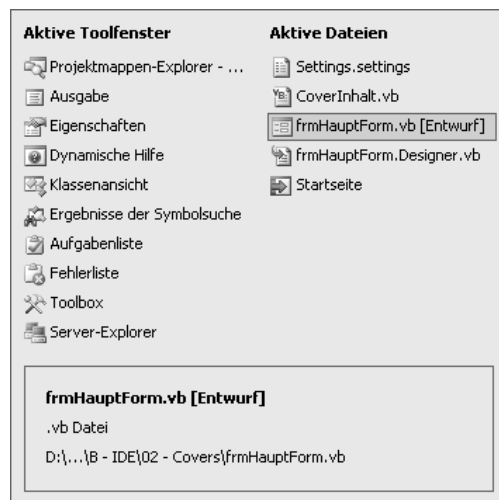


Abbildung 2.4: Mit **Strg+Tabulator** können Sie zwischen den geöffneten Dokumentfenstern wechseln

Die wichtigsten Toolfenster

Beim Stöbern durch die IDE werden Sie festgestellt haben, wie viele Werkzeuge Ihnen Visual Studio für den Produktiveinsatz anbietet – zu viele, um sie alle an dieser Stelle ausführlich darzustellen. Doch die wichtigsten Toolfenster, die, die Sie mit Abstand am häufigsten benötigen, finden Sie in den folgenden Abschnitten beschrieben.

Der Projektmappen-Explorer

Mithilfe des Projektmappen-Explorers navigieren Sie in Ihrem Projekt (in Abbildung 2.3 unter ④ zu sehen). Er zeigt eine Liste aller Dateien, aus denen Ihr Projekt bzw. Ihre Projektmappe besteht sowie eine Liste mit Verweisen auf alle Assemblies oder andere Projekte der Projektmappe, die ein Projekt verwendet. Die wichtigsten Funktionen zum Managen Ihres Projektes erreichen Sie, indem Sie das Kontextmenü aufrufen, wenn Sie sich mit dem Mauszeiger über dem Projektmappen-Explorer befinden.

Projektmappen und Projekte

Eine Projektmappe kann verschiedene Projekte enthalten. Dabei ist es egal, ob diese Projekte sich gegenseitig benötigen und aufrufen oder ob sie von einander völlig unabhängig sind. Sie definieren eines der Projekte als Startprojekt, indem Sie den Projektnamen mit der rechten Maustaste anklicken und den Menüpunkt *als Startprojekt festlegen* auswählen.

HINWEIS: Bitte verwechseln Sie diese Funktion nicht mit der Funktion *Startobjekt*, die Sie über die Eigenschaftenseite eines Projektes erreichen (Kontextmenü eines Projektes im Projektmappenexplorer), und mit deren Hilfe Sie bestimmen, welches Objekt innerhalb Ihres *Startprojektes* das *Startobjekt* sein soll.

Über das Kontextmenü einer Projektmappe oder eines Projektes erreichen Sie Funktionen, um ...

- ... die Projektmappe/das Projekt zu erstellen – dabei werden nur veränderte Dateien der Projektmappe/des Projektes neu kompiliert.
- ... einen Build – also das resultierende Kompilat eines Compilerdurchlaufs in Form von Assemblies, ausführbaren oder anderen Dateien – komplett zu bereinigen. Wenn Sie einen Build bereinigen, werden alle Zwischen- und Ausgabedateien gelöscht, sodass nur die Projekt- und Komponentendateien übrig bleiben. Anschließend können aus den Projekt- und Komponentendateien neue Instanzen der Zwischen- und Ausgabedateien erstellt werden.
- ... die Projektmappe/das Projekt *neu* zu erstellen – dabei werden alle Dateien der Projektmappe neu kompiliert.
- ... den Konfigurationsmanager für eine Projektmappe aufzurufen, um Abhängigkeiten, Prioritäten, Plattformeigenarten und Ähnliches festzulegen.
- ... ein neues oder vorhandenes Projekt oder Element der Projektmappe/dem Projekt hinzuzufügen.
- ... das Startprojekt der Projektmappe festzulegen.
- ... das Projekt im Debug-Modus oder im Einzelschrittmodus ablaufen zu lassen (dazu muss das Kontextmenü der Projektmappe ausgewählt worden sein).
- ... eine neue Instanz eines Projektes im Debug-Modus zu starten (dazu muss das Kontextmenü des Projektes ausgewählt worden sein).
- ... das (gesamte) Projekt zu speichern.
- ... die Projektmappe zur Quellcodeverwaltung (*Visual Source Safe*) hinzuzufügen.
- ... das Projekt/die Projektmappe umzubenennen.
- ... die Eigenschaften für die Projektmappe abzurufen.

Projektdateitypen

Die wichtigsten Projektdateitypen in Visual Basic sind (ASP.NET-Projekte einmal außen vor gelassen):





Symbol	Typ	Aufgabe
	Projektmappe (Solution)	Beinhaltet die Zusammenstellung der Projektmappendateien sowie die globalen Einstellungen der Projektmappe.
	Visual Basic-Projekt	Beinhaltet die Zusammenstellung eines Visual Basic-Projektes sowie die globalen Einstellungen für das Projekt.
	Formulardatei	Stellt die Quellcodedatei einer Klasse dar, die von <i>System.Windows.Forms</i> abgeleitet wurde, damit ein Formular verwaltet und mit dem Designer von Visual Studio .NET bearbeitet werden kann.
	Visual Basic-Klassendatei	Stellt die Quellcodedatei einer Visual Basic-Klasse, eines Moduls oder einer <i>AssemblyInfo</i> dar.

Tabelle 2.1: Die wichtigsten Projektdateien

TIPP: Wenn Sie per Doppelklick auf eine Formulardatei nicht den Designer, sondern direkt das entsprechende Codefenster anzeigen lassen möchten, öffnen Sie das Kontextmenü, indem Sie mit der rechten Maustaste auf die betroffene Datei klicken, und wählen anschließend den Eintrag *Öffnen mit*. Im Dialog, der jetzt gezeigt wird, wählen Sie *Microsoft Visual Basic-Editor* per Mausklick aus und klicken anschließend auf die Schaltfläche *Als Standard*.

