

Kapitel 5

Visual Studio 2010

In diesem Kapitel:

Einleitung	169
Versionsgeschichte	169
Erweiterungen für Visual Studio 2010	169
Grundfunktionen	170
Neuerungen in der Entwicklungsumgebung Visual Studio in Version 2010	171
Neuerungen in der Entwicklungsumgebung Visual Studio in Version 2010 mit Productivity Power Tools	172
Produktvarianten	172
WPF-Oberfläche	177
Fensterverwaltung	177
Programmiersprachen	182
Projektverwaltung	183
Projektkonvertierung (Migration)	197
Code-Editoren	200
Grafische Editoren (Designer)	215
Weitere Fenster	220
Modellierung	225 ►

In diesem Kapitel (*Fortsetzung*):

Architektur- und Codeanalyse	228
Arbeit mit Datenbanken	231
Kompilierung und Ausführung	241
Debugger	243
Testen	255
Verbreiten von Anwendungen (Deployment)	275
Erstellen von MSI-Paketen	276
Nutzung des Team Foundation Server (TFS)	291
Team Explorer und Source Control Explorer	293
Erweiterbarkeit	299

Einleitung

Visual Studio ist die primäre Entwicklungsumgebung für .NET-Anwendungen. In den letzten Jahren mausert sich Visual Studio von einer Entwicklungsumgebung im engeren Sinne zu einem Application Life Cycle Management (ALM)-Werkzeug, das auch Modellierung, Projektverwaltung und Testen abdeckt. Dieses Kapitel fokussiert aber auf die Kernfunktionen: Editor/Designer, Debugger, Compiler.

HINWEIS Es gibt verschiedene Varianten von Visual Studio. Dieses Kapitel beschreibt Visual Studio 2010 anhand der größten (und teuersten) Variante, Visual Studio 2010 Ultimate. Wenn Sie eine kleinere Variante benutzen, werden Sie einige Funktionen nicht finden. Bitte nutzen Sie die Tabelle im Abschnitt »Produktvarianten« um festzustellen, welche Funktionen in Ihrer Variante vorhanden sind oder welche Variante Sie erwerben möchten.

Microsoft Press ist es aus wirtschaftlichen Gründen nicht möglich, vier verschiedene Varianten dieses Buchs zu drucken. Somit bleibt für dieses Buch leider nur die Alternative, die vollständige Version zu beschreiben.

Dieses Buchkapitel ist aber nicht von dem Microsoft Team Foundation Server (TFS) abhängig. Der TFS wird nur am Rande erwähnt.

Versionsgeschichte

Die folgende Liste zeigt die zum Redaktionsschluss verfügbaren Versionen von Visual Studio 2005, 2008 und 2010.

Produktname	Interne Versionsnummer	Erscheinungstermin
Visual Studio 2005	8.0.50727.42	7.11.2005
Visual Studio 2005 Service Pack 1	8.0.50727.762	14.12.2006
Visual Studio 2008	9.0.21002.8	19.11.2007
Visual Studio 2008 Service Pack 1 (Beta)	9.0.30428.1	9.5.2008
Visual Studio 2010	10.0.30319.1	12.4.2010

Tabelle 5.1 Verfügbare Versionen

Erweiterungen für Visual Studio 2010

Bis zum Redaktionsschluss dieses Buchs (und leider zum Teil erst kurz davor) sind vier Erweiterungen für Visual Studio 2010 erschienen, die in diesem Buch ansatzweise berücksichtigt werden konnten:

- Visual Studio 2010 Pex Power Tools [nur über MSDN-Subscriber-Downloads]
- Entity Designer Database Generation Power Pack [VSGall01] siehe Kapitel 12 »Objektrelationales Mapping (ORM) mit dem ADO.NET Entity Framework 4.0«
- Visual Studio 2010 Productivity Power Tools [<http://visualstudiogallery.msdn.microsoft.com/en-us/d0d33361-18e2-46c0-8ff2-4adea1e34fef>]
- Visualization and Modeling Feature Pack [nur über MSDN-Subscriber-Downloads]

Grundfunktionen

Die Entwicklungsumgebung Visual Studio bietet für alle Anwendungsarten folgende Grundfunktionen:

- Projektverwaltungssystem mit Projekten, die Codedateien und sonstige Dokumente enthalten, und Projektmappen, die Projekte enthalten
- Vorlagen für Projekte und Projektelemente (abhängig von der gewählten .NET Framework-Version)
- Einbinden von COM- und .NET-Softwarekomponenten in kompilierter Form
- Unterstützung für spezielle Funktionen von .NET- und Windows (z.B. Signierung von Assemblys, Assembly-Metadaten)
- Code-Editoren mit Eingabeunterstützung für VB, C#, J# und C++ sowie XML-, XSD-, XSLT-, .NET-Konfigurationsdateien (*.config*-Dateien), Windows Script Host-XML-Dateien (*.wsf*)
- Vordefinierte Codefragmente in einer Codefragmentbibliothek (Code Snippets)
- Programmcode-Refactoring
- Editor mit Syntaxfarbhervorhebung für JScript und VBScript esc
- Einfacher Texteditor für übrige Textformate
- Grafische Editoren für Windows Forms, Windows Presentation Foundation (WPF), Webforms/HTML-Dokumente, Bitmaps (*.bmp*, *.cur*, *.ico*), Klassendiagramme, XML-Ressourcendateien (*.resx*), XML-Schemata (*.xsd*), Workflows, typisierte DataSets und Objektrelationales Mapping (ORM)
- Erstellung und Veränderung von Datenbanken
- Grafische Berichterstellung für Crystal Reports (*.rpt*) und SQL Server Reporting Services (*.rdlc*)
- Assistenten, z. B. für Berichte, Datenbankverbindungen und Webservice-Proxys
- Debugging für .NET, JavaScript und TSQL mit Unterbrechung, Einzelschrittmodus, Anzeige von Werten und Veränderung von Werten/Code während der Laufzeit
- Kompilierung für verschiedene .NET-Versionen (Multi-Targeting)
- Verbreitung von Anwendungen
- Kontextsensitive Hilfefunktionen (Integration der MSDN-Library)
- Import/Export von Einstellungen der Entwicklungsumgebung
- Anpassung durch Add-ins und Automatisierung durch Makros

HINWEIS Spezielle Funktionen im Bereich Web- und Windows-Oberflächen werden in den Kapiteln zu ASP.NET (dieses Zusatzkapitel können Sie als PDF auf dem Leser-Portal herunterladen), Windowsoberflächen mit Windows Forms und Windows Presentation Foundation (WPF) besprochen.

Neuerungen in der Entwicklungsumgebung Visual Studio in Version 2010

Es folgt eine Liste der wesentlichen Neuerungen in Visual Studio 2010:

- Neue Oberfläche basierend auf WPF
- Nahtloses Zoomen im Editorfenster mit Maus
- Neue Dialogfelder für Projekte und Projektelemente mit Suchfunktion
- Verbessertes Suchdialogfeld für Quellcode
- Teilwortsuche für IntelliSense
- Eigener IntelliSense-Modus für testgetriebenes Entwickeln
- Generierung von bisher nicht deklarierten Klassen und Klassenmitgliedern bei ihrer ersten Verwendung
- Unterstützung für UML
- Generierung von Sequenzdiagrammen
- Anzeige der kompletten Aufrufhierarchie eines Codeelements
- Verbesserungen des WPF-Designers (Datenbindung, Pinseldefinition, u. a.)
- Neuer Designer für Silverlight-Anwendungen (erstmals in Visual Studio!)
- Neuer Designer für XSD-Schema-Dateien (XSD)
- Neuer Designer für Windows Workflow 4.0
- Navigation und grafische Darstellung von Abhängigkeiten im Architecture Explorer
- Debugging mit historischen Zuständen (IntelliTrace)
- Anheften von Variablenzuständen beim Debugging (Debugger Data Tipps)
- Bezeichnungen für Haltepunkte im Debugger
- Verbesserungen des ADO.NET Entity Framework-Designers (Forward Engineering, Namensgebung, u. a.)
- Oberflächentests (UI-Tests)
- Testwirkungsanalyse
- Eine reduzierte Version von dem Drittanbieterprodukt InstallShield ist als Alternative zum Visual Studio Installer enthalten für die Erstellung von MSI-Paketen
- Einfachere Installation von Erweiterungen durch Visual Studio Extension Manager

Neuerungen in der Entwicklungsumgebung Visual Studio in Version 2010 mit Productivity Power Tools

Die Productivity Power Tools bringen nochmals einige wesentliche Verbesserungen:

- Verbesserte Registerkartendarstellung und -verwaltung
- Stark verbessertes Dialogfeld für Verweise u. a. mit Suchfunktion
- Solution Navigator kombiniert Funktionen als Solution Explorer und Klassenbrowser (auch mit Suchfunktion)
- Automatisches Einfügen von schließenden Blockelementen (z. B. Klammern)
- Hervorheben der aktuellen Zeile durch grauen Balken
- Einfachere Verwaltung vertikaler Hilfslinien
- Blockauswahl und Blockänderung
- Zeilenauswahl mit Dreifachclick inklusive Verschiebefunktion
- Suche über Visual Studio-Funktionen
- Angleichen von Zuweisungen

Produktvarianten

Bisher gab es von Visual Studio neben den kostenfreien Express-Versionen zahlreiche kommerzielle Versionen: Standard, Professional sowie mindestens vier Ausprägungen der Team System-Reihe mit *Team Suite* an der Spitze. Den Namen *Visual Studio Team System* als Oberbegriff für alle Versionen, die nicht nur Editieren und Debuggen, sondern auch andere Bereiche des Application Life Cycle Management (ALM) unterstützen, beerdigt Microsoft mit Visual Studio 2010 wieder. Fortan gibt es neben den Express-Versionen nur noch drei Varianten: Professional, Premium und Ultimate. Premium versteht sich gegenüber Professional als »Vollausstattung für Softwareentwickler und -tester«. Alle ALM-Werkzeuge bietet hingegen nur die Ultimate-Version.

Visual Studio Test Professional 2010 und Microsoft Test Manager 2010

Für Tester gibt es speziell noch die Variante Visual Studio Test Professional 2010. Visual Studio Test Professional 2010 ist weder »ein Visual Studio ohne Editor und Designer« noch ein »abgespecktes Ultimate für die Tester«, sondern ein komplett eigenes Produkt mit dem Microsoft Test Manager im Kern. Der Test Manager (Codename: Camano) ist ebenfalls eine WPF-Anwendung (die aber nicht auf Visual Studio basiert).

Microsoft Test Manager 2010 bietet insbesondere:

- Erfassen von Anforderungen
- Erstellung der Testsuiten mit Testschritten
- Aufzeichnen von Oberflächentests
- Testplanung auf Basis der Anforderungen
- Testparametrisierung (datengetriebene Tests)
- Aufbau einer Testfall- und Testschrittbibliothek

- Verlinkung zum Buildmanagement
- Steuerung für manuelle und automatisierte Testausführung
- Videoaufzeichnung und Screenshot während eines Testlaufs
- Testauswertung, Fehlerverfolgung, Reporting

Vergleichstabelle

Die folgende Tabelle entstammt der Microsoft Deutschland-Website und zeigt den Funktionsumfang der verschiedenen Varianten. Microsoft vermarktet die Visual Studio Premium- und Ultimate-Varianten primär zusammen mit einem Abonnement des Microsoft Developer Network (MSDN), alias *MSDN Subscription*. Der Einzelerwerb der Professional-Version ist aber auch möglich; die Premium- und Ultimate-Version von Visual Studio kann man ohne MSDN-Abonnement nicht erwerben.

