

Kapitel 2

Das ist neu in der IDE von Visual Studio 2008

In diesem Kapitel:

Der erste Start von Visual Studio	14
Verbesserungen an der integrierten Entwicklungsumgebung (IDE)	16
Übernehmen von Visual Studio 2005-Einstellungen	18
Visual Studio 2005-Projekte nach Visual Studio 2008 migrieren	19
Designer für Windows Presentation Foundation-Projekte	20
IntelliSense-Verbesserungen	22
Inhalte von System- oder externen Typen in Vorschaufenstern fürs Debuggen konfigurieren	24
Zugriff auf den .NET Framework-Quellcode beim Debuggen	26

Der erste Start von Visual Studio

Auf den ersten Blick sind Unterschiede zum Vorgänger von Visual Studio 2008 kaum sichtbar. Den ersten Start vollziehen Sie genau so, wie Sie es beim Vorgänger gemacht haben. Visual Studio zeigt Ihnen einen Dialog, in dem Sie die Voreinstellungen der Benutzeroberfläche bestimmen können, etwa wie in Abbildung 2.1 zu sehen:



Abbildung 2.1 Beim ersten Start des Programms bestimmen Sie, wie die Entwicklungsumgebung voreingestellt werden soll

In diesem Dialog bestimmen Sie, mit welchen Voreinstellungen die Entwicklungsumgebung konfiguriert werden soll. Wenn Sie es gewohnt sind, mit Visual Studio 2002, 2003 und 2005 in der Standardeinstellung zu arbeiten, wählen Sie an dieser Stelle *Allgemeine Entwicklungseinstellungen* aus.

TIPP *Allgemeine Entwicklungseinstellungen* ist die Einstellung, mit der die meisten Visual Studio 2005- und 2008-Installationen voreingestellt werden. Visual Basic-Entwicklungseinstellungen enthalten spezielle Anpassungen, bei denen Fensterlayout, Befehlsmenüs und Tastenkombinationen mehr auf das schnelle Erreichen spezieller Visual Basic-Befehle angepasst sein sollen. Der Dialog zum Anlegen eines neuen Projektes ist beispielsweise nur auf Visual Basic-Projekte angepasst, um nicht zu sagen: beschnitten. Einige Optionen, wie das automatische Anlegen einer Projektmappe sind beim Anlegen eines neuen Projektes ausgeblendet – Sie haben die Möglichkeit, Projekte namenlos zu erstellen und erst am Ende der Entwicklungssitzung unter einem bestimmten Namen zu speichern. Das gilt auch für Befehle, die Sie über Dropdown-Menüs abrufen können. Im Ergebnis sind unter den Visual Basic-Einstellungen viele Funktionen, die auch Visual Basic-Projekte betreffen könnten, nicht verfügbar.

Probieren Sie die für Sie am besten geeignete Voreinstellung aus. Falls Sie mit der hier gewählten später nicht mehr zufrieden sind, beherzigen Sie das im folgenden Absatz Gesagte.

Zurücksetzen der IDE-Einstellungen

Wenn Sie die gesamte IDE in ihren Ausgangszustand zurückversetzen möchten, wählen Sie aus dem Menü *Extras* den Menüpunkt *Einstellungen importieren und exportieren*. Visual Studio zeigt Ihnen anschließend den Dialog, den Sie auch in Abbildung 2.2 sehen können.

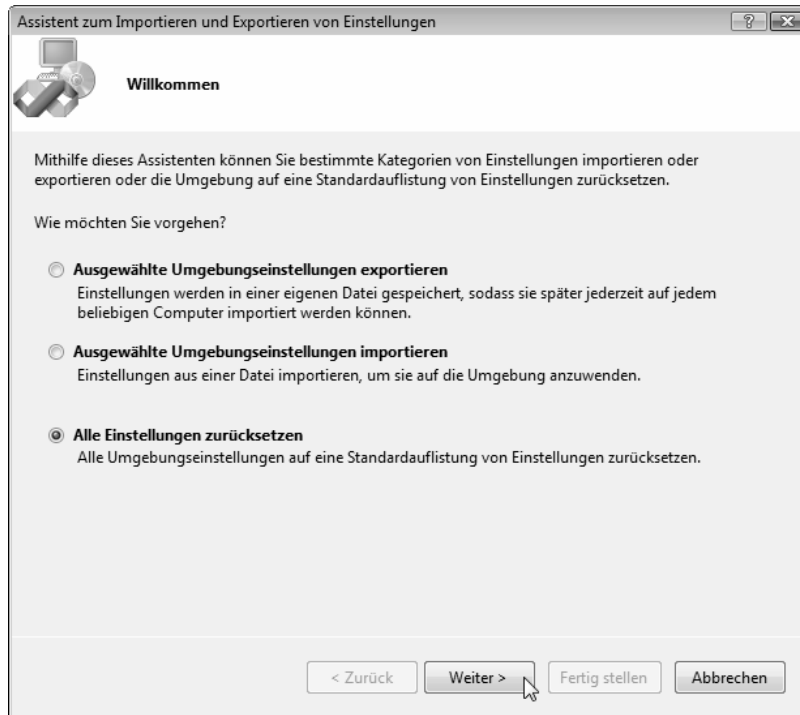


Abbildung 2.2 Um mit diesem Dialog die IDE-Einstellungen in den Ursprungszustand zurückzusetzen, wählen Sie aus dem Menü *Extras* den Menüpunkt *Importieren und Exportieren von Einstellungen*

HINWEIS Beachten Sie, dass der Vorgang, den Sie hier durchführen, die gesamte Entwicklungsumgebung in den Ausgangszustand zurücksetzt. Das bedeutet auch, dass bestimmte Schriftartenzuordnungen oder Tastaturkürzel, die Sie umdefiniert haben, dabei verloren gehen. Falls Sie nur das Fensterlayout von Visual Studio in den Originalzustand zurücksetzen wollen, wählen Sie aus dem Menü *Fenster* den Menüpunkt *Fensterlayout zurücksetzen*.

Klicken Sie auf *Weiter* und im Dialog, der jetzt erscheint, wählen Sie *Nein, nur Einstellungen zurücksetzen und die aktuellen Einstellungen überschreiben*. Der Dialog, der anschließend zu sehen ist, entspricht wieder weitestgehend dem in Abbildung 2.1. Wählen Sie Ihre bevorzugten Einstellungen und bestätigen Sie den Dialog mit *Fertig stellen*.

Verbesserungen an der integrierten Entwicklungsumgebung (IDE)

Die IDE wurde an vielen Punkten überarbeitet und modernisiert.

Einfacheres Positionieren und Andocken von Toolfenstern

Das Verschieben und Andocken von Toolfenstern ist leichter geworden; Das Verschieben und Andocken von Toolfenstern ist leichter geworden. Beim Ziehen mit der Maus wird ein so genanntes Diamant-Führungssymbol angezeigt. Indem Sie bei gedrückter Maustaste den Mauszeiger auf eines der Pfeilsymbole führen und dann loslassen, können Sie bequem die Zielposition bestimmen (siehe Abbildung 2.3).