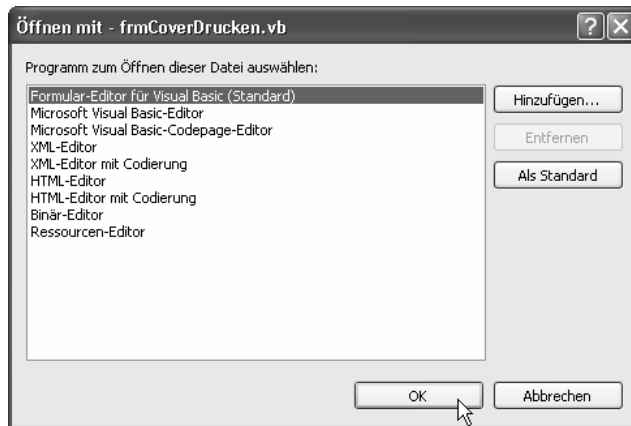



Abbildung 2.5: Mit diesem Dialog, den Sie durch *Öffnen mit* des Kontextmenüs einer Datei Ihres Projektes erhalten, bestimmen Sie, welcher Editor standardmäßig verwendet werden soll

Alle Projektdateien anzeigen

Einige Projektelemente eines Visual Basic-Projektes verfügen über mehrere Dateien, von denen standardmäßig nur die wichtigsten im Projektmappen-Explorer dargestellt werden. Dieses Ausblenden von für den Entwickler anfangs oft weniger wichtigen Projektdateien sieht man besonders schön bei Formular-Klassen. Im »Rohzustand«, wenn also nicht alle Dateien eines Projektes angezeigt werden, sieht man nach dem Erstellen eines neuen Formulars lediglich eine einzelne Klassendatei, die oben-drein noch völlig leer ist – gerade einmal die Klassendefinition ist dort zu finden.

»Unter der Haube« spielt sich jedoch schon eine ganze Menge ab, und »unter der Haube« meint in diesem Fall: Es gibt weitere Dateien, die zum Formular (und damit auch zum Projekt gehören).

Mit dem in Abbildung 2.3 unter  mit dem Pfeil gekennzeichneten Symbol können Sie alle Dateien eines Projektes ein- und ausblenden.

Weitere Funktionen des Projektmappen-Explorers

Die folgende Tabelle erklärt kurz die weiteren Funktionen, die sich durch die Symbole des Projektmappen-Explorers aufrufen lassen:







Symbol	Aufgabe
	Stellt die Eigenschaftenseite einer Projektmappe oder eines Projektes dar. HINWEIS: Anders als in Visual Basic.NET 2003 werden die Eigenschaften eines Projektes anschließend als Dokumentfenster in der aktuellen Registerkartengruppe für Dokumentfenster dargestellt und auch als solche behandelt. TIPP: Wenn Sie den Projektmappen-Explorer frei schwebend konfiguriert haben – beispielsweise um ihn auf einem zweiten Monitor darzustellen und so mehr Platz für Designer und Codeeditor zu haben – müssen Sie auf Grund einer »Unzulänglichkeit« die Eigenschaftenseite einer Projektmappe zweimal aufrufen, da sich beim ersten Aufruf die Selektion innerhalb des Projektmappen-Explorers verstellt. Dem Entwicklerteam ist dieser Bug bekannt – offensichtlich aus Kompatibilitätsgründen zur Visual Studio Extensibility kann er aber nicht ohne weiteres behoben werden. Der IntelliLink <i>B0201</i> verrät mehr.
	Schaltet um zwischen der eingeschränkten und der vollständigen Projektdateienanzeige. Bei der vollständigen Projektdateienanzeige sehen Sie ausnahmslos alle dem Projekt zugeordneten Dateien – beispielsweise wird der vom Designer automatisch erzeugte Code zur Erstellung eines Formulars dann unterhalb der eigentlichen Formulklassse eingeblendet. Die vollständige Anzeige erlaubt ebenfalls einen Blick in die von einem Projekt referenzierten Assemblies (die Sie benötigen, wenn Sie bestimmte Funktionalitäten des Framework oder eines anderen Projektes benötigen), die Sie anderenfalls nicht sehen würden.
	Aktualisiert die Projektdateienliste im Projektmappenexplorer.
	Ruft den Codeeditor auf und stellt die zuvor im Projektmappen-Explorer ausgewählte Codedatei dort dar.
	Ruft den Formular-Designer auf und stellt die zuvor im Projektmappen-Explorer ausgewählte Formulardatei dort dar.
	Falls Sie eine Klassendatei im Projektmappen-Explorer ausgewählt haben, gelangen Sie mit dieser Funktion zum visuellen Klassendesigner.

Tabelle 2.2: Die wichtigsten Projektdateien

Organisieren von Codedateien in Unterordnern

Visual Studio interessiert es nicht, ob eine Projektdatei, die Sie für ein Projekt benötigen, innerhalb des gleichen Verzeichnisses oder in einem Unterverzeichnis des Projektes liegt.



Abbildung 2.6: In umfangreichen Projekten helfen Unterverzeichnisse, dass die Übersicht über die vielen Codedateien eines Projektes nicht verloren geht

TIPP: Deswegen der Tipp: Machen Sie von Ordnern zur Organisation von Einheiten Ihres Projektes ruhig intensiv Gebrauch – Sie werden sich besser in größeren Projekten zurechtfinden. Legen Sie mithilfe des Projektmappen-Explorers Unterverzeichnisse an, in denen Sie die Codedateien zusammenfassen, die thematisch zusammengehören. Nutzen Sie dabei vor allem die Möglichkeiten von partiellen Klassen, mit deren Hilfe Sie den Code einer Klasse über mehrere Quellcodedateien verteilen können. ► Kapitel 6 erklärt mehr zum Thema partielle Klassen und dem Modifizierer `Partial`.

Abbildung 2.6 zeigt Ihnen, wie u. a. Klassen in Unterverzeichnissen in größeren Projekten zum leichteren Navigieren im Code organisiert sein können:

Dateioperationen innerhalb des Projektmappen-Explorers


Falls Sie übrigens einmal in die Lage kommen sollten, Code- oder sonstige Dateien kopieren, verschieben oder löschen zu müssen – dazu müssen Sie Visual Studio nicht verlassen. Zwar bietet Ihnen der Projektmappen-Explorer keine so komfortable Funktionalität wie der Windows-Explorer von XP mit Drag&Drop in Verbindung mit der rechten Maustaste und dem Kontextmenü oder gar der Windows Vista Explorer; aber Sie müssen Visual Studio immerhin nicht verlassen ...