	Visual Studio 2010 Ultimate mit MSDN	Visual Studio 2010 Premium mit MSDN	Visual Studio 2010 Professional mit MSDN	Visual Studio 2010 Test Professional mit MSDN	Visual Studio 2010 Professional
BASISFUNKTIONEN					
Editor	X	X	X		X
Debugger	X	X	X		X
Designer (Desktop- und Web-UI, Daten, Klassen)	X	X	X		X
Deployment	X	X	X		X
DEBUGGEN UND FEHLERDIAGNOSE					
IntelliTrace™ (historischer Debugger)	X				
Statische Codeanalyse	X	X			
Codemetriken	X	X			
Profiling	X	X			
QUALITÄTSSICHERUNG					
Unit Tests	X	X	X		X
Code Coverage	X	X			
Test Impact Analysis	X	X			
Coded UI Tests	X	X			
Web Performance Tests	X				
Lasttests	X				
Test Case Verwaltung	X			X	
Manuelle Ausführung von Tests	X			X	
Aufzeichnung und Abspielen von manuellen Tests	X			x	
Lab Management Konfiguration	X			x	

	Visual Studio 2010 Ultimate mit MSDN	Visual Studio 2010 Premium mit MSDN	Visual Studio 2010 Professional mit MSDN	Visual Studio 2010 Test Professional mit MSDN	Visual Studio 2010 Professional
QUALITÄTSSICHERUNG					
Unterstützung mehrerer Monitore	X	X	X		X
Multi-Targeting (Unterstützung mehrerer Versionen von .NET Framework)	X	X	X		X
Webbereitstellung mit einem Klick	X	X	X		X
Unterstützung von JavaScript und jQuery	X	X	X		X
Unterstützung mehrerer Monitore	X	X	X		X
DATENBANKENTWICKLUNG					
Bereitstellung von Datenbanken	X	X			
Veränderungsmanagement von Datenbanken	X	X			
Unit Tests für Datenbanken	X	X			
Generierung von Testdaten für Datenbanktests	X	X			
UNTERSTÜTZTE ENTWICKLUNGSPLATTFORM					
Windowsentwicklung	X	X	X		X
Webentwicklung	X	X	X		X
Office- und SharePoint-Entwicklung	X	X	X		X
Cloudentwicklung (für Windows Azure)	X	X	X		X
ARCHITEKTUR UND MODELLIERUNG					
Architekturexplorer	X				
UML 2.0 Diagramme (Aktivitäts-, Use Case-, Sequenz-, Klassen- und Komponentendiagramme)	X				
Schichtendiagramme und Validierung von Abhängigkeiten	X				
Viewer für UML-, Schichten- und DGML-Diagramme	X	X			



	Visual Studio 2010 Ultimate mit MSDN	Visual Studio 2010 Premium mit MSDN	Visual Studio 2010 Professional mit MSDN	Visual Studio 2010 Test Professional mit MSDN	Visual Studio 2010 Professional
VERWALTUNG VON TESTUMGEBUNGEN					
Einrichten und Zurücksetzen von virtuellen Umgebungen	X			X	
Vorlagenbasierte Bereitstel- lung von Testumgebungen	X			X	
TEAM FOUNDATION SERVER					
Versionskontrolle	X	X	X	X	(X)
Workitem Tracking	X	X	X	X	(X)
Buildautomatisierung	X	X	X	X	(X)
Team Portal	X	X	X	X	(X)
Reporting & Business Intelligence	X	X	X	X	(X)
Agile Planning Workbook	X	X	X	X	(X)
Testcase-Verwaltung	X	X	X	X	(X)
Visual Studio Team Explorer 2010	X	X	X	X	(X)

Tabelle 5.2 Variantenvergleich (übernommen und erweitert von [MS02])

HINWEIS (x) bedeutet, dass die Funktion vorhanden ist, aber eine separate Client Access Licence (CAL) für den Team Foundation Server (TFS) erfordert.

Express-Editionen

Mit den Express-Editionen nimmt Microsoft auch nicht-professionelle Entwickler ins Visier, deren Anzahl die Firma mit 18 Millionen weltweit dreimal so hoch beziffert wie die Anzahl der professionellen Entwickler. Für diese Zielgruppe – »Hobbyisten, Enthusiasten und Studenten« – hatte Microsoft lediglich Angebote im Bereich Webentwicklung (mit dem Editor Web Matrix) und Datenbanken (MSDE). Seit Visual Studio 2005 bietet Microsoft ihnen nun mit der *Visual Studio Express*-Produktfamilie eine Serie von vier verschiedenen Produkten an: C++ Express, C# Express, VB.NET Express und Visual Web Developer Express. J# Express, das es zu Visual Studio 2005 gab, wird seit 2008 nicht mehr angeboten.

Ursprünglich sollten die Express-Editionen übrigens einen Preis von 49 Dollar haben. Microsoft hat aber dann beim Erscheinen der 2005er-Versionen bekannt gegeben, dass diese Editionen dauerhaft kostenfrei sein sollen. Dies bleibt auch bei den 2010er-Versionen so.

WICHTIG Microsoft verwendet den Begriff *Visual Web Developer (VWD)* nicht nur für eine der Express-Editionen, sondern auch für den Teil von Visual Studio, in dem ASP.NET-Webanwendungen entwickelt werden. In VWD sind einige Funktionen anders als in den übrigen Visual Studio-Projektvorlagen.

Preise für Visual Studio 2010

Es gibt keine vorgeschriebenen Preise von Microsoft. Deshalb kann die nachstehende Tabelle nur eine Momentaufnahme zur groben Orientierung sein. Die folgende Tabelle nennt Preise von <http://emea.microsoftstore.com>, Stand 14.06.2010. Diese Plattform wird nicht von Microsoft, sondern der arvato GmbH bereitgestellt, einem unabhängigen Händler.

	Vollversion	Upgrade bzw. Verlängerung
Professional mit MSDN Essentials	949 Euro	649 Euro
Professional mit MSDN	1.279 Euro/Jahr	849 Euro/Jahr
Premium mit MSDN	5.859 Euro/Jahr	2.459 Euro/Jahr
Ultimate mit MSDN	12.769 Euro/Jahr	4.079 Euro/Jahr
Test Professional mit MSDN	2.299 Euro/Jahr	959 Euro/Jahr
Team Foundation Server (TFS)	519 Euro	419 Euro

Tabelle 5.3 Marktpreis Stand 14.6.2010

TIPPS Der tatsächliche Preis, den ein Unternehmen für Lizenzen zahlen muss, ist sehr stark abhängig von dem vertraglichen Verhältnis zu Microsoft. Über Partnerschaften und Rahmenverträge kann man viel bessere Konditionen erhalten als beim Einzelbezug über Händler. Dies gilt insbesondere auch für die sehr teuren Produkte wie Visual Studio Ultimate. Um die für Sie optimale Lösung zu finden, wenden Sie sich am besten an einen Microsoft Certified Licensing Specialist (MCLS).

Sie können zum Beispiel erheblich Kosten sparen, indem Sie mit Microsoft einen so genannten *Volumenlizenzvertrag* abschließen. In einem Volumenlizenzvertrag sind die Preise – als Faustregel – nur halb so hoch. Normalerweise ist ein Volumenlizenzvertrag bei Microsoft erst ab fünf Lizenzen möglich, bei Visual Studio 2010 und MSDN geht das aber ab der ersten Lizenz.

Ein MSDN-Abonnement hat mehrere Vorteile: Es sind Test- und Entwicklerlizenzen aller Microsoft Betriebssysteme, Serveranwendungen, Office-Anwendungen sowie kaufmännischer Anwendungen enthalten. Ab der Premium-Version erhält man für Microsoft Office 2010, Microsoft Visio und Microsoft Project sogar eine Lizenz für den produktiven Einsatz (nicht aber eine Lizenz des Betriebssystems!). Zudem ist im MSDN-Abonnement eine Lizenz für Team Foundation Server (TFS) enthalten. Ein MSDN-Abonnement umfasst zudem Support-Anrufe bei Microsoft.

MSDN Essentials ist kein vollwertiges MSDN-Abonnement, sondern eine Art Light-Version von MSDN und bietet für ein Jahr Zugriff auf Windows 7 Ultimate, Windows Server 2008 und SQL Server 2008 Datacenter für Test- und Entwicklungszwecke. Nach Ablauf dieses Jahres dürfen die Lizenzen nicht mehr weiter verwendet werden, es kann aber preisgünstig auf ein vollwertiges MSDN-Abonnement hochgestuft werden. Die Lizenz für Visual Studio Professional, mit welcher das MSDN Essentials ausgeliefert wird, ist natürlich nicht zeitlich befristet.

WPF-Oberfläche

Die *Windows Presentation Foundation* (WPF) gehört wegen ihres Leistungshungers und der noch nicht ganz ausgereiften Entwicklerunterstützung in Visual Studio zu den umstrittenen Teilen von .NET. Windows Vista sollte ursprünglich eine WPF-Oberfläche erhalten, aber dieses Projekt scheiterte. So verwundert es nicht, dass viele Entwickler verunsichert waren, dass ausgerechnet Visual Studio, das auch bisher schon manchmal sehr zäh arbeitet, als erste große Anwendung aus Redmond eine WPF-Oberfläche erhalten hat.

Geschwindigkeit

Mit der Beta 1 von Visual Studio 2010 bestätigten sich diese Befürchtungen, denn diese Vorabversion war ausgesprochen langsam. Ab der Beta 2-Version lief Visual Studio flüssiger. Die Verschiebung des Erscheinungstermins von März auf April 2010 hatte ihre Ursache in den anhaltenden Leistungsproblemen. Die ausgelieferte Endfassung nun ist deutlich schneller als die Vorabversionen, aber zumindest gefühlt leicht langsamer als Visual Studio Version 2008, gerade beim Öffnen großer Projektmappen.

Oberflächlich wenig Neues

Neben der Geschwindigkeit war die zweite Befürchtung der Entwicklergemeinde, dass die Entwicklungsumgebung nun »verspielt« ist und mit allerhand Effekten den Entwickler eher von der Arbeit ablenkt. Tatsächlich hat sich aber oberflächlich wenig verändert. Anstelle des grauen Grundtons ist nun ein blau getreten. Ansonsten sind die Fenster einschließlich der Fensterverwaltung, die Menüpunkte und die Symbolleisten immer noch die gleichen und auch die Grundprinzipien der Bedienung mit Projektmappen-Explorer, Server-Explorer und Team-Explorer von Visual Studio sind bis auf kleinere Verbesserungen gleich geblieben.