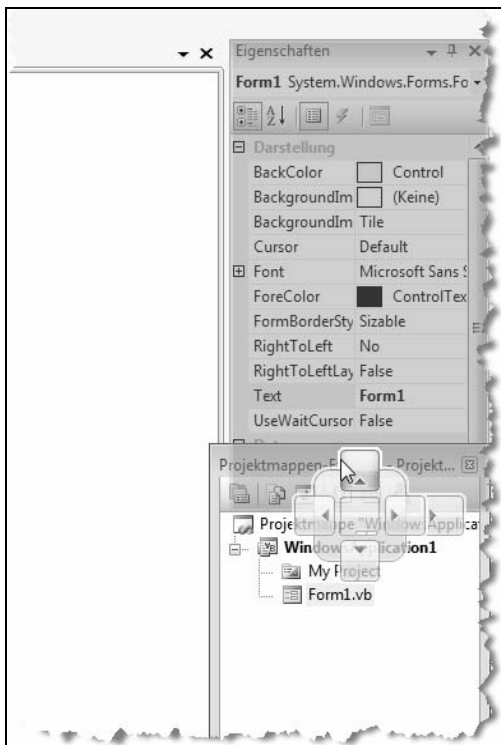



Abbildung 2.3 Mithilfe des Diamant-Führungssymbols lassen sich Fenster schnell und bequem positionieren

Navigieren durch Dateien und Toolfenster mit dem IDE-Navigator

Um im Codeeditor zu einer bestimmten geöffneten Datei zu navigieren, egal wann diese zuletzt verwendet wurde, können Sie den IDE-Navigator verwenden. Die Funktionsweise des IDE-Navigators ist in etwa dieselbe, wie die des Task-Switchers in Windows selbst, den Sie mit der Tastenkombination **Alt** 

bedienen. Der IDE-Navigator ist nicht über Menüs verfügbar. Sie können ihn über die Tastenkombinationen **Strg** **↩** bzw. **Strg** **↪** bedienen – je nachdem, in welcher Reihenfolge Sie durch die Dateien navigieren möchten (siehe Abbildung 2.4).

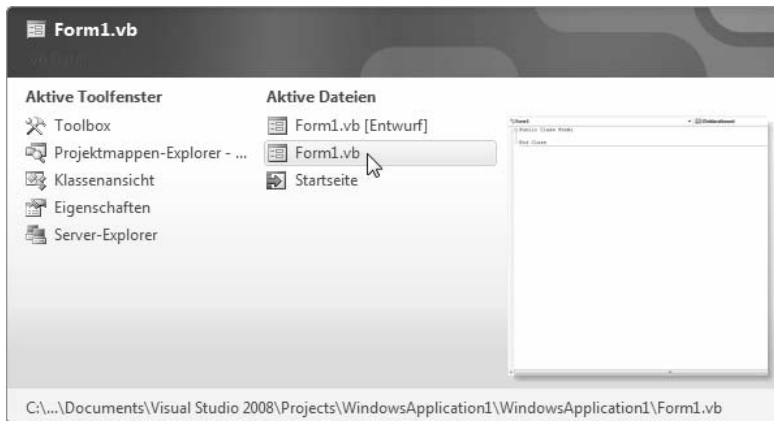


Abbildung 2.4 Den IDE-Navigator rufen Sie mit **Strg** **↩** oder **Strg** **↪** ins Leben

Drücken Sie bei gehaltener **Strg**-Taste – für die umgekehrte Richtung halten Sie zusätzlich die **↵**-Taste gedrückt) – die Tabulatortaste so oft **↩**, bis die gewünschte Datei ausgewählt ist oder klicken Sie diese direkt mit der Maus an.

TIPP

Alternativ klicken Sie in der oberen rechten Ecke des Editors auf die Schaltfläche *Aktive Dateien* neben der Schaltfläche *Schließen*, und wählen Sie die gewünschte Datei aus der Liste aus.

Mit dem IDE-Navigator können Sie auch zwischen den Toolfenstern navigieren, die in der IDE geöffnet sind. Je nachdem, in welcher Reihenfolge Sie navigieren möchten, können Sie die Tastenkombinationen **Alt** **F7** oder **Alt** **↵** **F7** verwenden.

Umgebungsschriftart definieren

Für nicht näher bestimmte IDE-Elemente konnte man bislang keine Schriftart definieren, was insbesondere bei der Entwicklung auf Monitoren größer als 24 Zoll Probleme machte. Wählen Sie aus dem Menü *Extras* den Menüpunkt *Optionen*, erweitern Sie den Knoten *Umgebung* und wählen Sie *Schriftarten und Farben*. In der Klappliste unter *Einstellungen anzeigen für*: wählen Sie die Option *Umgebungsschriftart*. Anschließend bestimmten Sie eine neue Schriftart und den entsprechenden Schriftgrad.

Betroffen von der Umgebungsschriftart sind alle Schriften, die Sie nicht extra festlegen können, also Drop-down-Menüschriften, Dialogschriften, Projektmappen-Explorer, Toolbox, Registerkartenbeschriftungen, wie beispielsweise in Abbildung 2.5 zu sehen.

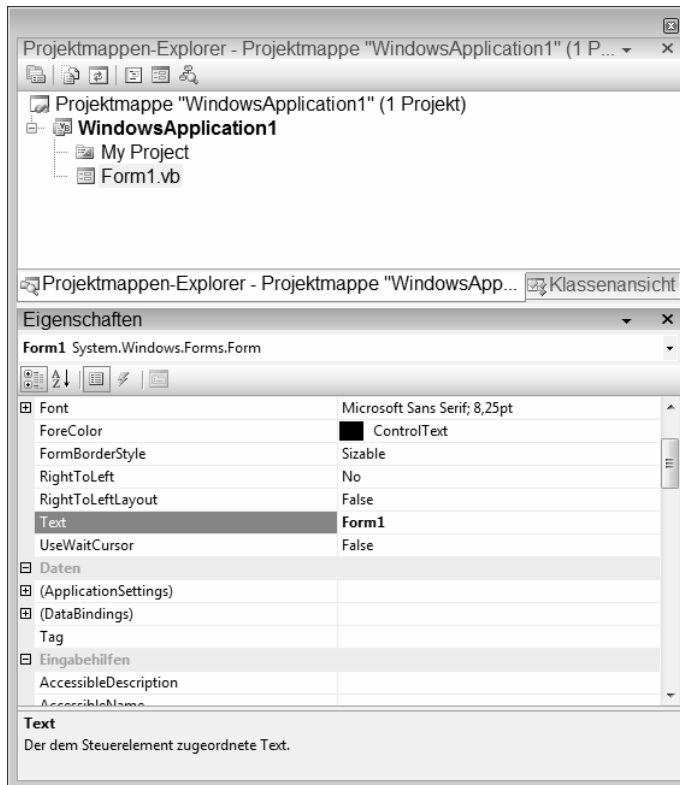


Abbildung 2.5 Mit der Umgebungsschriftart lässt sich die Basisschriftart der IDE verändern; das hat Auswirkungen auf verschiedene Toolfenster, Dialoge und Dropdown-Menüs – wie hier im Bild am Beispiel des im Projektmappen-Explorers gezeigt