Über das Kontextmenü der Codedatei eines Projektes können Sie die Funktionen *Verschieben* oder *Kopieren* abrufen. Nachdem Sie die Datei auf diese Weise markiert haben, klicken Sie entweder das Projekt oder einen Ordner innerhalb des gleichen oder eines anderen Projektes an – wiederum mit


der rechten Maustaste – und wählen aus dem Kontextmenü, das anschließend erscheint, *Einfügen*. Tastenkürzel funktionieren übrigens im Projektmappen-Explorer ähnlich gut wie im Windows-Explorer.

WICHTIG: Wenn Sie Formulare kopieren oder verschieben, sollten Sie ohnehin nicht den Windows-Explorer sondern den Projektmappen-Explorer verwenden. Nur mit ihm stellen Sie implizit sicher, dass Sie alle zum Formular dazugehörigen Dateien »erwischen«. Das gilt unter Umständen für andere Projektdateien wie beispielsweise typisierte Data-Sets ebenfalls.

Das Eigenschaftfenster

Mithilfe des Eigenschaftfensters – in Abbildung 2.3 unter  zu sehen – definieren Sie Eigenschaften von Elementen. Elemente in diesem Zusammenhang sind nicht nur Steuerelemente innerhalb des Designers – der Gültigkeitsbereich des Eigenschaftfensters erstreckt sich auf weite Bereiche der Visual Studio IDE. Natürlich werden Sie dieses Fenster in der Regel dann (und auch deswegen recht oft) verwenden, um Eigenschaften von Steuerelementen zu bestimmen, die Sie im Formular-Designer bearbeiten. Sie benötigen das Eigenschaftfenster allerdings auch, um Eigenschaften anderer Projektmappen-Elemente zu bestimmen: So legen Sie beispielsweise auch Parameter wie den Namen eines zu installierenden Programms, den Produkthersteller oder die Setup-Version Ihrer Anwendung in Setup- und Bereitstellungsprojekten mit dem Eigenschaftfenster fest.

Einrichten und Verwalten von Ereigniscode mit dem Eigenschaftfenster

Das Eigenschaftfenster verwenden Sie ebenfalls, um die einem mit dem Designer editierbaren Objekt zugeordneten Ereignisse visuell zu bearbeiten. Sie schalten mit den Symbolen, die durch den von Punkt  abgehenden Pfeil in Abbildung 2.3 markiert sind, zwischen den Ereignissen und den Eigenschaften eines Objektes um.

Ereignisbehandlungsroutinen für Steuerelemente werden im Code bereitgestellt. Klickt der Anwender beispielsweise auf eine Schaltfläche, wird – so vorhanden – der Code der Ereignisbehandlungsroutine für diese Schaltfläche ausgeführt.

Sie haben die Möglichkeit, diese Ereignisbehandlungsroutinen ausschließlich durch den Codeeditor festzulegen – sollten aber aus Gründen der Bequemlichkeit und der Schnelligkeit schon auf das Eigenschaftfenster zurückgreifen, denn Sie können es für mehrere Aufgaben in Sachen Ereignisse verwenden:

- Als Navigationshilfe: Sämtliche Ereignisbehandlungsroutinen des Objektes, die bereits definiert wurden, finden Sie in der Liste wieder. Ein Doppelklick auf das entsprechende Ereignis reicht aus, um zur entsprechenden Ereignisbehandlungsroutine im Codeeditor zu gelangen.
- Zum Erstellen einer Ereignisbehandlungsroutine mit einem Standardnamen: Dazu doppelklicken Sie einfach auf ein Ereignis; die IDE fügt anschließend automatisch einen entsprechenden Funktionsrumpf (eine so genannte »Stub«) in die Codedatei ein, und benennt diese als eine Kombination aus Ereignisnamen und Steuerelementnamen.
- Zum Erstellen einer benannten Ereignisbehandlungsroutine: Dazu geben Sie den gewünschten Funktionsnamen der Ereignisbehandlungsroutine neben dem Ereignisnamen ein und drücken an-

schließend **Eingabe**. Die IDE fügt anschließend automatisch einen entsprechenden Stub in die Code-Datei mit dem von Ihnen vorgegebenen Namen ein (siehe Abbildung 2.7).

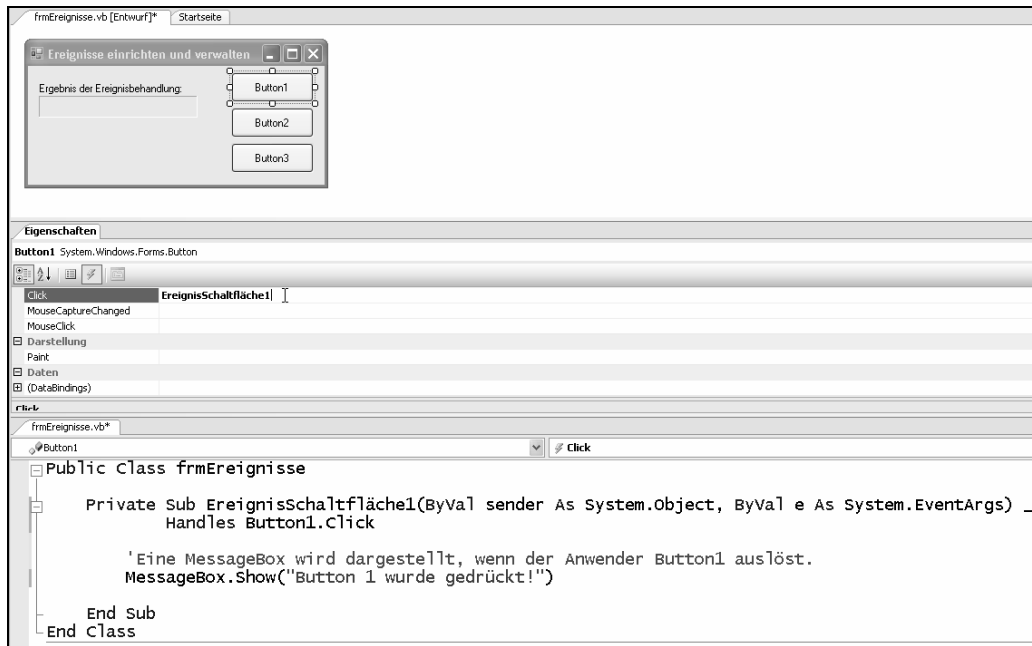


Abbildung 2.7: Der Anwender markiert die Schaltfläche mit dem Namen *Button1* (oberes Drittel) und gibt den Namen der Ereignisbehandlungsroutine (mittleres Drittel) ein. Die Behandlungsroutine trägt dann diesen Namen (*EreignisSchaltfläche1*) und behandelt (*Handles*) das entsprechende Ereignis (*Click*) des Objektes (*Button1*).