Fensterverwaltung

Für Visual Studio benötigt ein Softwareentwickler einen großen Monitor, denn die Entwicklungsumgebung bietet eine hohe Anzahl verschiedener informativer und hilfreicher Fenster. Neben Code-Editoren und visuellen Editoren (die in beliebiger Anzahl gleichzeitig angezeigt werden können) gehören zu den wichtigsten Fenstern der *Projektmappen-Explorer* (*Solution Explorer*) (der die Projektmappe, die Projekte und die Projektelemente zeigt), das *Eigenschaftsfenster* (das für ein gewähltes Element die verfügbaren Einstellungen darstellt), die *Werkzeugleiste* (*Toolbox*) (mit den verfügbaren grafischen Elementen für Web und Windows Forms), die *Fehlerliste* (mit den Compiler-Fehlern), die *Klassenansicht* (die die Klassen des aktuellen Projekts den Namensräumen gemäß hierarchisch anzeigt) und das Fenster *Dynamische Hilfe* (das hilfreiche Links zu dem gerade ausgewählten Steuerelement bzw. dem gerade codierten Objekt zeigt).

Während des Debug-Vorgangs sind dagegen andere Fenster wichtig, z.B. das *Lokal*-Fenster (mit den aktuellen Variablenwerten), das *Output*-Fenster (das man über `Debug.WriteLine()` mit Informationen versorgen kann) und das *Direktfenster* (*Intermediate Window*) (in dem man nach dem Anhalten des Programms Variablen abfragen oder Befehle absetzen kann). Die Größe und Anordnung der einzelnen Fenster ist frei wählbar – und Visual Studio merkt sich auch, welche Konfiguration der Entwickler bei Codierung und Debugging bevorzugt.

Fenstermodi

Bereits seit Visual Studio .NET 2002 kann die Vielfalt der Fenster dem Entwickler als eine Bedrohung erscheinen. Die Anzahl der verschiedenen Entwicklungsumgebungsfenster ist weiter gewachsen, der Positionierungsvorgang innerhalb von Visual Studio ist seit der Version 2005 aber deutlich besser als in den Versionen 2002/2003. Jedes Fenster unterstützt drei Modi:

- Dokument im Registerkartenformat (*Tabbed*),
- unverankert (*Floating*) und
- andockbar (*Dockable*).

Tabbed bedeutet, dass das Fenster als Registerkarte im Hauptbereich erscheint. Neu ist, dass jetzt nicht nur Dokumentfenster, sondern alle Fenster den *Tabbed*-Modus besitzen. *Floating* bedeutet, dass das Fenster frei über allen anderen Fenstern liegt. *Dockable* heißt, dass das Fenster an andere Fenster »angeklebt« werden kann oder als Registerkarte in anderen Fenstern erscheint. Das Ankleben war in Visual Studio .NET 2002/2003 ein sehr schwieriger Vorgang. Visual Studio bietet seit Version 2005 dafür Navigationshilfen (siehe folgende Abbildung). Ein Ankleben ist in allen vier Himmelsrichtungen möglich. Durch das Fallenlassen des Fensters auf dem Zentrum eines Navigationselements wird es als Registerkarte in das bestehende Fenster einsortiert.

TIPP

Den Modus eines Fensters schalten Sie um, indem Sie das Kontextmenü der Fensterleiste aufrufen. Den *Tabbed*-Modus können Sie über eine Einstellung unter *Extras/Optionen/Umgebung/Allgemein/Fensterlayout* durch einen Multi-Dokument-Modus ersetzen.

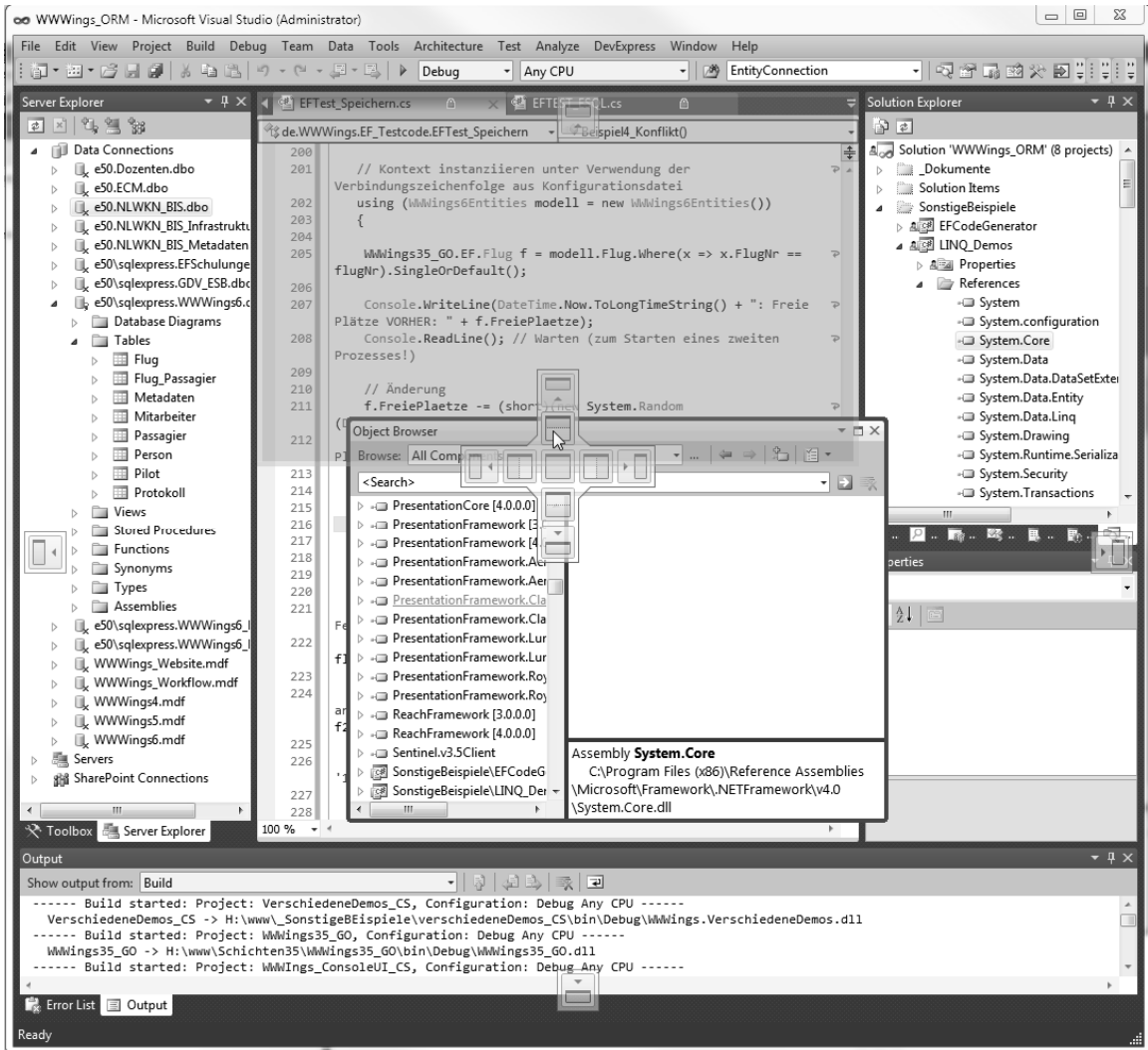


Abbildung 5.1 Hilfe beim Andocken von Fenstern in Visual Studio

Auch die Behandlung der Dokumente im *Tabbed*-Modus wurde verbessert. Am rechten Ende des Hauptfensters existiert ein Auswahlménü, das die Liste aller Fenster zeigt. Durch die Tastenkombination **[Strg] + [F4]** erscheint ein Dialogfelder, das die Iteration durch die Fenster mit der Taste **[Tab]** ermöglicht.

Jedes Dokumentfenster bietet mit *Vollständigen Pfad kopieren (Copy Full Path)* und *Enthaltenen Ordner öffnen (Open Containing Folder)* zwei sinnvolle neue Funktionen, um den physischen Speicherort des Dokuments zu ermitteln.

HINWEIS Über die Menüs *Extras/Optionen* und *Extras/Anpassen* können Sie zahlreiche Einstellungen für die Entwicklungsumgebung vornehmen. Neu in Visual Studio seit Version 2005 ist, dass Sie diese Einstellungen im- und exportieren können über *Extras/Einstellungen importieren und exportieren*. Mit dieser Funktion kann man auch eigene Einstellungen komplett zurücksetzen.

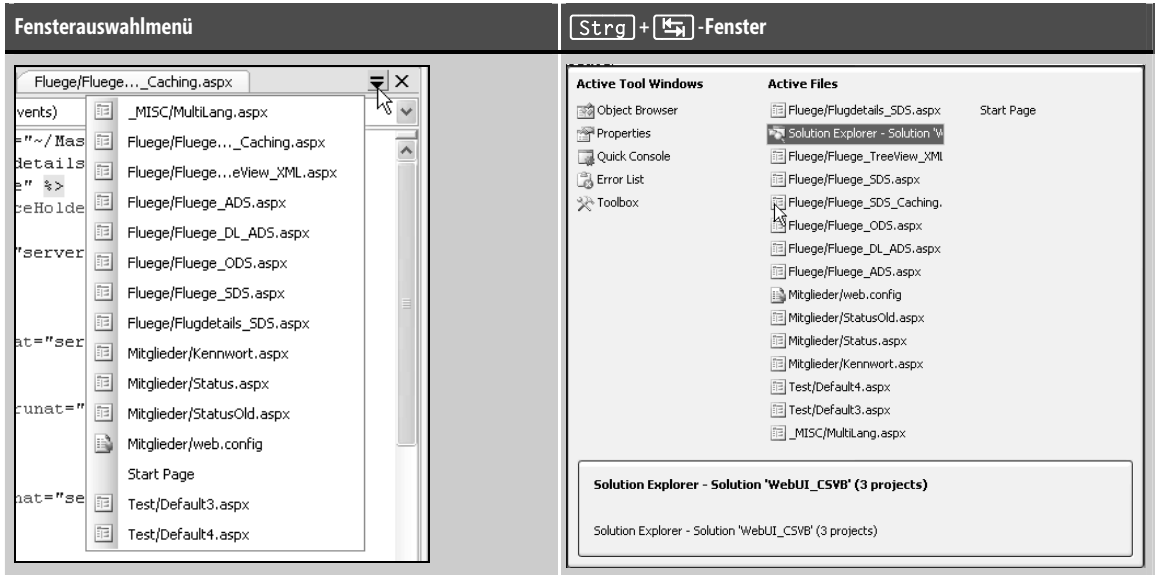


Tabelle 5.4 Fensterverwaltung in Visual Studio

Verbesserungen der Registerkartenverwaltung (engl. Tab Well)

Visual Studio zeigt seit Langem für jede geöffnet Codedatei im Hauptbereich eine Registerkarte. Mit den Visual Studio 2010 Productivity Power Tools hat Microsoft die Verwaltung der Registerkarten für die geöffneten Dateien erheblich verbessert:

- Registerkarten aus verschiedenen Projekten werden farblich anders dargestellt. Die Farben sind in den Visual Studio-Einstellungen wählbar. Man kann die Farben auch auf der Grundlage von regulären Ausdrücken auf Basis des Dateinamens vergeben lassen.
- Registerkarten werden auf Wunsch seitlich vertikal angeordnet
- Der Benutzer kann Registerkarten »festheften«, damit diese immer sichtbar sind. Optional können Sie in einer eigenen Zeile oder Spalte angezeigt werden.
- Man kann das Symbol wählen, mit dem Visual Studio in der Registerkarte anzeigt, dass eine Datei im aktuellen Zustand noch nicht gespeichert wurde