Übernehmen von Visual Studio 2005-Einstellungen

Wenn Visual Studio 2005 und Visual Studio 2008 auf demselben Computer installiert sind, können Sie beim ersten Starten von Visual Studio 2008 die meisten Einstellungen von Visual Studio 2005 übernehmen. Codeausschnitte (auch bekannt unter dem Namen Code Snippets) können hingegen nicht *automatisch* übernommen werden und müssen für die Verwendung in Visual Studio 2008 manuell neu installiert werden. Wenn Visual Studio 2005 und Visual Studio 2008 nicht auf demselben Computer installiert sind, können Sie Ihre Visual Studio 2005-Einstellungen immer noch manuell für die Verwendung in Visual Studio 2008 migrieren. In diesem Fall exportieren Sie die Einstellungen in eine Datei, indem Sie aus dem Menü *Extras* den Menüpunkt *Einstellungen importieren und exportieren* wählen. Visual Studio zeigt Ihnen wieder den Dialog, den Sie in Abbildung 2.2 sehen können. Wählen Sie die Option *Ausgewählte Umgebungseinstellungen importieren* und im Dialog, der anschließend erscheint, die Einstellungsdatei aus und folgen Sie den weiteren Anweisungen.

Visual Studio 2005-Projekte nach Visual Studio 2008 migrieren

Grundsätzlich hat Visual Studio 2008 einen anderen Projektdateien- und Projektmappendateiaufbau als Visual Studio 2005. Das heißt erstmal: Ein Projekt, das Sie mit Visual Basic 2005 (oder einer noch früheren Version) erstellt haben, können Sie nicht direkt in Visual Studio 2008 öffnen. Visual Basic 2005 und Visual Studio 2008 können aber auf ein und demselben Rechner problemlos koexistieren, und wenn Visual Studio 2005 zusammen mit der 2008er Version auf einem Rechner installiert ist, haben Sie folgende Optionen:

- Sobald Sie eine Projekt- oder Projektmappe im Explorer doppelklicken, starten Sie damit automatisch die Instanz der »richtigen« (also der dem Projekt zugehörigen) Version von Visual Studio. Geregelt wird das dadurch, dass die entsprechende Dateiendung nicht mehr einer bestimmten Visual Studio-Version sondern einem so genannten *Visual Studio Version Selector* zugewiesen ist, der sich die aufzurufende Projektdatei anschaut, analysiert und dann entscheidet, welche Version von Visual Studio gestartet werden soll.



Abbildung 2.6 Nach dem Öffnen eines Visual Basic 2005-Projektes startet sofort der Migrationsassistent, mit dem Sie das Projekt in das neue Projektformat konvertieren können

- Wenn Sie eine Projekt- oder eine Projektmappe in Visual Studio 2008 öffnen, die in Visual Studio 2005 erstellt wurde, wird der Migrationsassistent aktiv, der es Ihnen ermöglicht, Ihr vorhandenes Projekt in das neue Format zu migrieren (siehe Abbildung 2.6).

HINWEIS

Nach der Konvertierung eines Projektes zielt das in das Visual Basic 2008-Format konvertierte Projekt auf das .NET Framework 2.0 – Sie haben hier also weiterhin »nur« die Möglichkeit, alte .NET 2.0-Funktionalität in Ihrem Projekt weiterzuverwenden, aber dafür ist das Kompilat Ihres Projektes auch in der Lage, ältere Windows 2000-Clients¹ zu bedienen, die mit den neueren .NET Framework-Versionen nicht ausgestattet werden können.

Visual Basic 2008 erlaubt es jedoch, auch gegen unterschiedliche .NET Framework-Versionen zu entwickeln – der nächste Absatz hält dazu genauere Informationen bereit. Möchten Sie Ihr konvertiertes Projekt umstellen, sodass Sie auch die Funktionalitäten der neueren 3.0 bzw. 3.5 .NET Framework-Versionen nutzen können, ändern Sie die Projekteinstellungen, wie im ersten Abschnitt des nächsten Kapitels beschrieben.

Designer für Windows Presentation Foundation-Projekte

Mit dem Windows Presentation Foundation (WPF)-Designer können Sie WPF-Anwendungen und benutzerdefinierte Steuerelemente in der IDE erstellen. Der WPF-Designer kombiniert die Echtzeitbearbeitung der XAML (Extended Application Markup Language) mit einer verbesserten grafischen Entwurfszeiterfahrung. Ein neues WPF-Projekt erstellen Sie, indem Sie

1. aus dem Menü *Datei* den Befehl *Neu/Projekt* auswählen
2. im Dialog, der jetzt erscheint, unter *Projekttypen* den Eintrag *Windows* wählen und unter *Vorlagen* den Eintrag *EPF-Anwendung*.
3. unter *Name* einen neuen Namen für Ihr Projekt bestimmen, ebenso den Speicherort und anschließend auf *OK* klicken, um das neue WPF-Projekt bearbeiten zu können.

Der WPF-Designer unterscheidet sich vom Windows-Forms-Designer grundlegend. Anders als beim Windows-Forms-Designer kann der Aufbau eines WPF-Forms in einer so genannten XAML-Datei gespeichert werden – der Aufbau eines WPF-Forms nur durch reinen Code wie beim Windows-Forms-Designer ist natürlich ebenfalls möglich; die Trennung zwischen Code und Aufbaubeschreibung macht es jedoch möglich, dass ein beauftragter Designer unabhängig vom Entwickler der Formularlogik das Formular designen kann.

Dem Thema WPF ist ein eigenes Kapitel in diesem Buch gewidmet (siehe Kapitel 11). Die hier beschriebenen Features und Vorgehensweisen sollen nur einen kurzen und groben Überblick über die Neuheiten aus IDE-Sicht bieten.

Die folgenden Features sind im WPF-Designer neu.

- Mithilfe der SplitView-Funktionalität können Sie Objekte im grafischen Designer anpassen und die Änderungen des zugrunde liegenden XAML-Codes direkt anzeigen lassen. Entsprechend werden Änderungen am XAML-Code sofort im grafischen Designer umgesetzt. Abbildung 2.7 vermittelt Ihnen einen Eindruck davon.