TIPP: Falls Sie sich wundern, wieso das Eigenschaftfenster in Abbildung 2.7 als Dokument erscheint. Im Abschnitt »Toolfenster« auf Seite 18 erfahren Sie, wie Sie Toolfenster als Dokument einer Dokumentenregisterkartengruppe hinzufügen können.

- Zum Zuweisen der gleichen Ereignisbehandlungsroutinen an mehrere Ereignisse. Anders als noch in Visual Basic 6.0 kann eine einzelne Ereignisbehandlungsroutine durchaus mehrere Ereignisse behandeln, wenn diese signaturkompatibel sind (also die gleichen Parameter beim Aufrufen zur Übergabe anbieten). In diesem Fall weisen Sie mit dem Eigenschaftfenster einem Ereignis eine Ereignisbehandlungsroutine zu: Klappen Sie die Aufklappliste neben dem entsprechenden Ereignis auf, und Sie sehen in der Liste die Namen der signaturkompatiblen Ereignisbehandlungsroutinen. Wählen Sie eine, die dieses Ereignis (ebenfalls) behandeln soll, aus der Liste aus (siehe Abbildung 2.11).

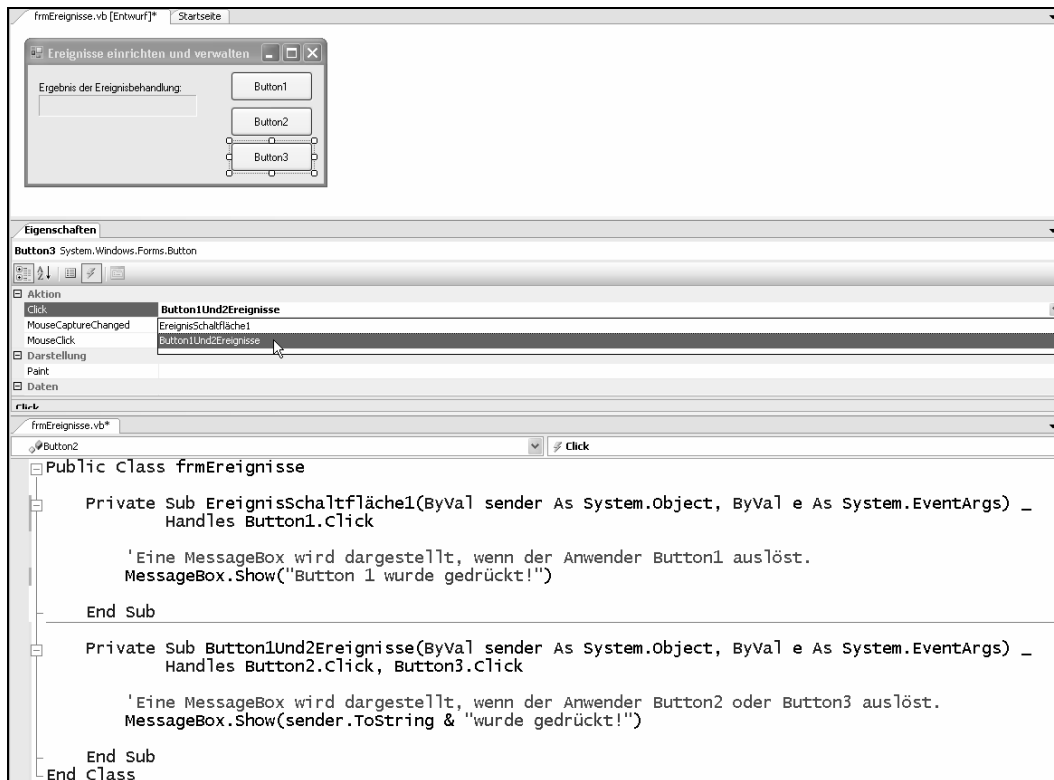


Abbildung 2.8: Der Anwender markiert die Schaltfläche mit dem Namen *Button3* (oberes Drittel) und wählt den Namen der Ereignisbehandlungsroutine (mittleres Drittel) aus. Die schon vorhandene Behandlungsroutine behandelt dann dieses Ereignis ebenfalls (unteres Drittel).

Die Fehlerliste

Visual Basic verfügt über einen so genannten Background-Compiler. Dieser Compiler läuft im Hintergrund, und er überprüft ständig Ihre vorgenommenen Änderungen und Ergänzungen des Programmcodes auf syntaktische Richtigkeit. Der Vorteil: Sie müssen nicht einen bei großen Projekten u.U. schon recht lange dauernden Compilerlauf anstoßen, um die Flüchtigkeitsfehler im Code zu finden – Sie sehen sie sofort, direkt nach der Eingabe einer neuen Zeile. Die Fehlerliste hilft Ihnen dabei, die Übersicht über Fehler im Programm nicht zu verlieren.

Sie sehen die Fehlerliste in Abbildung 2.3 mit ⑩ markiert (das linke Fenster). Die Fehler, die in der Fehlerliste erscheinen, haben übrigens eine direkte Verbindung zum Code (und der Codedatei), der den Fehler verursacht: Ein Doppelklick auf den Fehler bringt Sie an die Stelle im Codeeditor, an der die IDE den Fehler vermutet.

Die drei verschiedenen Meldungstypen der Fehlerliste

Es gibt übrigens drei verschiedene Meldungstypen in der Fehlerliste, deren Anzeige Sie nach Belieben konfigurieren können:

- **Fehler:** Richtige Fehler verhindern, dass eine Anwendung vor Behebung des Fehlers mit der zugrunde liegenden Codequelle überhaupt gestartet werden kann. Sie müssen den Fehler korrigieren, um das Programm (oder die Assembly) lauffähig zu machen.
- **Warnungen:** Eine Warnung ist ein Hinweis für Sie, dass etwas eigentlich nicht so läuft, wie es sollte; dieses Etwas ist allerdings nicht so schwerwiegend, dass es ein Funktionieren der Anwendungen völlig blockieren würde. Sie können den Schweregrad von Warnungen (welche Ereignisse führen zu Warnungen, welche Warnungen sollen wie Fehler behandelt werden) im Übrigen für jedes Projekt festlegen. Der folgende Abschnitt zeigt, wie es geht.
- **Meldungen:** Geben Ihnen zusätzliche Hinweise und Informationen, die aber in der Regel die Funktionsfähigkeit einer Anwendung nicht beeinflussen.