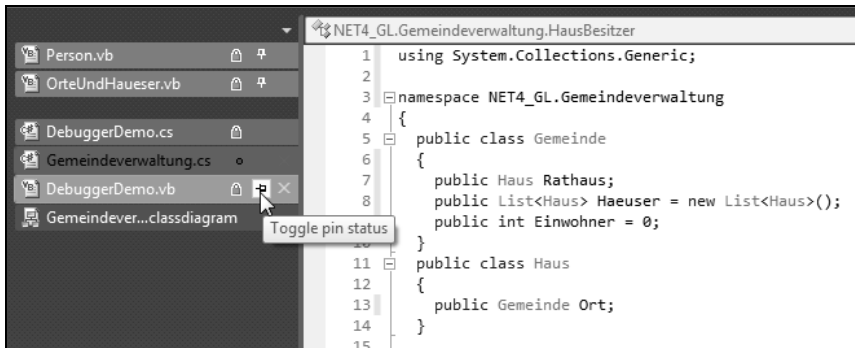


Abbildung 5.2 Vertikale Anordnung, Festheften und farbliche Hervorhebung

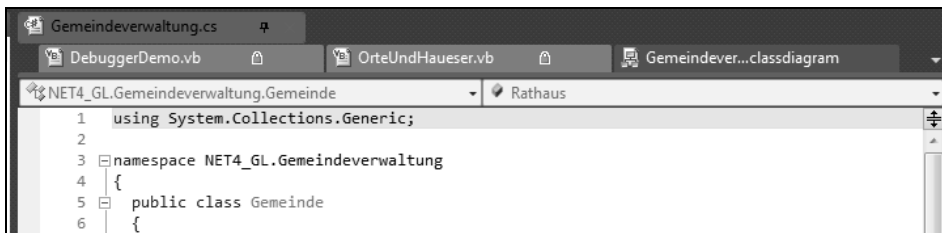


Abbildung 5.3 Anzeige der angehefteten Registerkarten in einer eigenen Zeile

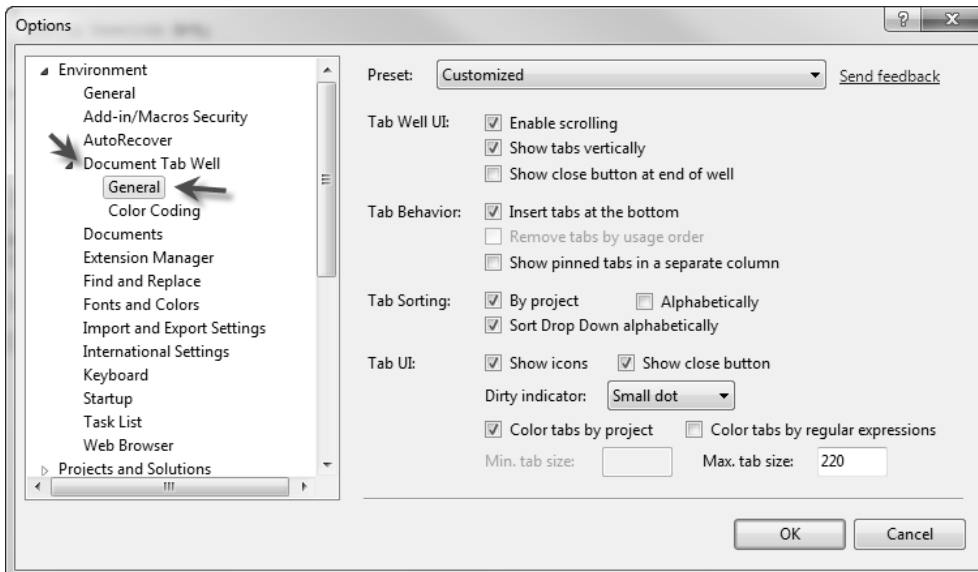


Abbildung 5.4 Einstellungen für Registerkarten (unter Tools/Options/Document Tab Well/General)

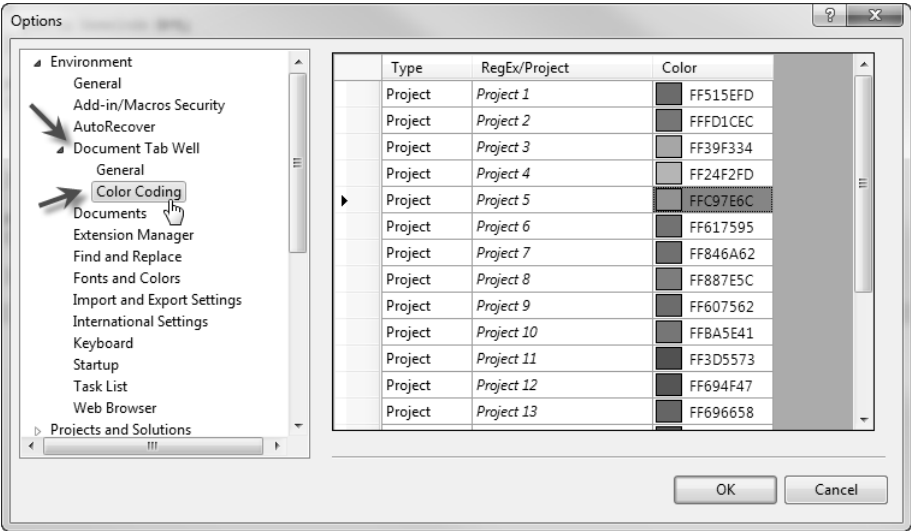


Abbildung 5.5 Farbwahl für Registerkarten (unter Tools/Options/Document Tab Well/Color Coding)

Programmiersprachen

Mit dem .NET Framework liefert Microsoft auch kostenlose Kommandozeilen-Compiler für die Sprachen C#, Visual Basic und JScript .NET aus. Jeder beliebige Texteditor kann folglich für die Entwicklung von .NET-Anwendungen eingesetzt werden. Mehr Komfort und höhere Produktivität verspricht jedoch der Einsatz einer integrierten Entwicklungsumgebung (Integrated Development Environment, IDE). Die von Microsoft gelieferte Entwicklungsumgebung heißt Visual Studio.

Neben C#, C++ und Visual Basic unterstützt Visual Studio 2010 auch die Programmiersprache F#, eine von Microsoft Research entwickelte Multi-Paradigmen-Sprache, die ML und OCaml ähnelt. F# ist insbesondere im Bereich des funktionalen Programmierens stärker als C# und VB. F# wird in diesem Buch aber nicht behandelt.

Sprachcompiler	Enthalten in .NET Framework 4.0	Unterstützung in Visual 2010	Kostenfreie Express-Variante
CSharp (C#) 4.0	Ja	Ja	Ja
Visual Basic 10.0	Ja	Ja	Ja
Unmanaged Visual C++ 10.0	Nein	Ja	Ja
Managed Visual C++ 10.0 (C++/CLI)	Nein	Ja (aber nicht für Webanwen- dungen)	Ja (aber nicht für Webanwen- dungen)
JSharp (J#) 8.0 für J# 2.0	Nein (Add-on)	Ja (Add-On)	Nein (nur auf dem Stand von Visual Studio 2005)
JScript .NET 10.0	Ja	Nein	Nein
F#	Nein	Ja	Nein

Tabelle 5.5 Sprachunterstützung im .NET Framework und in Visual Studio

HINWEIS An Visual Studio kommt niemand vorbei, der mit C++ entwickeln möchte: Der Microsoft Visual C++-Compiler ist nicht im .NET Framework Redistributable enthalten. Man erhält ihn aber mit Visual C++ Express ebenfalls kostenlos.

Projektverwaltung

Einzelne Codedateien können zwar bearbeitet werden, seine ganze Kraft entfaltet Visual Studio aber erst beim projektbasierten Arbeiten. Folglich erwartet die Entwicklungsumgebung beim Start das Erstellen eines Projekts oder das Öffnen eines vorhandenen Projekts. Das umfangreiche Dialogfeld *Neues Projekt*, der die verfügbaren Projektvorlagen anzeigt, erreicht man über das Menü *Datei/Neu/Projekt*. Die Funktion *Online-vorlagen durchsuchen* stand zum Redaktionsschluss dieses Buchs noch nicht zur Verfügung.

Ein Visual Studio-Projekt besitzt eine Projektdatei mit der Dateierweiterung *.vbproj*, *.csproj*, *.vjsproj* oder *.vcproj*. Die Buchstaben vor dem *proj* zeigen dabei die gewählte Programmiersprache an.

Ein Visual Studio-Projekt dient dem Anlegen einer Ein-Datei-Assembly in einer bestimmten Sprache. Visual Studio unterstützt auch in der Version 2010 nicht das Erstellen von Mehr-Dateien-Assemblies, was mit den Kommandozeilen-Compilern jedoch möglich ist. Eine Ausnahme bildet der Visual Web Developer, der im Normalfall selbst gar keine Assemblys erzeugt. Wie im Kapitel zu ASP.NET (dieses Zusatzkapitel können Sie als PDF auf dem Leser-Portal herunterladen) erläutert, obliegt die Kompilierung hier dem ASP.NET Page Framework.

Projektvorlagen

Die Projektvorlagen sind nach Sprachen sortiert. Welche Sprache dabei an exponierter Stelle genannt wird (siehe Abbildung), hängt von der beim ersten Start der Entwicklungsumgebung gewählten Präferenzsprache ab. Diese Präferenzsprache kann im Menü *Extras/Einstellungen importieren und exportieren* geändert werden. Man sieht in dem Dialogfeld maximal die Programmiersprachen, die Sie installiert haben bzw. die in Ihrer Visual Studio-Produktvariante enthalten sind.

Das Dialogfeld zum Anlegen eines Projekts ist in Visual Studio 2010 völlig neu gestaltet. Der Entwickler kann nun endlich die installierten Vorlagen nach Namen sortieren und in Vorlagennamen und Beschreibungstexten suchen. Angesichts der stetig wachsenden Anzahl von Vorlagen ist dies eine wichtige Erweiterung.

HINWEIS Wie die folgende Abbildung beweist, sucht der Projektdialog nicht nur in den Namen der Projektvorlagen, sondern auch in deren Beschreibungstexten, sonst würde *WPF Application* nicht bei der Suche nach *Windows* gefunden worden.