¹ Oder, mit eingeschränkter Funktionalität, sogar alte Windows 98-Clients – doch das sei wirklich nur der Vollständigkeit halber erwähnt, denn schon aus Sicherheitsaspekten sollten Sie Ihren Kunden gegenüber fairerweise gar nicht mehr erwähnen, dass Ihre Software theoretisch auch unter Windows 98 lauffähig ist.

HINWEIS

Da der WPF-Designer lange nicht über den gleichen Funktionsumfang wie der Windows Forms-Designer verfügt, ist dieses erste Feature auch gleichzeitig sein bestes: In vielen Fällen ist es leichter, den für den Formularaufbau erforderlichen XAML-Code zu erstellen, als zu versuchen, beispielsweise Steuerelementkombinationen mit dem Designer interaktiv zusammenzuklicken. Aus diesem Grund werden wir später, ab dem 11. Kapitel, das Augenmerk auch mehr auf XAML an sich als auf den Umgang mit dem Designer legen.

Eine viel bessere Alternative zum eingebauten WPF-Designer ist das Tool *Expression Blend*, mit dem Sie sogar WPF-konform interaktiv Animationen erstellen können – ein Feature, das der eingebaute Designer überhaupt nicht unterstützt. Mehr zum Thema Expression Blend erfahren Sie unter <http://www.microsoft.com/expression/>.

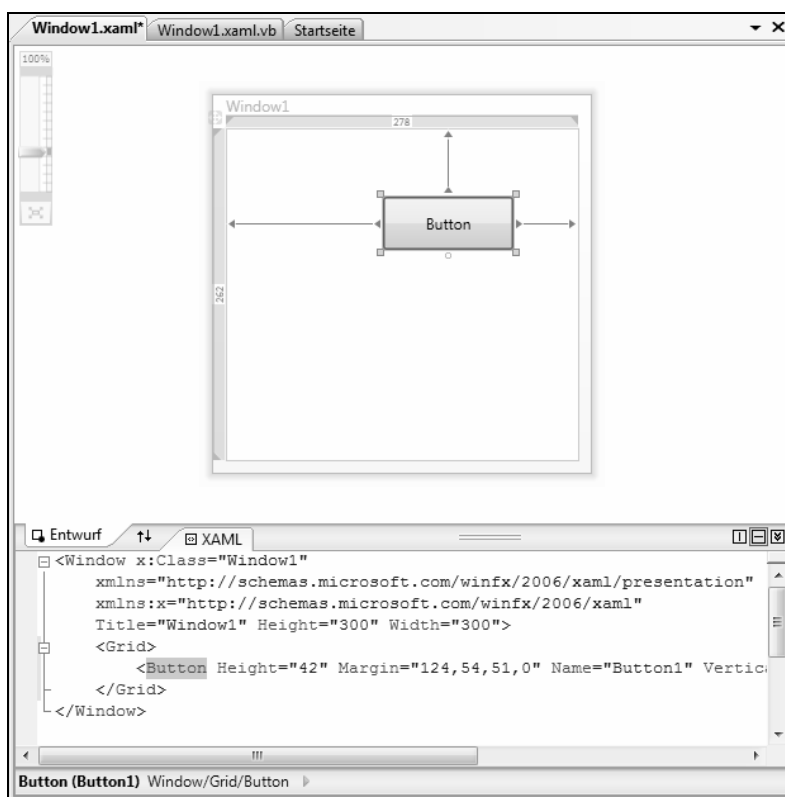


Abbildung 2.7 Der WPF-Designer bietet Split-View-Funktionalität: Grafische Änderungen werden direkt in XAML-Code umgesetzt, Änderungen im XAML-Code spiegeln sich direkt in der grafischen Repräsentation wieder.

- Im Fenster *Dokumentgliederung* können Sie den XAML-Code bei vollständiger Auswahlsynchronisierung zwischen Designer, Dokumentgliederung, XAML-Editor und Eigenschaftfenster anzeigen lassen und darin navigieren.
- IntelliSense im XAML-Editor ermöglicht den schnellen Codeeintrag. IntelliSense unterstützt jetzt selbstdefinierte Typen.

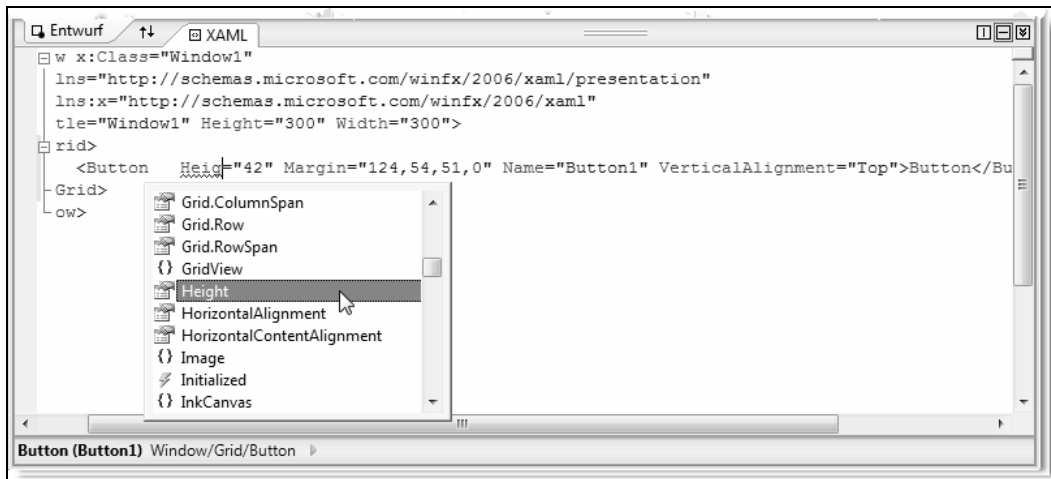


Abbildung 2.8 Der XAML-Editor verfügt über volle IntelliSense-Unterstützung

- Rasterlinien können den Rastern im Designer hinzugefügt werden, um die einfache rasterbasierte Platzierung von Steuerelementen zu ermöglichen.
- Steuerelemente und Text können leicht an Ausrichtungslinien ausgerichtet werden.
- Der Designer unterstützt jetzt das Laden der von Ihnen definierten Typen. Dazu gehören benutzerdefinierte Steuerelemente und Benutzersteuerelemente.
- Sie können das Laden großer XAML-Dateien abbrechen.
- Die Entwurfszeiterweiterung unterstützt Entwurfsmodus und Eigenschaften-Editoren.

Dem Thema WPF ist ein eigener Buchteil gewidmet, der sich auch ein wenig näher mit dem WPF-Designer auseinandersetzt. Ab Kapitel 11 finden Sie Näheres zu diesem Thema.

IntelliSense-Verbesserungen

Das Visual Basic-Team hat die Unterstützung des Entwicklers beim Coden durch IntelliSense in Visual Studio 2008 in verschiedenen Punkten verbessert.