Welche der Kategorien angezeigt werden, bestimmen Sie übrigens mit den gleich lautenden Schaltflächen am oberen Rand des Fensters. Klicken Sie auf eine der Schaltflächen, um eine Kategorie auszublenden; klicken Sie abermals auf die Schaltfläche, um eine Kategorie wieder darstellen zu lassen.

Konfigurieren von Warnungen in den Projekteigenschaften

Welche Zustände Ihres Codes Warnungen oder Fehler generieren, lässt sich für jedes Projekt individuell einstellen. Anders als noch in Visual Studio 2003 werden die Eigenschaften eines Projektes als Dokumentfenster in der entsprechenden Registerkartengruppe dargestellt.

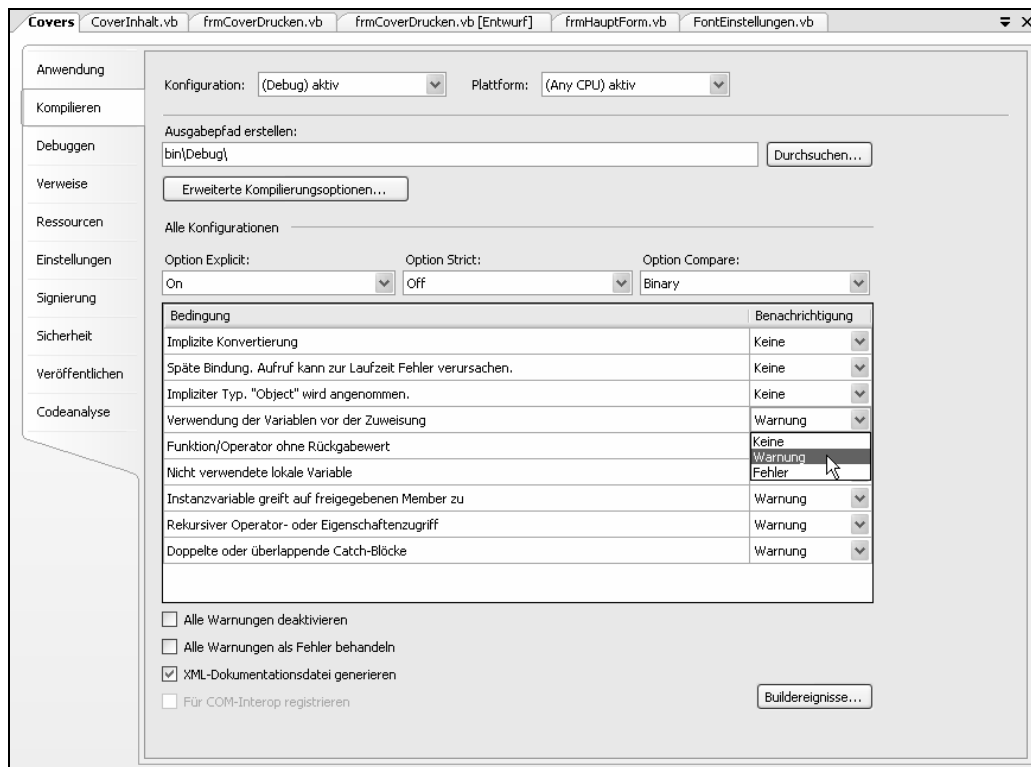


Abbildung 2.9: Mit dem Register *Kompilieren* stellen Sie das Fehlermeldungsverhalten eines Projektes ein

Um die Eigenschaften eines Projektes zu öffnen, rufen Sie das Kontextmenü des entsprechenden Projektes mit der rechten Maustaste auf und wählen anschließend *Eigenschaften*. Sie sehen anschließend einen Dialog, wie er etwa auch in Abbildung 2.9 zu sehen ist. Dort können Sie dann die verschiedenen Meldungstypen individuell konfigurieren.

Die Aufgabenliste

Die Aufgabenliste verhält sich in Visual Basic 2005 ähnlich wie die Fehlerliste, nur mit dem Unterschied, dass dort Punkte erscheinen, die Sie selber eintragen.

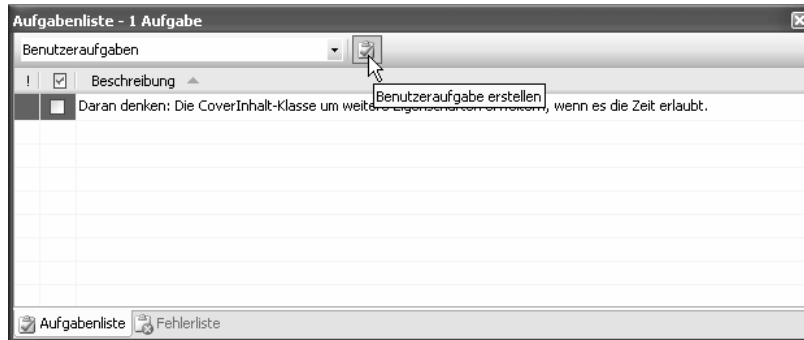


Abbildung 2.10: Die Aufgabenliste zeigt in Abhängigkeit der Auswahl in der Aufklappliste entweder benutzerdefinierte, manuell hinzugefügte Aufgaben oder durch Kommentare im Code eingefügte

Meldungen, die in der Aufgabenliste erscheinen, können durch zwei verschiedene Methoden dort hineingelangen.

- Durch das manuelle Hinzufügen einer Aufgabe zur Aufgabenliste. Dazu wählen Sie aus der Aufklappliste den Punkt *Benutzeraufgaben*, und klicken Sie anschließend auf das rechts daneben stehende Symbol, etwa wie in Abbildung 2.10 zu sehen.
- Durch Einfügen von Kommentaren innerhalb des Listings. In Abbildung 2.3 sehen Sie unter Punkt ⑤ einen Pfeil, der vom Codefenster in das Aufgabenfenster reicht. Sobald Sie einen Kommentar im Codelisting, wie dort zu sehen, einfügen, der mit bestimmten Schlüsselwörtern beginnt, sehen Sie diesen Kommentar in der Aufgabenliste – vorausgesetzt Sie haben zuvor in der Aufgabenliste aus der Aufklappliste den Punkt *Kommentare* gewählt.