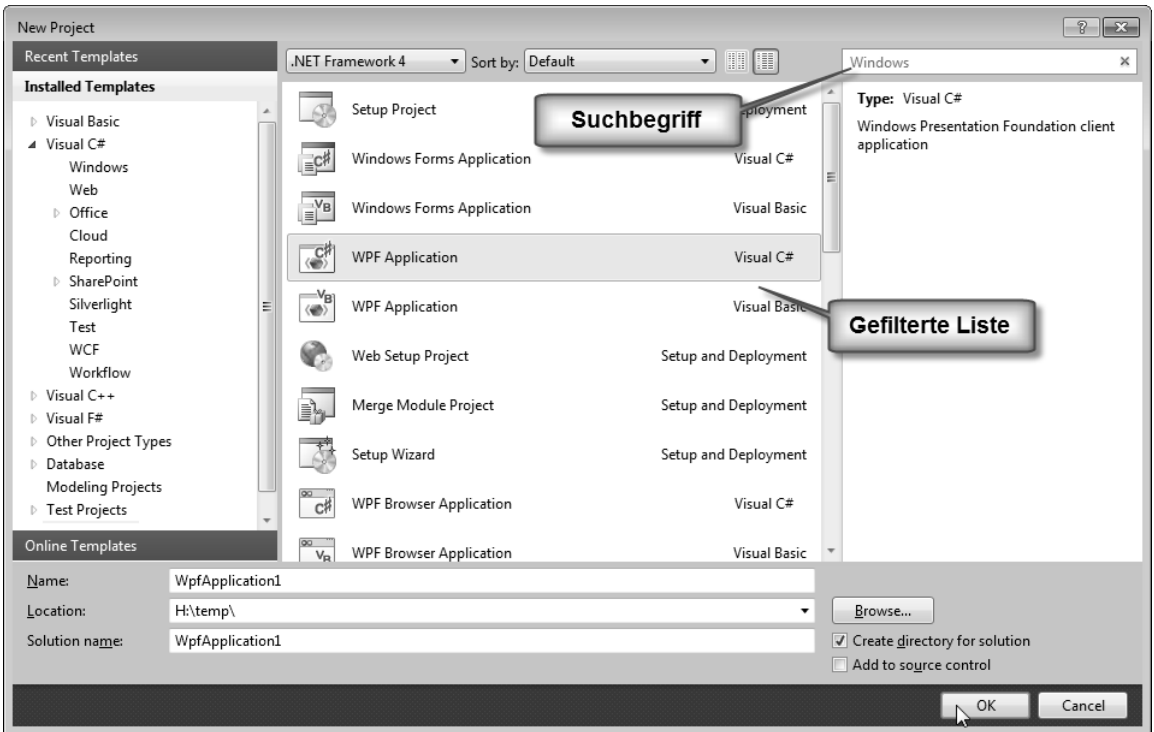


Abbildung 5.6 Auswahl einer Projektvorlage

Die richtige Auswahl der Projektvorlage ist wichtig: Die Sprache kann der Entwickler nachträglich nicht mehr ändern (außer bei dem Website-Modell). Eine Mischung verschiedener Programmiersprachen ist nur für Webprojekte möglich.

Auch die Ausgabeform ist in vielen Fällen durch die Erstausswahl festgeschrieben: Während man aus einer DOS-Anwendung nachträglich noch eine Windows-Anwendung machen kann, führt z.B. der Weg von einer Windows-Anwendung zu einem Webservice nur über das Neuanlegen eines Projekts (einzelne Code-dateien können dann manuell übernommen werden).

Sprachneutral sind die Projektvorlagen zur Erstellung von Installationsdateien und Datenbanken. Die Projektvorlage *Visual Studio Add-in* fragt über einen Assistenten nach der von Ihnen gewünschten Programmiersprache.

Der Visual Web Developer bietet zwei verschiedene Modelle zur Entwicklung von ASP.NET-Webanwendungen (Webseitenmodell und Webanwendungsmodell). Dies ist Thema ist dem Buch [HSJF01].

TIPPS

Unter dem Eintrag *Online Templates* in der Navigationsleiste links finden sich Zusatzvorlagen, die Microsoft zum Download bereitstellt.

Sie können in Visual Studio jedes konkrete Projekt als eine Projektvorlage für andere Projekte speichern über die Funktion *Datei/Vorlage exportieren (File/Export Template)*.

Multi-Targeting

Wie schon die Vorgängerversion unterstützt Visual Studio 2010 *Multi-Targeting*, also das Programmieren für verschiedene .NET Framework-Versionen. Die neue Version unterstützt .NET 2.0, 3.0, 3.5 und 4.0. Die Auswahl kann man schon in dem Projektdialog (siehe vorgehender Abschnitt) treffen, dann werden nur die für die gewählte .NET Framework-Version passenden Vorlagen angezeigt. Nach dem Anlegen fügen sich auch IntelliSense-Unterstützung, Objekt-Browser und Compiler der Versionsvorgabe ein. Microsoft nennt die Funktion *Multi-Targeting*.

Die .NET Framework-Version kann der Entwickler auch jederzeit noch nachträglich in den Projekteigenschaften umschalten. Während ein C#-Entwickler die Auswahl für *Target Framework* direkt auf der Registerkarte *Application* findet, ist diese Auswahl für Visual Basic Entwickler etwas versteckt (siehe Abbildung).

ACHTUNG Leider fehlt aber auch in Visual Studio 2010 wie in der Vorgängerversion die Differenzierung zwischen den Grundversionen und den Service Packs, die ja zum Teil erhebliche Erweiterungen mit sich bringen. Das führt in der Praxis zu Problemen, denn .NET-Anwendungen stürzen in dem Moment ab, wenn sie eine Funktion aufrufen, die in einem Service Pack hinzugefügt wurde und dieses Service Pack auf dem Zielsystem nicht vorhanden ist.

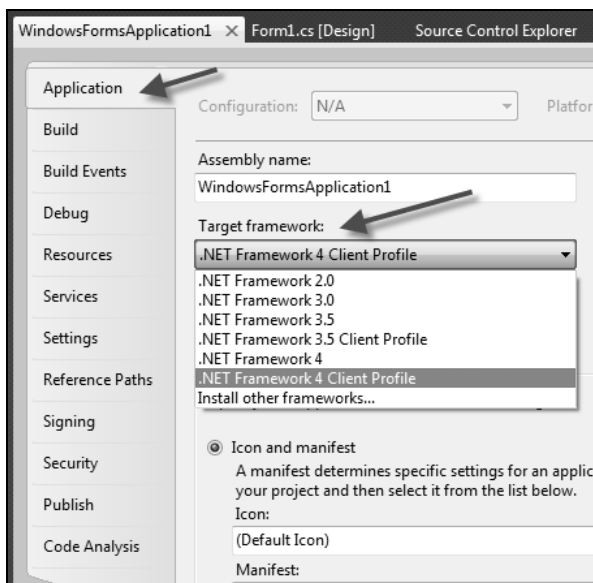


Abbildung 5.7 Multi-Targeting-Einstellung in C#-Projekten

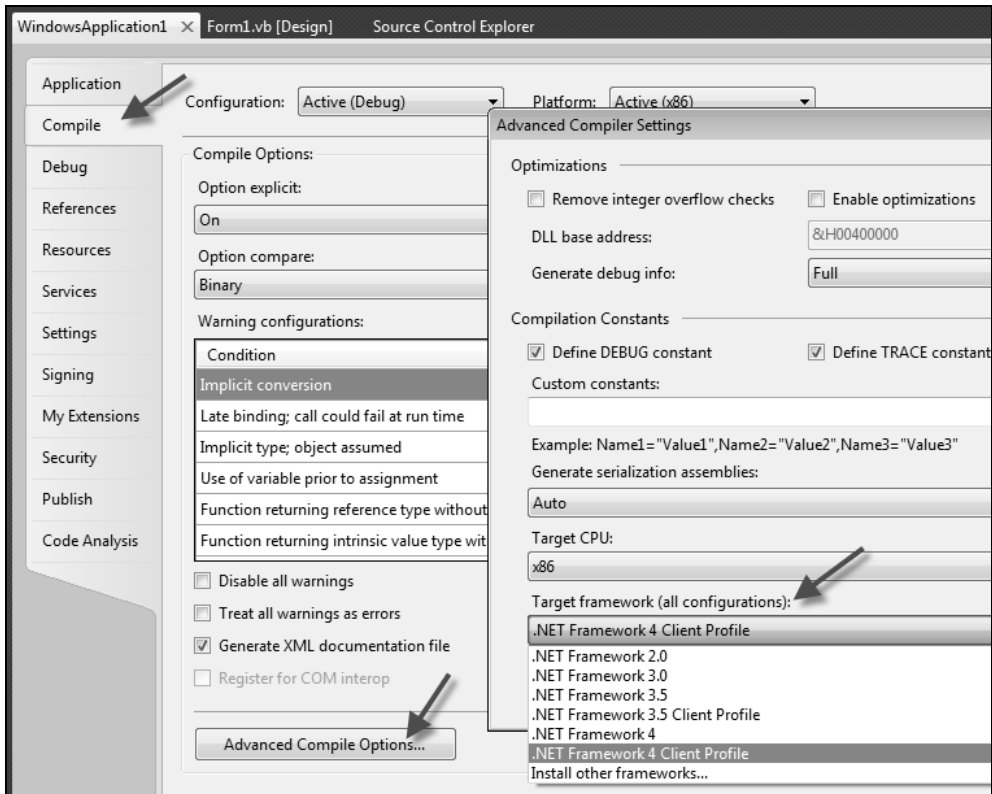


Abbildung 5.8 Multi-Targeting-Einstellung in Visual Basic-Projekten

Die nachträgliche Änderung des Zielframeworks ist mit einer Warnmeldung verbunden, die insbesondere dann ernst zu nehmen ist, wenn man vor hat, die Version herabzusetzen.

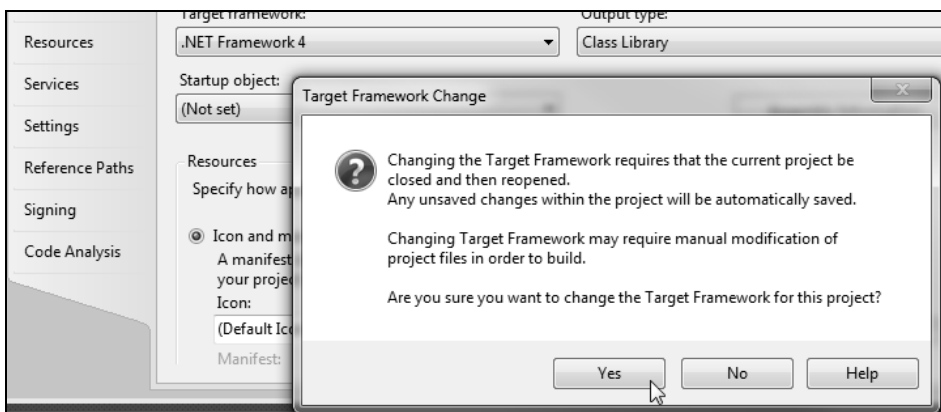


Abbildung 5.9 Warnung bei Änderung des Zielframeworks

HINWEIS Man kann in einer Assembly mit einer höheren .NET-Version (z.B. 4.0) Assemblys nutzen, die mit einem niedrigeren Zielframework (z.B. 2.0) kompilieren. Umgekehrt ist dies aber nicht möglich. Eine Ausnahme sind hier allerdings .NET 2.0, 3.0 und 3.5, die alle auf der gleichen CLR-Version 2.0 basieren und daher beliebige Referenzen zwischen Assemblys verschiedener .NET-Versionen mit der gleichen CLR-Version erlauben.

TIPP Wenn Visual Studio sich beim Umstellen des Target Framework über mangelnde Zugriffsrechte beschwert, ist die Datei wahrscheinlich schreibgeschützt, z.B. weil Sie mit TFS offline arbeiten. Bei anderen Einstellungen im Projektdialog bekommt Visual Studio es hin, den Benutzer um Entfernung des Schreibschutzes zu fragen, nicht aber bei Target Framework.

Das löst man so: Man ändert eine andere Einstellung, z.B. Default Namespace und speichert. Dann kommt die Nachfrage zur Entfernung des Schreibschutzes, die zu bejahen ist. Dann ändert man wieder den Default Namespace und dann das Target Framework.