Syntax-Tooltips

Am auffälligsten ist die Einführung sogenannter Syntax-Tooltips, wie Sie sie vielleicht schon kennen, sollten Sie nicht nur in Visual Basic sondern auch schon in C# 2005 entwickelt haben. In Visual Basic 2008 zeigt die Entwicklungsumgebung das IntelliSense-Fenster bereits an, wenn Sie die ersten Buchstaben gedrückt haben (siehe folgende Abbildungen).

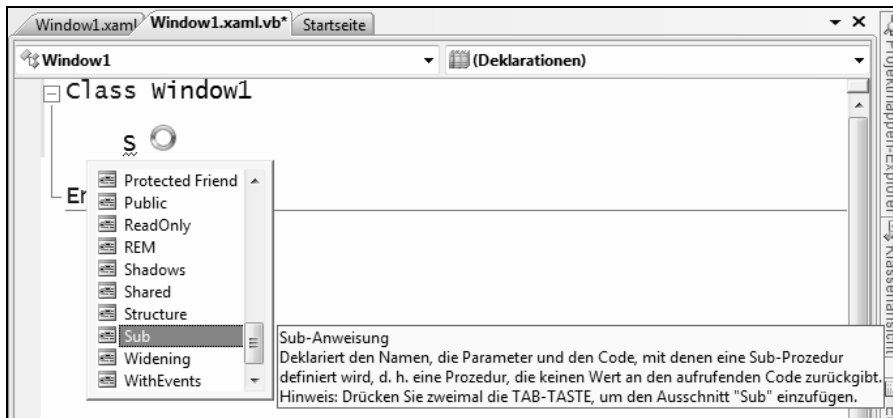


Abbildung 2.9 IntelliSense wird bereits aktiv, wenn Sie den ersten Buchstaben in das Code-Fenster eingeben und unterstützt Sie nicht nur mit der Auflistung bestimmter Schlüsselworte ...

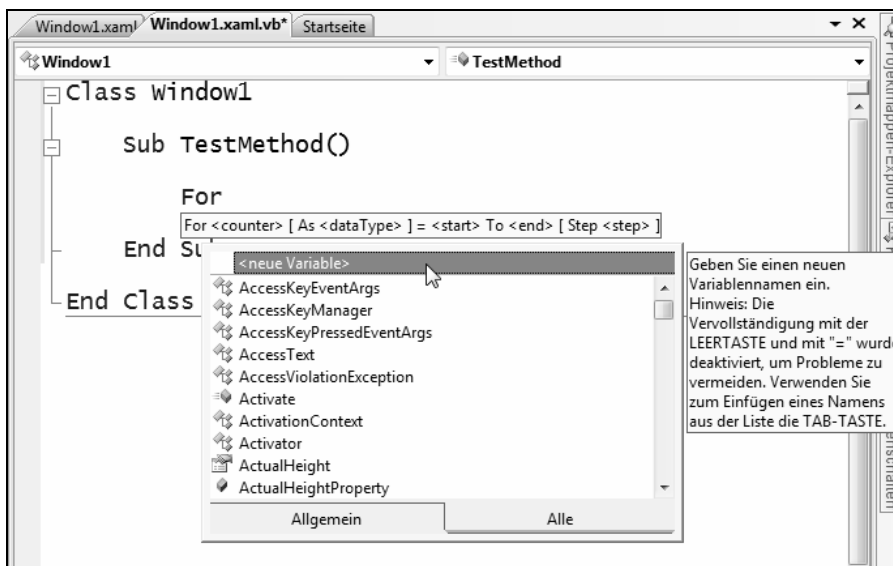


Abbildung 2.10 ... sondern zeigt in ausführlichen Tooltips auch die genaue Syntax und Funktionsweise von Strukturbefehlen an

TIPP In vorherigen Versionen störten die IntelliSense-Listen und Tooltips ab und an, da man nicht durch sie hindurch sehen konnte. Das wird mit Visual Studio 2008 anders: Drücken Sie einfach `[Strg]`, während die IntelliSense-Elemente angezeigt werden, und sie geben die Sicht auf den darunterliegenden Code frei!

Filtern beim Tippen

In vorherigen Versionen wurde man oftmals durch die langen IntelliSense-Listen erschlagen. In der 2008er Version werden die Inhalte der IntelliSense-Listen beim Tippen gefiltert, wie in den folgenden beiden Abbildungen zu sehen:

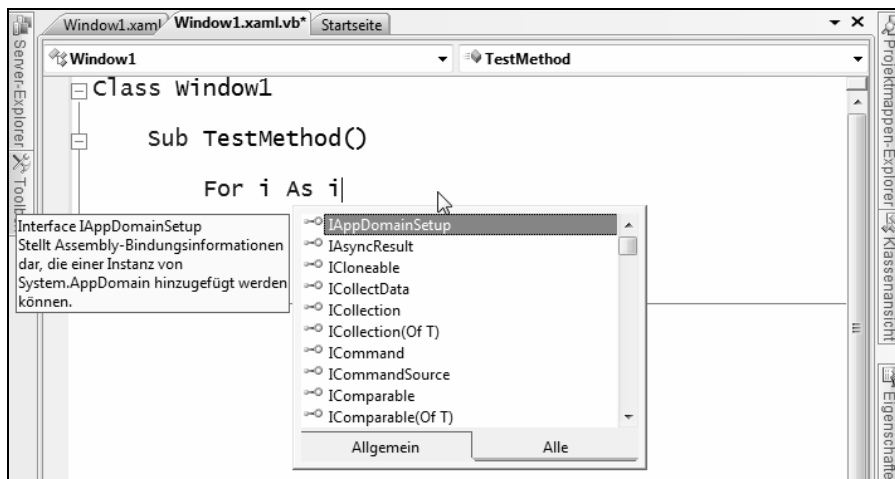


Abbildung 2.11 Die IntelliSense-Auswahlliste wird durch die bislang eingegebenen Zeichen gefiltert. Mit jedem weiteren getippten Buchstaben ...

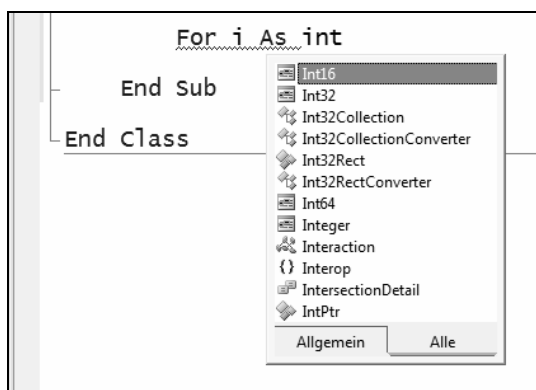


Abbildung 2.12 ...verkleinert sich die Liste auf die Elemente, die dem bislang Getippten entsprechen.