Die Schlüsselwörter, auf die das Aufgabenfenster sozusagen »reagiert«, lassen sich übrigens nach Belieben konfigurieren. Wählen Sie dazu aus dem Menü *Extras* den Menüpunkt *Optionen*. Im Bereich *Umgebung/Aufgabenliste* konfigurieren Sie vorhandene Schlüsselwörter (so genannte »Tokens«) oder richten weitere ein. Abbildung 2.11 zeigt, wie es geht. Neben der Einrichtung der Schlüsselwörter können Sie auch deren Priorität festlegen. Dazu wählen Sie aus der Aufklappliste die *Priorität*, die eine Codezeile mit dem entsprechenden Token automatisch bekommen soll, wenn sie in die Aufgabenliste eingetragen wird. Hohe oder niedrige Prioritäten in der Liste werden anschließend mit einem entsprechenden Symbol zur besseren Wiedererkennung gekennzeichnet.

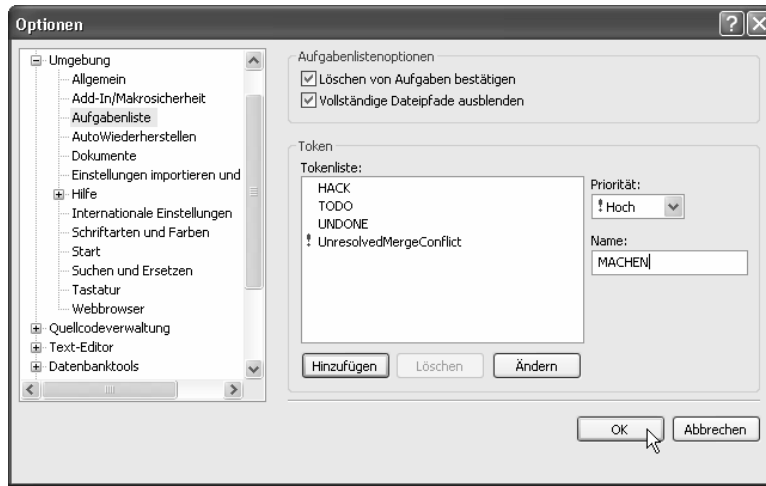


Abbildung 2.11: Zusätzliche Schlüsselwörter für die Aufnahme von Aufgaben in die Aufgabenliste durch Codekommentare lassen sich individuell konfigurieren

Tipps für die Aufgaben- und die Fehlerliste

Sowohl die Aufgaben- als auch die Fehlerliste enthalten Listenelemente, die Sie mithilfe der Spaltenköpfe sortieren können. Klicken Sie dazu auf einen Spaltenkopf, um die Liste nach der Spalte zu sortieren. Klicken Sie ein zweites Mal auf den gleichen Spaltenkopf, wird die Liste nach dieser Spalte in absteigender Reihenfolge sortiert.

Navigieren mit der Aufgaben- und der Fehlerliste

Mithilfe beider Fenster können Sie im Codeeditor navigieren. Möchten Sie zu einem in der Aufgabenliste ausgewiesenen Kommentar im Code gelangen, doppelklicken Sie einfach auf den entsprechenden Listeneintrag. Das gleiche gilt für Fehler, die Sie in der Fehlerliste sehen.

Das Ausgabefenster

Das Ausgabefenster in Visual Studio 2005 erfüllt zwei Funktionen. Zum einen gibt es während des Kompilierens eines Projektes oder einer Projektmappe die Meldungen des Compilers aus. Zum anderen zeigt es bestimmte Statusmeldungen während des Programmablaufs an oder erlaubt auch, dass Ihre eigenen Programme beispielsweise mit einer Anweisung wie `Debug.Print` eigene Ausgaben im Ausgabefenster vornehmen. Welche Ausgabentypen im Ausgabefenster dargestellt werden, bestimmen Sie mit der Aufklappliste *Ausgabe anzeigen von*.

Ausgabe anzeigen von *Erstellen*

Abbildung 2.12 zeigt Ihnen beispielhaft die Ausgabe eines Projekt-Builds, in dem (mit Absicht natürlich) zwei Compiler-Fehler aufgetreten sind.

```

Ausgabe
Ausgabe anzeigen von: Erstellen

----- Erstellen gestartet: Projekt: Covers, Konfiguration: Debug Any CPU -----
Der Buildvorgang wurde um 25.10.2005 14:33:20 gestartet.
CoreResGen-Ziel:
  Es gibt keine Ressourcen, die im Hinblick auf ihre Quelldateien veraltet sind.
  Die Ressourcengenerierung wird übersprungen.
CoreCompile-Ziel:
  C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Vbc.exe /noconfig /imports:
  Microsoft.VisualBasic,System.Collections,System.Collections.Generic,System.
  Data,System.Drawing,System.Diagnostics,System.Windows.Forms /nowarn:42016,41999,42017
  ,42018,42019,42032,42036,42020,42021,42022 /rootnamespace:Covers /doc:obj\Debug\
  Covers.xml /define:"CONFIG=\"Debug\",DEBUG=-1,TRACE=-1,_MyType=\"WindowsForms\",
  PLATFORM=\"AnyCPU\" " /reference:C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.
  Data.dll,C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Deployment.dll,C:\
  WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.dll,C:\WINDOWS\Microsoft.NET\
  Framework\v2.0.50727\System.Drawing.dll,C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727
  \System.Windows.Forms.dll,C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Xml.
  dll /main:Covers.My.MyApplication /debug+ /debug:full /out:obj\Debug\Covers.exe /
  resource:obj\Debug\Covers.frmCoverDrucken.resources /resource:obj\Debug\Covers.
  frmHauptForm.resources /resource:obj\Debug\Covers.Resources.resources /target:winexe
  CoverInhalt.vb FontEinstellungen.vb frmCoverDrucken.Designer.vb frmCoverDrucken.vb
  frmHauptForm.Designer.vb frmHauptForm.Designer.vb "My Project\AssemblyInfo.vb" "My Project\
  Application.Designer.vb" "My Project\Resources.Designer.vb" "My Project\Settings.
  Designer.vb"
D:\Dev.Net 2005\VBWhidbey\E - Ein- und Umstieg\Covers\frmCoverDrucken.vb(120) : error
  BC30451: Der Name "locOffset_X" wurde nicht deklariert.
D:\Dev.Net 2005\VBWhidbey\E - Ein- und Umstieg\Covers\frmCoverDrucken.vb(121) : error
  BC30451: Der Name "locOffset_Y" wurde nicht deklariert.

Fehler beim Buildvorgang.

Vergangene Zeit 00:00:00.07
===== Build: 0 erfolgreich oder aktuell, Fehler bei 1, 0 Übersprungen =====

```

Abbildung 2.12: Dieses Beispiel zeigt die Build-Protokollierung (*Ausgabe von Erstellen*) eines Visual Basic-Projektes mit normaler Ausführlichkeit