Beschränkung auf das .NET Framework Client Profile

In Visual Studio 2010 steht in den Projekteigenschaften als »Target Framework« auch *3.5 Client Profile* und *4.0 Client Profile* zur Verfügung. Das *.NET Framework Client Profile* ist eine abgespeckte Variante des .NET Framework (siehe Kapitel 1 »Einführung«). Visual Studio stellt hier eine Kompilierungsoption bereit, die warnt, wenn Bibliotheken verwendet werden, die nicht im *.NET Framework Client Profile* enthalten sind. Das Kompilat ist aber dann auch lauffähig auf dem vollständigen .NET Framework.

Nach der Aktivierung des .NET Client Profile in den Projekteigenschaften stehen einige Assemblys nicht mehr zur Referenzierung zur Verfügung, z.B. *System.Web.dll*, *System.Workflow.Activities.dll*, *CrystalDecisions.CrystalReports.Engine.dll*, u.v.a.

ACHTUNG Beim Anlegen eines neuen Projekts mit Target *4.0* ist immer das Client-Profile angewählt. Diese Standardangabe lässt sich nicht ändern. Die Standardvorlage führt zwangsläufig zu Fehlern, wenn man in einer Projektmappe Projekte mit Client Profile und vollständigem .NET Framework mischt.

Die dann auftretenden Compiler-Fehlermeldungen können sehr kurios sein. Beispielszenario: Eine Assembly A, die mit dem .NET Client Profile arbeitet, verwendet eine Assembly B, die mit dem kompletten .NET Framework arbeitet. Sobald die referenzierte Assembly B Klassen verwendet, die es nicht im Client Profile gibt, blendet Visual Studio anstelle einer aussagekräftigen Fehlermeldung die ganze Assembly B aus. Folglich ist der Fehler, dass die Klassen aus B nicht mehr gefunden wird, obwohl B weiterhin in A referenziert ist (Details siehe [HS08]).

Projektmappen

Nach dem Erstellen des Projekts fällt sofort auf, dass Visual Basic im so genannten *Projektmappen-Explorer* (*Solution Explorer*) auch noch eine übergeordnete Projektmappen-Datei (*Solution*) anzeigt, die automatisch angelegt wird und die Dateierweiterung *.sln* erhält.

Eine Projektmappe enthält ein oder mehrere Projekte. Sie kann auch in einem speziellen Verzeichnis mit Namen *Projektmappen-Elemente* (engl. *Solution Items*) direkt projektübergreifende Elemente beinhalten. Der Entwickler kann Projekte einer Projektmappe in logische Ordner sortieren. Die Sortierung ist dabei unabhängig von der physikalischen Struktur im Dateisystem.

Eine Projektmappe darf Projekte verschiedenen Typs (Windows, Konsole, Web etc.) und in verschiedenen Programmiersprachen enthalten. Ob und in welcher Reihenfolge Visual Studio die einzelnen Projekte kompiliert, ist über den Kontextmenüeintrag *Configuration Manager* der Projektmappe steuerbar.

Visual Studio speichert ein neu angelegtes Projekt und die Projektmappe zusammen in dem im *Neues Projekt*-Dialogfeld angegebenen Verzeichnis ab. Für jedes weitere über *Hinzufügen/Neues Projekt* hinzugefügte Projekt kann man den Pfad individuell wählen. Möchte man den Standort der Projektmappe ändern, muss man entweder eine neue leere Projektmappe anlegen und die einzelnen Projekte hinzufügen oder die in der *.sln*-Datei hinterlegten Pfade manuell mit einem Texteditor verändern.

TIPP

Ein Projekt kann in mehreren Projektmappen enthalten sein. Man sollte jedoch vermeiden, ein Projekt mehrmals gleichzeitig zu öffnen (außer wenn ein Versionsverwaltungssystem verwendet wird).

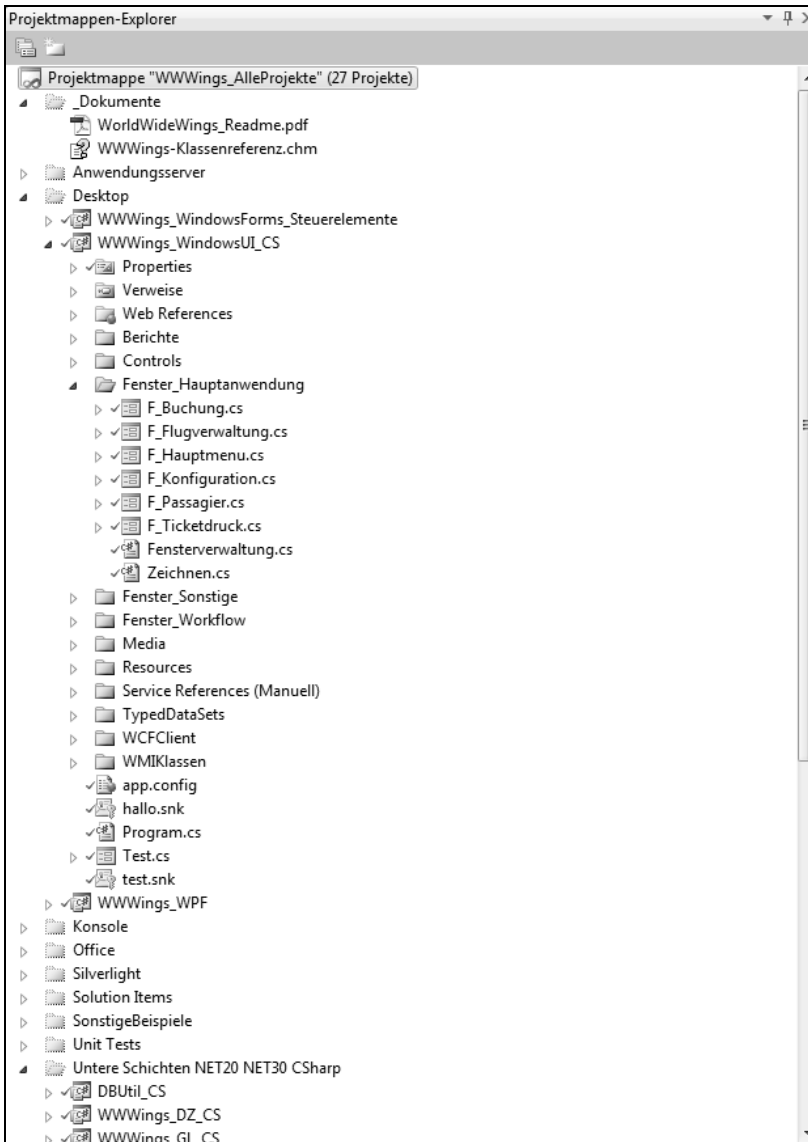


Abbildung 5.10 Ein Webprojekt mit zusätzlichen Klassenbibliotheksprojekten im Projektmappen-Explorer

TIPP Im Kontextmenü eines Projekts steht der Punkt *Open Folder in Windows Explorer* zur Verfügung, mit dem man direkt an den physikalischen Standort eines Projekts im Dateisystem springen kann.

Wenn man die Visual Studio 2010 Productivity Power Tools installiert, gibt es alternativ zum Solution Explorer auch noch einen Solution Navigator. Dieser stellt eine Baumansicht zur Verfügung, die nicht nur wie der Solution Explorer bis auf Dateiebene geht, sondern herunter bis in die Klassen, die Klassenmitglieder und die lokalen Variablen in den Methoden. Eine Suchfunktion erlaubt die Volltextsuche über alle Elemente (siehe Abbildung).



Abbildung 5.11 Solution Navigator

Projektelemente

Ein Projekt kann aus den unterschiedlichsten Elementen bestehen, z.B. aus Codedateien, XML-Dateien, HTML-Dateien und Grafiken. Beim Anlegen eines Projekts legt die Entwicklungsumgebung spezifisch für jeden Projekttyp einige Elemente mit an, beispielsweise ein leeres Formular bei einer Windows-Anwendung. Zusätzliche Elemente aus der Menge der verfügbaren Elementvorlagen kann man jederzeit über *Hinzufügen/Neues Element* oder *Hinzufügen/Vorhandenes Element* (jeweils im Kontextmenü eines Projekts) ergänzen. Die hinzufügbaren Elemente unterscheiden sich zwischen dem VWD und dem übrigen Visual Studio. Außerdem hängen sie von der gewählten Programmiersprache ab.

TIPP

Im Menü *Hinzufügen/Vorhandenes Element* kann man auswählen, ob das Element in das Projekt kopiert werden soll oder verlinkt werden soll.

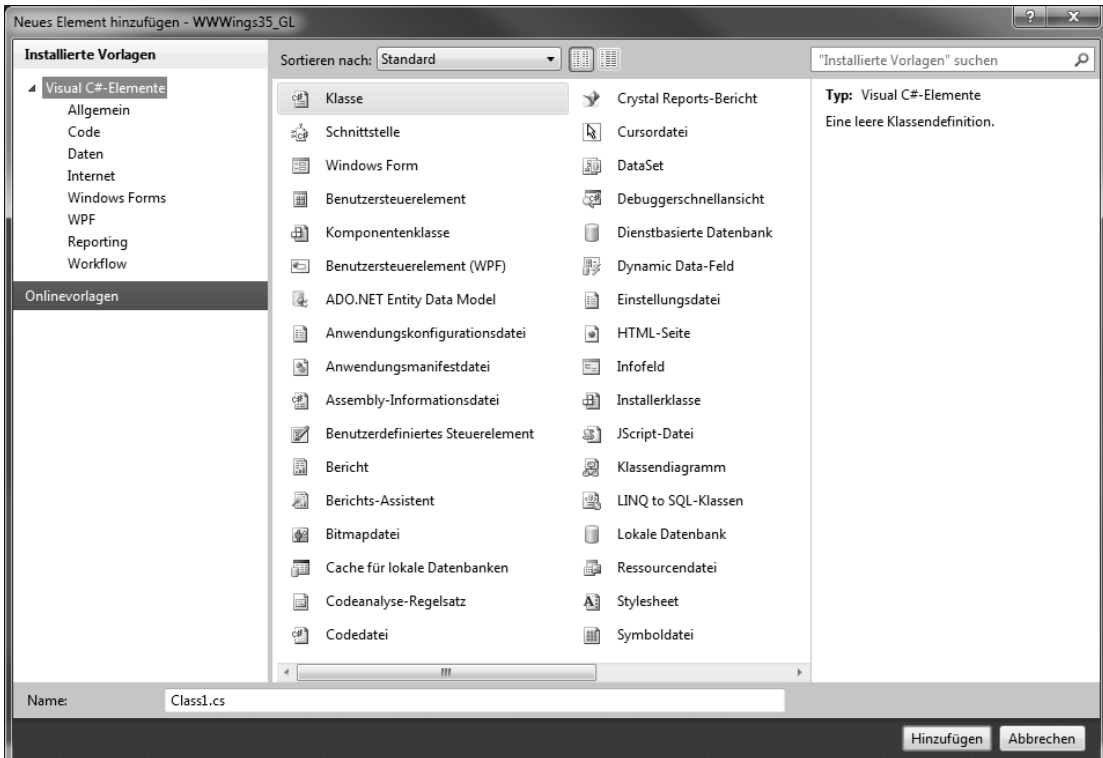


Abbildung 5.12 Hinzufügen eines neuen Elements zu Visual Studio-Projekten

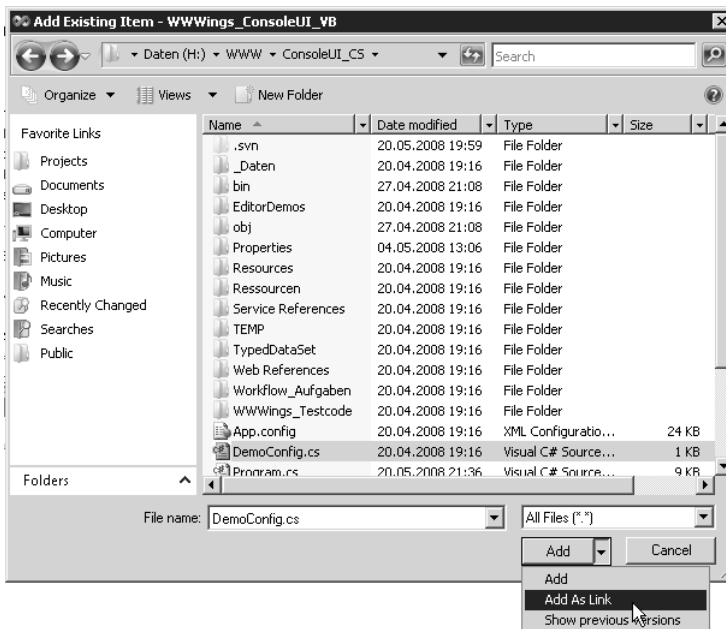


Abbildung 5.13 Verknüpfen eines bestehenden Elements zu einem Visual Studio-Projekt