Inhalte von System- oder externen Typen in Vorschaufenstern fürs Debuggen konfigurieren

Beim Debuggen von Anwendungen war die vollständige Konfiguration der Wertentsprechungen von Systemtypen oder von Typen, zu denen Sie den Quellcode nicht besitzen, in Variablen-Vorschaufenstern (anders als beispielsweise in C#) nicht implementiert. In Visual Basic 2008 wurde dieses Manko ausgemerzt und die Unterstützung für `DebuggerDisplayAttribute` komplettiert.

Worum geht's genau? Ganz vereinfacht ausgedrückt um die Darstellung des eigentlichen Wertes eines Typs beim Debuggen in einem Vorschaufenster, etwa wie in der folgenden Abbildung zu sehen:

BEGLEITDATEIEN

Unter `.\Samples\Chapter02\DebugExtTypes` finden Sie das Beispielprojekt dieses Abschnittes.

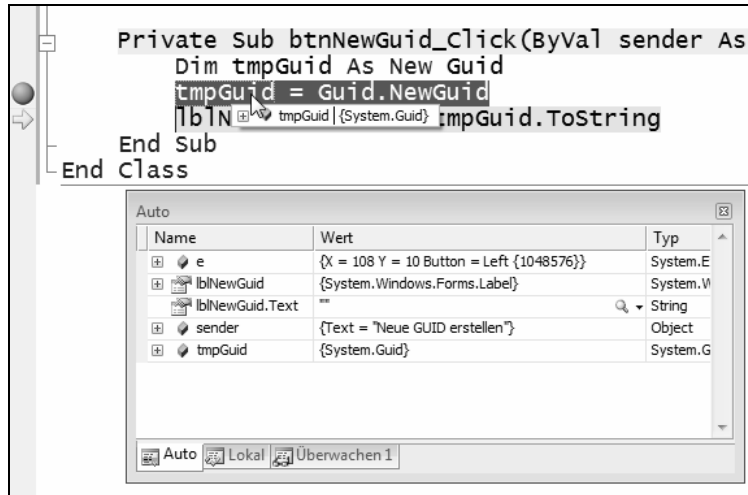


Abbildung 2.13 Bei bestimmten Typen ist keine Entsprechung bei der Ausgabe seines Inhalts als Text definiert – stattdessen wird, wie hier bei Guid, der nicht sehr informative Typname ausgegeben

- Um dem Abhilfe zu leisten, definieren Sie eine neue Assembly in Form einer Klassenbibliothek. Wählen Sie dazu aus dem Kontextmenü des Projektmappen-Explorers über die Projektmappe (nicht das Projekt!) den Menüpunkt *Hinzufügen/Neues Projekt*.
- Wählen Sie unter Projekttypen *Visual Basic/Windows* und unter Vorlagen *Klassenbibliothek* aus, bestimmen Sie Projektnamen (beispielsweise *CustomDebugDisplays*) und Speicherort, und klicken Sie auf *OK*. Falls nur das Projekt, nicht aber die Projektmappe im Projektmappen-Explorer zu sehen ist, wählen Sie aus dem Menü *Datei* den Befehl *Neu/Projekt* aus, um an den entsprechenden Dialog zu gelangen.
- Klicken Sie im Projektmappen-Explorer das neu hinzugefügte Projekt an, und wählen Sie das Projektmappen-Explorer-Symbol mit der Tooltipp-Beschreibung *Alle Dateien anzeigen* aus. Öffnen Sie anschließend den Zweig *My Projekt*.
- Doppelklicken Sie auf *AssemblyInfo.vb*, um die Assembly-Infos im Editor anzeigen zu lassen.
- Fügen Sie über das *Assembly*-Attribut die entsprechenden *DebuggerDisplay*-Attribute hinzu, etwa:

```
<Assembly: DebuggerDisplay("{ToString}", Target:=GetType(Guid))>
```

- Mit dieser Anweisung bestimmen Sie die *ToString*-Methode (erstes Argument) zur Anzeige des *Guid*-Typs (zweites Argument).
- Erstellen Sie die Projektmappe neu, und kopieren Sie die neue Assembly in das Verzeichnis *Visual Studio 2008/Visualizers*, das Sie im *Eigene Dokumente*-Verzeichnis Ihres Windows-Laufwerkes finden. Sie finden die Assembly im Unterverzeichnis *.\bin\debug* des Projektverzeichnisses.

Beim erneuten Debuggen finden Sie die korrekte Darstellung des *Guid*-Typs in allen Vorschaufenstern:

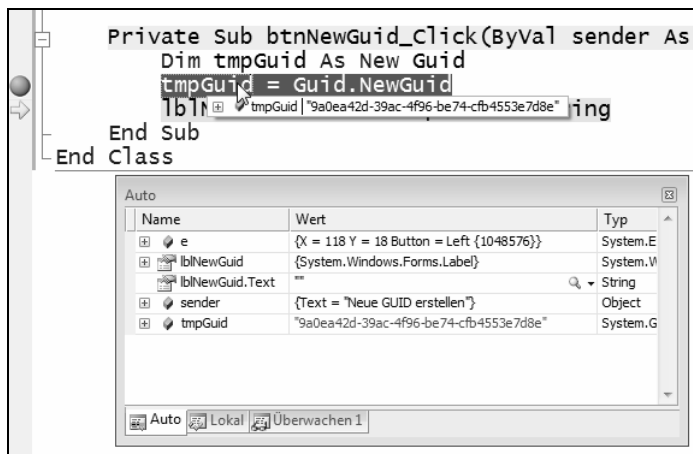


Abbildung 2.14 Nachdem die neue Assembly in *Visual Basic 2008/Visualizers* unterhalb des *Eigene Dokumente*-Ordners kopiert wurde, wird der Inhalt einer Guid-Instanz in den Vorschaufenstern korrekt dargestellt.

Zugriff auf den .NET Framework-Quellcode beim Debuggen

Dieser Abschnitt beschäftigt sich mit einem Thema, dessen Ankündigung in der Presse schon für eine kleine Sensation sorgte. Die simple und dazu umgekehrt proportional Aufmerksamkeit erregende Meldung lautete: »Microsoft legt .NET-Quelltexte offen²«. Das gelesen, stellten sich viele enthusiastische C#-Entwickler dann vor, »Oh wie cool – dann lade ich die Quelltexte herunter und bastele mir mein eigenes .NET Framework³«. Doch ganz so einfach hat es uns Microsoft nicht gemacht, an die begehrten Quellen des Frameworks zu kommen.

Zunächst einmal benötigen Sie mindestens Visual Studio 2008 in der Standardversion. Ältere Visual Studio-Versionen und auch sämtliche Express-Versionen bleiben außen vor. Und dann können Sie die Quelltexte der jeweiligen Komponenten, die Sie interessieren, nur dann bekommen, wenn Sie innerhalb von Visual Studio in diese hinein debuggen – und zu nichts anderem sind die Quellcodes bzw. die Freigabe derselben auch gedacht: Nämlich um Sie beim Debuggen innerhalb Ihrer eigenen Projekte zu unterstützen.