Mit den Symbolen neben der Aufklappliste am oberen Fensterrand haben Sie die Möglichkeit, bestimmte Funktionalitäten abzurufen, die Ihnen das Ausgabefenster zur Verfügung stellt. Die folgende Tabelle zeigt, welche Funktionen es gibt:

Symbol	Aufgabe
	Wenn Sie in das Ausgabefenster auf eine Compiler-Fehlermeldung klicken, können Sie dieses Symbol als Navigationshilfe verwenden und zur entsprechenden Zeile im Code springen, an der der Fehler aufgetreten ist. Alternativ bringt Sie ein Doppelklick auf die entsprechende Compiler-Fehlermeldung ebenfalls zur entsprechenden Quellcodetextstelle. Abbildung 2.3 verdeutlicht das auch unter Punkt ④.
	Falls es im Ausgabefenster mehr als eine Meldung wie beispielsweise Fehlermeldungen des Compilers gibt, können Sie mit diesem Symbol zur vorherigen Meldung navigieren.
	Falls es im Ausgabefenster mehr als eine Meldung wie beispielsweise Fehlermeldungen des Compilers gibt, können Sie mit diesem Symbol zur nächsten Meldung navigieren.
	Löscht den Inhalt des Ausgabefensters.
	Schaltet den Zeilenumbruch ein. In diesem Fall werden eigentlich einzeilige Meldungen in neue Zeilen umbrochen, falls der Platz im Fenster nicht ausreicht. Der horizontale Schieberegler wird dann natürlich nicht mehr benötigt und verschwindet.

Tabelle 2.3: Symbole, mit der Sie erweiterte Funktionalitäten des Ausgabefensters steuern

Ausgabe anzeigen von *Debuggen*

Diese Ausgaben stehen Ihnen nur zur Verfügung, wenn Ihr Projekt im Debug-Modus gestartet wird bzw. beispielsweise durch das Setzen eines Haltepunktes unterbrochen wurde. Abbildung 2.13 zeigt eine typische Ausgabe einer Windows Forms-Anwendung nach dem Start.

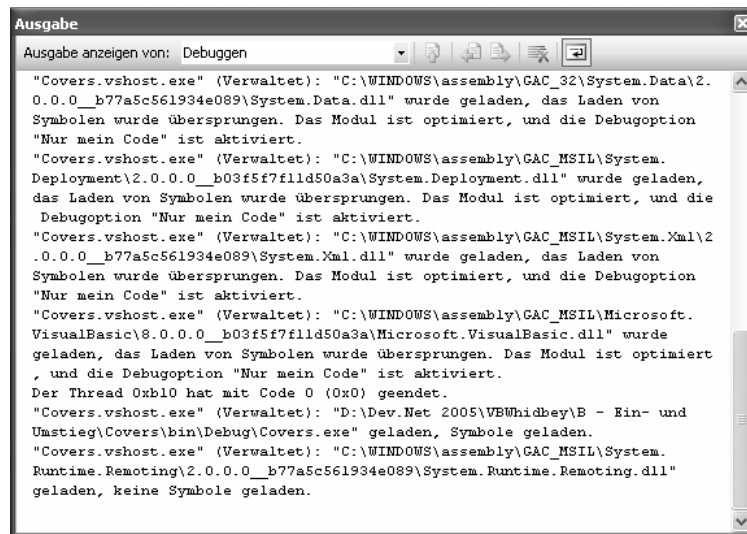


Abbildung 2.13: Dieses Beispiel Meldungen im Ausgabefenster (*Ausgabe von Debuggen*) eines Visual Basic-Projektes nach dem Start im Debug-Modus

Die dynamische Hilfe

Das Fenster »dynamische Hilfe« bietet Ihnen Schlagwörter zum aktuellen Kontext an, mit deren Hilfe Sie sich per Mausklick das entsprechende Thema in der Hilfe anzeigen lassen können. Abbildung 2.3 zeigt die Verbindung von aktuellem Kontext im Toolfenster (Punkt ⑥ zu Punkt ⑦) zur dynamischen Hilfe mit einem Pfeil. Diese Verbindungsherstellung funktioniert natürlich auch in anderen Zusammenhängen – sehr flexibel und stets brauchbar gerade dann, wenn Sie im Editor arbeiten.

TIPP: Der aktuelle Kontext spiegelt sich in der dynamischen Hilfe wider, wann immer Visual Studio ein Schlüsselwort oder einen Objektnamen im Visual Basic-Editor erkannt hat und sich der Cursor darauf befindet. Bei Ereignissen funktioniert das natürlich nur dann, wenn Sie sich mit dem Cursor im eigentlichen Ereignisnamen befinden – und der »eigentliche Ereignisname« befindet sich hinter dem `Handles`-Schlüsselwort.

Performance-Einbrüche im Editor verursacht durch die dynamische Hilfe

Wenn der Editor gegen Ende einer umfangreichen Quellcodedatei auf nicht so leistungsfähigen Maschinen zu langsam wird, könnte die dynamische Hilfe der Grund dafür sein. Schließen Sie das Fenster der dynamischen Hilfe einfach, und Sie können anschließend Ihre Quellcodedatei in der gewohnten Geschwindigkeit bearbeiten.

Anpassen des Inhalts der dynamischen Hilfe

Sie können den Inhalt der dynamischen Hilfe anpassen. Dazu rufen Sie mit *Extras* | *Optionen* den Optionsdialog von Visual Studio auf.

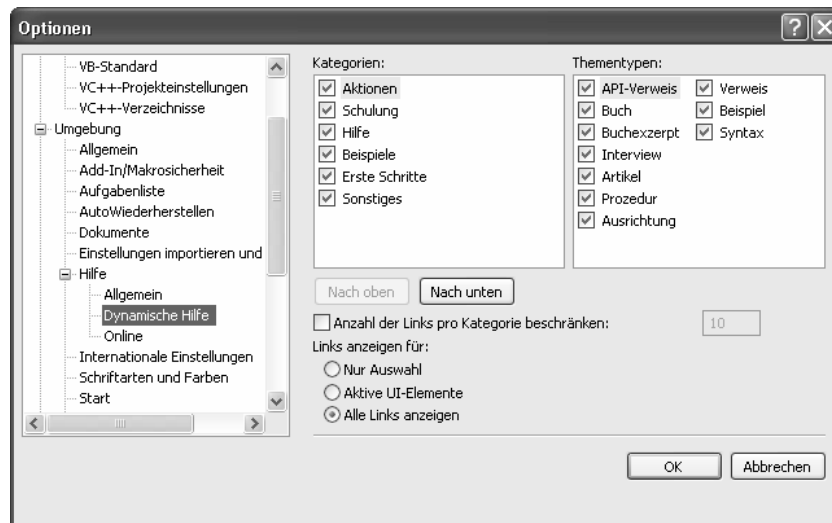



Abbildung 2.14: Hier konfigurieren Sie, wie umfangreich die dynamische Hilfe Hilfethemen finden soll


Unter *Umgebung/Hilfe/Dynamische Hilfe* finden Sie die Registerkarte (siehe Abbildung), mit der Sie die Themen bestimmen können, die im Kontext angezeigt werden sollen.