HINWEIS Zu jedem Projektelement kann der Entwickler im Eigenschaftsfenster einstellen, ob dieses Element kompilierbaren Programmcode enthält, eine in die Assembly einzubettende Ressource (z. B. Grafik) darstellt oder vom Compiler ignoriert werden soll. Diese Auswahloption steht aber nicht im VWD zur Verfügung. Hier werden beim Kompilieren innerhalb der Entwicklungsumgebung alle Codedateien berücksichtigt, außer solche Dateien, die die Dateierweiterung *.excluded* besitzen.

TIPP Elementvorlagen können genauso wie Projektvorlagen über *Datei/Vorlage exportieren* erzeugt werden.

Ordnerstruktur

Innerhalb eines Projekts können die Projektelemente in Unterordnern angeordnet werden. Ob der Unterordner Einfluss auf die Codierung hat, hängt von Projekttyp und Programmiersprache ab. Bei Webprojekten ist die Ordnerstruktur für die Adressierung der Webseiten signifikant. Bei den übrigen Projektarten besitzen C#-Projekte die Eigenschaft, den Ordnernamen als Unternamensraum mit in neuen Codedateien zu verwenden, während in Visual Basic-Projekten die Ordnerstruktur völlig ignoriert wird.

Dateien können zwischen Ordnern per Drag & Drop verschoben werden. Wenn es in dem Zielverzeichnis bereits eine entsprechende Datei gibt, warnt Visual Studio, dass diese Datei überschrieben werden könnte.

HINWEIS Beim Drag & Drop zwischen Projekten in einer Projektmappe wird die Datei aber kopiert statt verschoben.

Projekteigenschaften

In den Projekteigenschaften legt der Entwickler Compiler- und Debugger-Optionen fest. Hier findet man beachtliche Unterschiede je nach verwendeter Programmiersprache.

Registerkarte	C#	Visual Basic
Application	Anwendungsart .NET Framework-Version Name der Assembly Standardnamensraum Symbol Windows-Manifest Annotationen auf Assembly-Ebene	Anwendungsart Name der Assembly Wurzelnamensraum Symbol Windows-Manifest Annotationen auf Assembly-Ebene Einstellungen zum VB-Anwendungsmodell
Build bzw. Compile	Compiler-Einstellungen	Compiler-Einstellungen .NET Framework-Version
Build Events	Skripte vor/nach dem Kompilieren	<i>Nicht verfügbar</i>
Debug	Debugger-Einstellungen	Debugger-Einstellungen
Resources	Globale Ressourcen	Globale Ressourcen
Services	Zugriff auf ASP.NET-Anwendungsdienste	Zugriff auf ASP.NET-Anwendungsdienste
Settings	Definition von Benutzereinstellungen	Definition von Benutzereinstellungen
Reference Path bzw. References	Ordner zum Suchen von Assemblys	Liste der Verweise Liste der eingebundenen Namensräume ▶

Registerkarte	C#	Visual Basic
Signing	Digitale Signierung von Assemblys	Digitale Signierung von Assemblys
Security	Einstellungen zur Code Access Security	Einstellungen zur Code Access Security
Publish	Veröffentlichung über Click-Once-Deployment	Veröffentlichung über Click-Once-Deployment
Code Analysis	Einstellungen zur statischen Codeanalyse (FxCop-Regeln)	Einstellungen zur statischen Codeanalyse (FxCop-Regeln)
My Extensions	<i>Nicht verfügbar</i>	Einbinden von Erweiterungen für den My-Namensraum

Tabelle 5.6 Ausgewählte Registerkarten in den Projekteigenschaften

Anwendung

Konfiguration: Nicht zutr. Plattform: Nicht zutr.

Erstellen

Buildereignisse

Debuggen

Ressourcen

Dienste

Einstellungen

Verweispfade

Signierung

Sicherheit

Veröffentlichen

Codeanalyse

Assemblyname: WorldWideWings_WindowesUI

Standardnamespace: WindowsUI

Zielframework: .NET Framework 3.5

Ausgabebetyp: Windows-Anwendung

Startobjekt: (Nicht festgelegt)

Assemblyinformationen...

Ressourcen

Geben Sie an, wie die Anwendungsressourcen verwaltet werden sollen:

☒ Symbol und Manifest

Mit einem Manifest werden bestimmte Einstellungen für eine Anwendung festgelegt. Um dieses Manifest einzubetten, fügen Sie es dem Projekt hinzu und wählen es dann in der Liste unten aus.

Symbol: Media\worldwidewings.ico

Manifest: Anwendung ohne Manifest erstellen

☐ Ressourcendatei:

Abbildung 5.14 Projekteigenschaften eines C#-Projekts (Windows Forms)

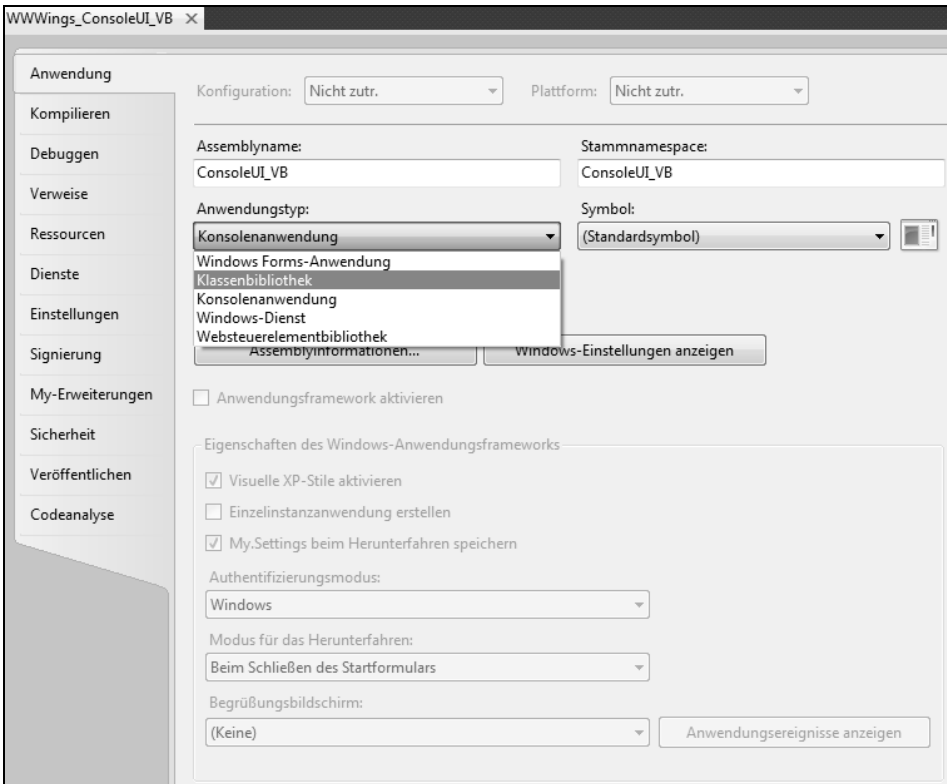


Abbildung 5.15 Projekteigenschaften eines Visual Basic-Projekts (Windows Forms)

Verweise (Komponenten- und Projektreferenzen)

Um einen Typ aus einer anderen Assembly nutzen zu können, benötigt der Compiler eine Referenz auf die Assembly, die diesen Typ implementiert. Diese Verweise sind nicht mehr wie einst in Visual Studio 6.0 in einem Menü versteckt, sondern gut sichtbar in den Projektmappen-Explorer integriert (Ordner *Verweise* bzw. *References*). Ein Projekt kann Verweise zu einer Assembly des .NET Framework, zu anderen .NET- oder COM-Komponenten, zu anderen Visual Studio-Projekten oder zu XML-Webservices haben.

Arten von Verweisen

Für jeden Projekttyp ist eine bestimmte Menge von Verweisen auf verschiedene Framework-Assemblies automatisch vordefiniert.

- Über den Kontextmenüpunkt *Verweis hinzufügen* (*Add Reference*) im Ordner *Verweise* (*References*) können Verweise hinzugefügt werden
- Über *Serviceverweis hinzufügen* (*Add Service Reference*) kann man Webservices einbinden: Visual Studio generiert automatisch eine Wrapper-Klasse, sodass der Zugriff auf den Webservice völlig transparent ist

Verbesserungen des Dialogfelds

Das Dialogfeld *Add Reference* (in der deutschen Version: Verweis hinzufügen) ist seit Langem der oberste »Produktivitätskiller« bei der Arbeit mit Visual Studio, denn die Anzahl der Assemblys im .NET Framework ist stetig gestiegen, was die Auswahlliste immer länger und unübersichtlicher machte. Zudem dauerte das erste Laden der Liste immer länger (mehrere Sekunden).

Die Veränderung, die Microsoft zunächst in Visual Studio 2010 vorgenommen hat, ist jedoch eindeutig eine Verschlimmbesserung. Unter dem Vorwand der Geschwindigkeitssteigerung lädt Microsoft die Inhalte nun asynchron. Statt eines Geduldsspiels ist die Auswahl nun aber ein Geschicklichkeitsspiel, denn das Dialogfeld »verblättert« durch das asynchrone Laden die aktuelle Position immer wieder. Hier wäre ein einfaches Suchdialogfeld wie der, den man nun in der Auswahl der Projekt- und Projektelementvorlagen findet, eine echte Hilfe gewesen. Aber suchen kann man hier leider immer noch gar nicht.

Daher sollte man unbedingt die Productivity Power Tools installieren. Hier ist ein ganz neues Referenzdialogfeld mit zwei wesentlichen Verbesserungen enthalten:

- Die gefundenen Assemblys werden zwischengespeichert durch Caching. Das Geschicklichkeitsspiel beim asynchronen Laden entfällt daher.
- Es gibt ein Suchdialogfeld!