Und für uns Visual Basic-Entwickler gibt es eventuell noch eine kleine Einschränkung, und ich möchte mich hier bewusst nicht weiter auf das Vorwort dieses Buches beziehen: Der verwaltete Teil des .NET Frameworks ist natürlich zu ganz erheblichen Teilen in C# geschrieben worden. Zu welchen genau entzieht sich meiner Kenntnis, aber müsste ich schätzen, dann würde ich sagen: 95% sind in C#, der Rest (beispielsweise die *Microsoft.VisualBasic.dll*-Assembly) in Visual Basic. Das wiederum bedeutet: Sie sehen den Quelltext natürlich auch nur in der Sprache, in der er entwickelt wurde, und das ist eben in den meisten Fällen C#.

² So der Heise-Ticker unter www.heise.de am 4.10.2007 um 11:15. Unter <http://www.heise.de/newsticker/meldung/96909> können Sie diese Meldung direkt abrufen (Stand: 18.1.2008).

³ Kleine Anekdote am Rande: Hier bei ActiveDevelop sorgte die Meldung auch für lustiges Tohuwabohu, nachdem die *schon* ein wenig sarkastische Diskussion über den Nutzen des Frameworks »entglitt« und unser Chefentwickler sich der Realisierung seiner Weltübernahmepläne mithilfe einer eigenen Framework-Version einen Schritt näher wähnte... ;-))

Doch auch bevor das geschehen kann, müssen Sie – Stand 18.01.2008 – noch ein paar Vorbereitungen treffen, um in den Genuss des NET Framework-Debuggings zu gelangen. Wie es genau geht, zeigt die folgende Schritt-für-Schritt-Anleitung.

- Zunächst benötigen Sie ein Projekt, das Sie debuggen möchten. In guter alter Fernsehköchemanier haben wir da mal was vorbereitet – und was könnte es anderes sein, als die 132-gazillionste Version von »Hello World«!

BEGLEITDATEIEN

Das Projekt, mit dem wir das Debuggen des .NET Framework-Quellcodes demonstrieren wollen, befindet sich in `.\Samples\Chapter02 - NeuInIde\FrameworkDebugging`



Abbildung 2.15 Diese spektakuläre Windows-Anwendung wird das Debuggen von .NET Framework-Code demonstrieren: Entdecken Sie, was wirklich passiert, wenn Sie die Schaltfläche drücken!

- Und bevor wir uns daran machen können, herauszufinden, was wirklich passiert, wenn wir einem Label-Steuererelement eine neue Zeichenkette zuweisen ...

```
'Mehr iss nich!
Public Class Form1

    Private Sub btnShowHelloWorld_Click(ByVal sender As System.Object, _
                                        ByVal e As System.EventArgs) _
        Handles btnShowHelloWorld.Click

        'Text der Schaltfläche setzen
        lblHelloWorld.Text = "Hallo Welt!"

    End Sub
End Class
```

- ... müssen wir noch einige Einstellungen an der Benutzeroberfläche von Visual Studio vornehmen.⁴ Dazu wählen Sie in der IDE von Visual Studio aus dem Menü *Extras* den Menüpunkt *Optionen* aus.
- Im Dialog, der anschließend erscheint, wählen Sie in der linken Liste den Eintrag *Debugging*.

⁴ Shawn Burke bloggt den Tipp, zuvor einen Patch zu installieren. Zu finden war dieser Patch – Windows-Live ID vorausgesetzt – unter <https://connect.microsoft.com/VisualStudio/Downloads/DownloadDetails.aspx?DownloadID=10443&wa=wsignin1.0>. Shawn Burke ist Entwickler bei Microsoft; sein Blog gibt's unter <http://blogs.msdn.com/sburke/default.aspx> (Stand: 18.1.2008)

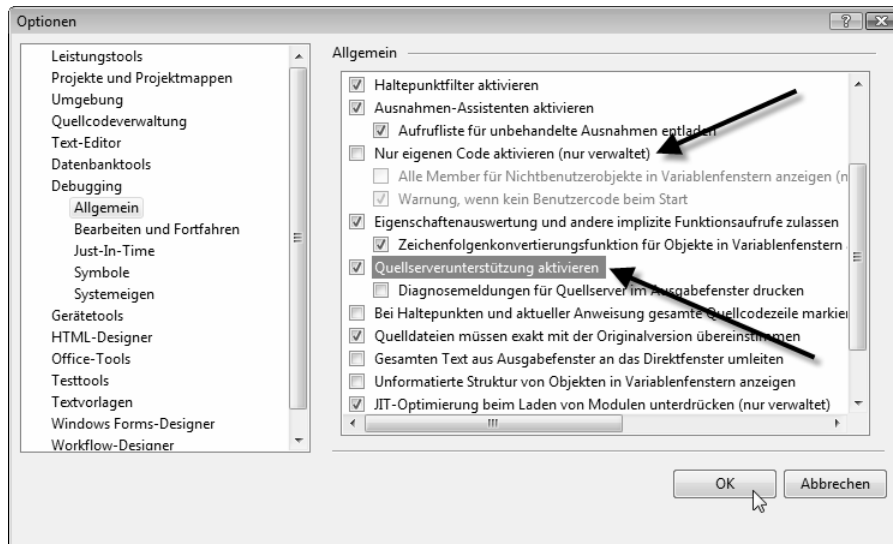


Abbildung 2.16 Deselektieren Sie die Option *Nur eigenen Code aktivieren (nur verwaltet)* und aktivieren Sie die Option *Quellserverunterstützung aktivieren*

- Auf der rechten Seite des Dialogs werden nun die allgemeinen Debugging-Einstellungen angezeigt. Hier deselektieren Sie die Option *Nur eigenen Code aktivieren (nur verwaltet)* und aktivieren die Option *Quellserverunterstützung aktivieren*.
- Wechseln Sie in der linken Spalte desselben Dialogs auf den *Debugging*-Untereintrag *Symbole*. Dort klicken Sie auf die *Neu*-Schaltfläche (das Symbol, auf das die Maus in Abbildung 2.17 zeigt), und geben Sie den Microsoft-Symboldateienserver an: <http://referencesource.microsoft.com/symbols>.⁵
- Unter dem Punkt *Symbole vom Symbolserver in diesem Verzeichnis zwischenspeichern* geben Sie anschließend einen Speicherort auf Ihrer Festplatte an, an dem die Quellcodedateien zwischengespeichert werden. Das Zwischenspeichern hat den Vorteil, dass die Dateien nur beim ersten Mal vom Microsoft-Server aus dem Internet geladen werden müssen – schon beim zweiten Quellcode-Debuggen fällt dieser Schritt weg, und die Quellcode-Dateien werden viel schneller aus diesem Cache-Speicher entnommen.
- Wenn Sie den Dialog nun mit *OK* bestätigen, sehen Sie einen EULA, den Sie ebenfalls nach natürlichem Studium mit *OK* bestätigen müssen – und dann dauert es ein kleines Weilchen, bis das Nächste passiert, da Visual Studio sofort anfängt, benötigte Symboldateien für das Beispielprojekt herunterzuladen. In der Statuszeile von Visual Studio können Sie beobachten, was passiert.