TIPP: Wenn Sie Performance-Einbußen im Editor feststellen, die durch die dynamische Hilfe verursacht werden (siehe auch vorheriger Absatz), Sie aber dennoch nicht komplett auf die dynamische Hilfe verzichten möchten, wählen Sie hier so wenig wie möglich Kategorien und Thementypen aus, und beschränken Sie sich auf das Wesentliche. Auf diese Weise erhöhen Sie die Performance in der Regel schon ausreichend, um zügig weiterarbeiten zu können.

Die Klassenansicht

Die Klassenansicht dient zur aufgeteilten Ansicht Ihres Projektes auf Klassenbasis und kann Ihnen auch bei der Navigation im Projekt behilflich sein. Die Klassenansicht teilt die Elemente eines Projektes hierarchisch ein – Sie müssen nur auf das jeweilige Pluszeichen vor einem Element klicken, um eine Ebene zu öffnen. Ein Doppelklick auf das entsprechende Element bringt Sie anschließend zur entsprechenden Definition im Quellcode. In Abbildung 2.3 finden Sie unter  ein Beispiel für das Klassenansicht-Toolfenster, das in diesem Fall als Dokumentfenster in einer Registerkartengruppe eingefügt wurde.

Codeeditor und Designer

Bei diesen Werkzeugen hat sich seit Visual Studio 2003 eine ganze Menge getan. So gibt es beispielsweise beim Designer eine ausgeklügelte Positionierungshilfe; der Codeeditor glänzt, – wie in Abbildung 2.3 unter  in der Vergrößerung zu sehen, mit zusätzlichen Orientierungshilfen zum Speicherzustand und anderen Infos. Diesen beiden wichtigen Werkzeugen sei daher ein eigenes – das anschließende – Kapitel gewidmet.

Die wichtigsten Tastenkombinationen auf einen Blick

Kurzbeschreibung	Tastenkomb.	Beschreibung	Befehlsname
Cursor zum nächsten Element	F8	Verschiebt den Cursor zum nächsten Element – beispielsweise einer Aufgabe oder einem Fehler im Aufgabenfenster.	Bearbeiten.GehezunächsterInstanz
Cursor zum vorherigen Element	Umschalt+F8	Verschiebt den Cursor zum vorherigen Element (Fehler, Aufgabe im Aufgabenfenster).	Bearbeiten. _ GehezuvorherigerInstanz
Cursor zur Definition	Umschalt+F12	Verschiebt den Cursor zur Definition des Elementes, das sich derzeit unterhalb des Cursors befindet.	Bearbeiten.GehezuVerweis
Rückgängig	Strg+Z oder Alt+Rücktaste	Macht die letzte Aktion rückgängig.	Bearbeiten.Rückgängig
Rückgängig rückgängig machen	Strg+Y	Stellt die zuvor rückgängig gemachte Aktion wieder her.	Bearbeiten.Wiederholen
Alles speichern	Strg+Umschalt+S	Speichert alle Dateien, an denen seit dem letzten Speichern Änderungen vorgenommen wurden.	Datei.AllesSpeichern
Zur letzten Änderung springen	Strg + –	Setzt den Cursor auf die Position im Code, an der Sie die letzte Änderung vorgenommen haben. Sie können nicht nur einen Schritt zurücknavigieren.	Ansicht.Rückwärtsnavigieren
Zur vorherigen Änderung springen	Strg+Umschalt + –	Nachdem Sie rückwärts navigiert haben (siehe vorherigen Eintrag), erreichen Sie damit wieder die ursprüngliche Position.	Ansicht.Vorwärtsnavigieren
Suchdialog öffnen	Strg+F	Öffnet den Suchen-Dialog.	Bearbeiten.Suchen
Inkrementelles Suchen	Strg+I	Startet das inkrementelle Suchen.	Bearbeiten. _ InkrementelleSuche
Zur Zeile mit Nummer springen	Strg+G	Öffnet den Dialog »Gehezu Zeilennummer« und erlaubt den Sprung zur Zeile mit angegebener Nummer. 	Bearbeiten.GeheZu

Kurzbeschreibung	Tastenkomb.	Beschreibung	Befehlsname
Automatischen Zeilenumbruch ein- und ausschalten	Strg+R, Strg+R	Schaltet den automatischen Zeilenumbruch ein und aus. Drücken Sie beide Tastenkombinationen dazu nacheinander.	Bearbeiten. _ Zeilenumbruchumschalten
Lesezeichen einfügen/entfernen	Strg+K, Strg+K	Schaltet ein Lesezeichen in einer Zeile ein bzw. wieder aus, wenn bereits eines gesetzt war.	Bearbeiten. _ Lesezeichenumschalten
Zum nächsten Lesezeichen	Strg+K, Strg+N	Setzt den Cursor in die Zeile, in der sich das nächste Lesezeichen befindet.	Bearbeiten. _ NächstesLesezeichen
Zum vorherigen Lesezeichen	Strg+K, Strg+P	Setzt den Cursor in die Zeile, in der sich das vorherige Lesezeichen befindet.	Bearbeiten. _ VorherigesLesezeichen
Aufgabeneintrag für aktuelle Codezeile hinzufügen/löschen	Strg+K, Strg+H	Fügt einen Aufgabeneintrag in die Aufgabenliste mit Verweis auf die aktuelle Zeile hinzu bzw. löscht den Eintrag wieder, wenn für die Zeile bereits einer vorhanden ist.	Bearbeiten. _ Aufgabenverknüpfung_ umschalten

Tabelle 2.4: Die wichtigsten Tastaturkommandos. Bitte beachten Sie, dass die Kommandos zur Tastaturanpassung in VB-Manier mit dem Unterstrich getrennt sind und eigentlich eine Zeile bilden