⁵ Stand: 18.01.2008 – dieser Eintrag könnte sich, wenn auch mit nicht großer Wahrscheinlichkeit, ändern – das sollte dann aber ausreichend früh von Microsoft kommuniziert werden.

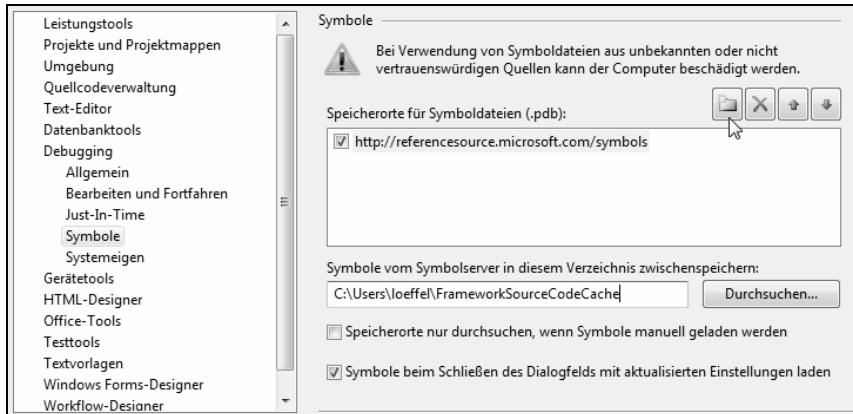


Abbildung 2.17 Im Symbole-Dialog geben Sie die Quelle der Symboldateien an, die auch den eigentlichen Quellcode des .NET Frameworks enthalten sowie ein Verzeichnis, in dem die Quellcode-Dateien zwischengespeichert werden können

- Den anschließenden Sicherheitshinweis bestätigen Sie ebenfalls mit *Ja* (natürlich auch nicht, bevor Sie ihn aufmerksam gelesen haben!).
- Und damit haben Sie quasi die Voraussetzungen für das Debuggen gelegt. Um das Debuggen nun zu starten, setzen Sie mit **[F9]** einen Haltepunkt – am besten an der Stelle, an der im Programmcode die Textzuweisung von »Hallo Welt!« an das Label erfolgt:

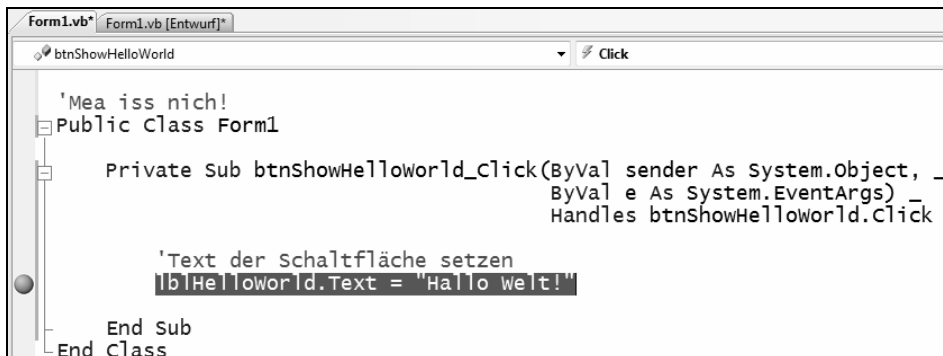


Abbildung 2.18 Setzen Sie in der Zeile den Haltepunkt, die die Ausgangsbasis für den Sprung ins .NET Framework sein soll

- Starten Sie anschließend Debuggen mit **[F5]**.
- Wenn das Programm gestartet ist, klicken Sie auf die Schaltfläche *Zeig "Hallo Welt"*.
- Das Programm trifft auf die entsprechende Haltepunktzeile, und die Programmausführung wird an dieser Stelle unterbrochen.
- Wählen Sie jetzt aus dem Menü *Debuggen*, den Menüpunkt *Fenster* und weiter *Aufrufliste*. Alternativ drücken Sie **[Strg] [Alt] [C]**.
- Sollten die Einträge für die Assembly *System.Windows.Forms*, anders als in der folgenden Abbildung zu sehen, ausgegraut sein, öffnen Sie das Kontext-Menü mit der rechten Maustaste und wählen den Menüpunkt *Symbole laden*.

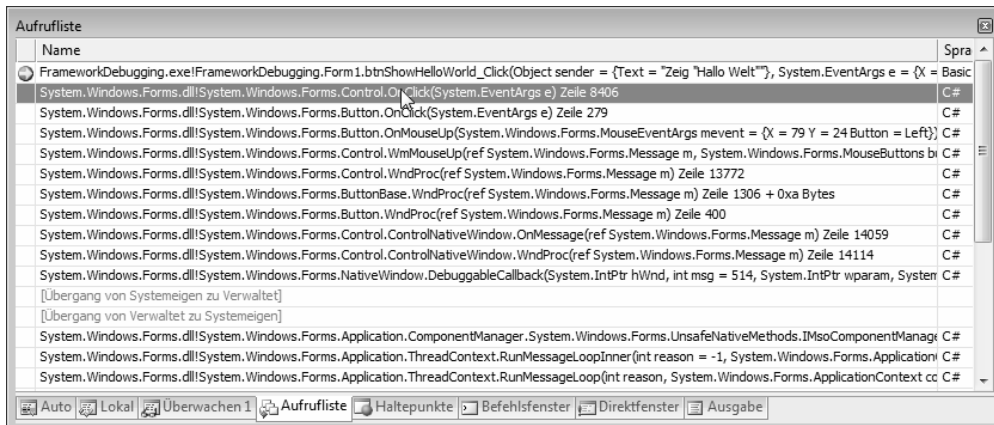


Abbildung 2.19 Im Unterschied zu »sonst«, sollten die .NET Framework-Assembly-Methoden und Eigenschaften in der Aufrufliste nicht ausgegraut sein, als Zeichen dafür, dass Sie in sie hineindebuggen können. Falls doch: Symbole über das Kontextmenü laden!

- Wenn Sie anschließend mit **F11** in die Routine hineinstepsen, landen Sie tatsächlich im .NET Framework-Quell-Code, wie Abbildung 2.20 zeigt.



Abbildung 2.20 Voila! – Der Source-Code des .NET Frameworks, natürlich nur ein ganz, ganz kleiner Ausschnitt (hier: Label.cs – der Sourcecode des Label-Controls)