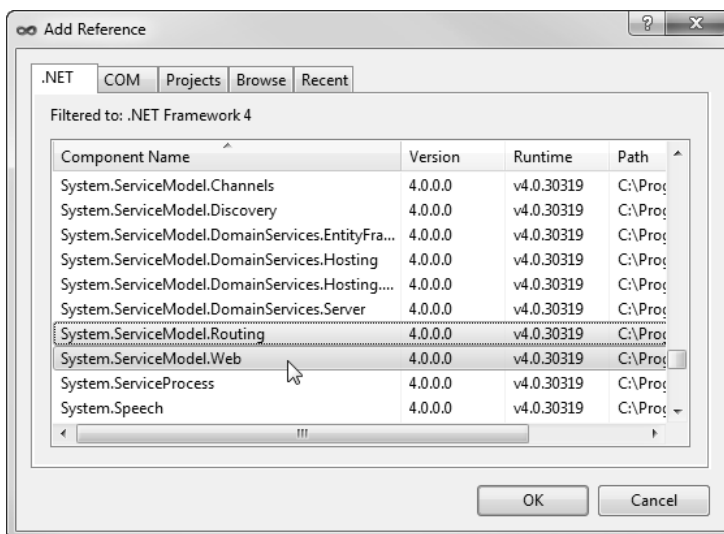


Abbildung 5.16 Das *Add Reference*-Dialogfeld in Visual Studio 2010 ohne Power Tools

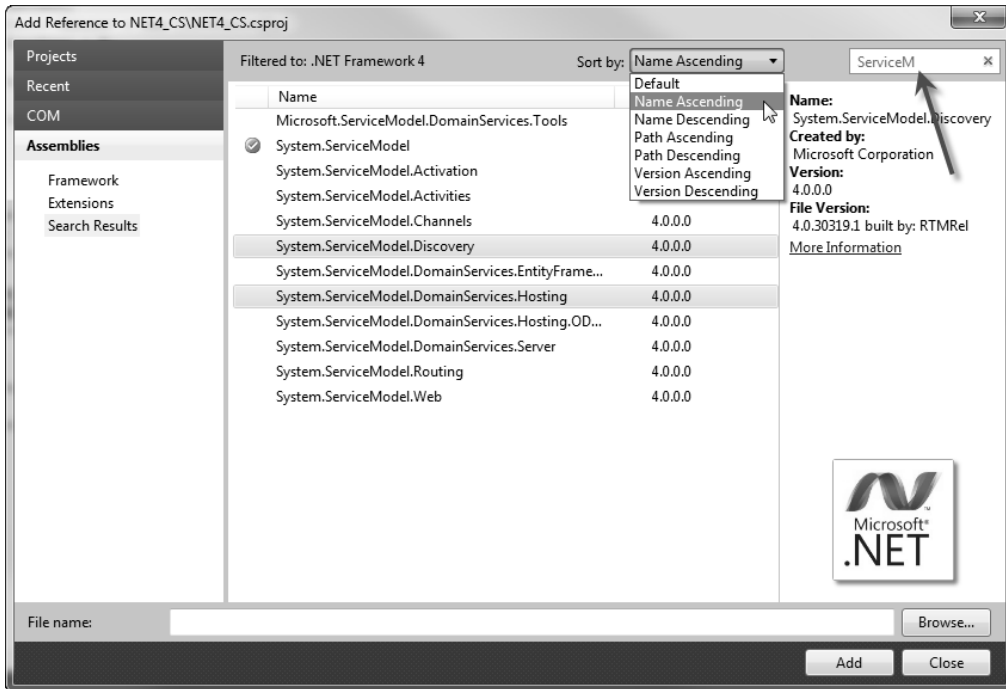


Abbildung 5.17 Das neue *Add Reference*-Dialogfeld aus den Visual Studio 2010 Productivity Power Tools

HINWEIS Webprojekte können nur Verweise auf DLL-Assemblys besitzen. Konsolen- und Windows-Anwendungen können auch auf EXE-Assemblys verweisen. Webprojekte nach dem Websitemodell zeigen die Verweise nicht in einem eigenen Ast im Projektmappen-Explorer an, sondern nur in den Projekteigenschaften.

TIPP Die in dem Dialogfeld *Verweise* automatisch gezeigte Liste der referenzierbaren Assemblys kann man erweitern durch Setzen eines Registrierungsdatenbank-Schlüssels:

```
HKEY_CURRENT_USER\SOFTWARE\Microsoft\ .NETFramework\AssemblyFolders\WWWings]@=
"H:\WWWings\Komponenten"
```

Diesen Schlüssel kann man auch auf Ebene von HKEY_LOCAL_MACHINE global für alle Benutzer setzen.

Speichern und Autowiederherstellung

Neu seit Visual Studio 2005 ist eine automatische Speicherfunktion für Projekte, die unter *Extras/Optionen/Umgebung/Auto-Wiederherstellen* konfiguriert werden kann.

Im Gegensatz zum Vorgänger können seit Visual Studio 2005 einige Projektarten (z.B. Windows Forms und Konsolenanwendungen) auch geöffnet, bearbeitet und kompiliert werden, ohne sie zu speichern. Dies ist von Vorteil, wenn ein Entwickler einmal kurz etwas ausprobieren möchte, ohne sich die Festplatte mit nicht mehr benötigten Projekten zu verunreinigen (Option *Extras/Optionen* → *Projekte und Projektmappen* → *Allgemein* → *Neue Projekte beim Erstellen speichern*).

Projektkonvertierung (Migration)

Projekte aus Visual Studio 2002/2003/2005/2008 müssen beim Öffnen in Visual Studio 2010 konvertiert werden, weil sich das Projektformat zwischen jeder Version geändert hat. Sie können danach nicht mehr in der alten Version geöffnet werden. Man sollte also unbedingt eine Sicherungskopie der alten Projektdaten anlegen. Außerdem muss man bedenken, dass alle Projektmitglieder gleichzeitig auf Visual Studio 2010 umsteigen müssen. Visual Studio 2010 bietet zur Konvertierung der Projekte einen Konvertierungsassistenten (Visual Studio Conversion Wizard).

TIPP

Da Visual Studio .NET 2003, Visual Studio 2005, Visual Studio 2008 und Visual Studio 2010 auf einem System koexistieren können, ist es problemlos möglich, zunächst einige Projekte umzustellen, während man in anderen Projekten noch in der älteren Version arbeitet.

Konvertierungsassistent

Das Ergebnis der Konvertierung wird immer an dem ursprünglichen Speicherort abgelegt und die vorherige Version wird überschrieben (In-Place-Konvertierung). Der Konvertierungsassistent bietet an, eine Sicherungskopie der Anwendung an einem anderen Ort zu erstellen. Die Konvertierung ist an der Kommandozeile möglich mit *devenv.exe Projektdatei /upgrade*.

Der Konvertierungsassistent legt ein Konvertierungsprotokoll an. Das Konvertierungsprotokoll wird als Textdatei (*ConversionReport.txt*) und XML-Datei (*UpgradeLog.xml*) abgelegt. Die XML-Datei wird in Visual Studio als HTML-Datei angezeigt.

Die Breaking Changes zwischen den .NET-Versionen können zu Kompilierungsfehlern oder Änderungen im Verhalten führen. Hier muss der Entwickler eingreifen. Darüber hinaus wird er zahlreiche Warnungen sehen, dass einige der verwendeten Klassen, Methoden, Attribute und Ereignisse inzwischen »obsolete« sind, weil es dafür neuere Implementierungen gibt. Der Entwickler kann den Code in diesen Punkten ändern oder aber die Warnungen ignorieren.



Abbildung 5.18 Visual Studio-Konvertierungsassistent

ACHTUNG Bei der Umstellung zwischen den Version 2005/2008 und 2010 ändert der Konvertierungsassistent nur das Format der Projektdateien (*.vbproj*, *.csproj*). Bei der Umstellung von der 2003er Version erfolgt – bei einigen Projektarten – auch ein Eingriff in den von den Designern generierten Programmcode. Unangetastet lässt der Assistent die verwendeten Steuerelemente. So bleiben Steuerelemente wie DataGrid, MainMenu und StatusBar erhalten, auch wenn es dafür ab .NET 2.0 neuere Steuerelemente wie GridView, MenuStrip und StatusStrip gibt. Diese Migration, die sehr aufwendig sein kann, muss der Entwickler von Hand vornehmen.

Erfahrungen

Bei der Umstellung von mehreren Projekten von Visual Studio 2008 auf Visual Studio 2010 konnte beobachtet werden:

- Der Konvertierungsassistent läuft ohne Probleme durch (getestet mit vielen verschiedenen Projektvorlagen, aber sicherlich nicht mit allen möglichen)
- Der Konvertierungsassistent fragt bei Webprojekten nach, ob er diese auf ASP.NET 4.0 hochstufen soll. Er entfernt dann Einträge aus der *web.config*-Datei, die in ASP.NET 4.0 nicht mehr notwendig sind. Um die Kompatibilität beim Rendering zu setzen, wird jedoch festgelegt (siehe auch Kapitel zu ASP.NET 4.0):

```
<pages controlRenderingCompatibilityVersion="3.5" .../>
```

- VSTO-Projekte mit Microsoft Office 2003 können nicht konvertiert werden, weil Visual Studio 2010 nur noch Microsoft Office 2007 und höher unterstützt (Meldung: »You do not have a version of Office that is supported with this version of Visual Studio. Please install Office 2007 or greater then try again.«)
- Alle anderen Projekttypen werden auf dem bisherigen *TargetFramework* belassen
- Microsoft hat einige Klassen in andere DLLs verschoben, z.B. die Basisklasse *MembershipProvider* von *System.Web* nach *System.Web.ApplicationServices*. Man muss nun händisch entsprechend neue Assembly-Referenzen setzen. Man erhält aber entsprechende Fehlermeldungen beim Kompilieren.

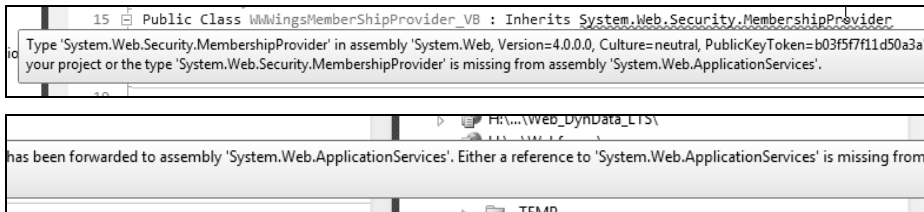


Abbildung 5.19 Aufforderung zum Setzen einer neuen Assembly-Referenz nach der Umstellung auf .NET 4.0

- Manche Projekte kompilieren trotzdem nicht, weil einige Assembly-Referenzen nicht mehr funktionieren. (z.B. System.ServiceModel und System.Workflow.Activities und System.Web.Mobile). Wenn man diese neu einbindet, geht es wieder!
- Der Microsoft-ReportViewer (Microsoft.ReportViewer.Common.dll, Microsoft.ReportViewer.WebForm.dll, Microsoft.ReportViewer.WinForms.dll) wird in Version 10.0 mitgeliefert. Alle bestehenden Referenzen auf die 9.0-Komponenten müssen manuell aktualisiert werden.

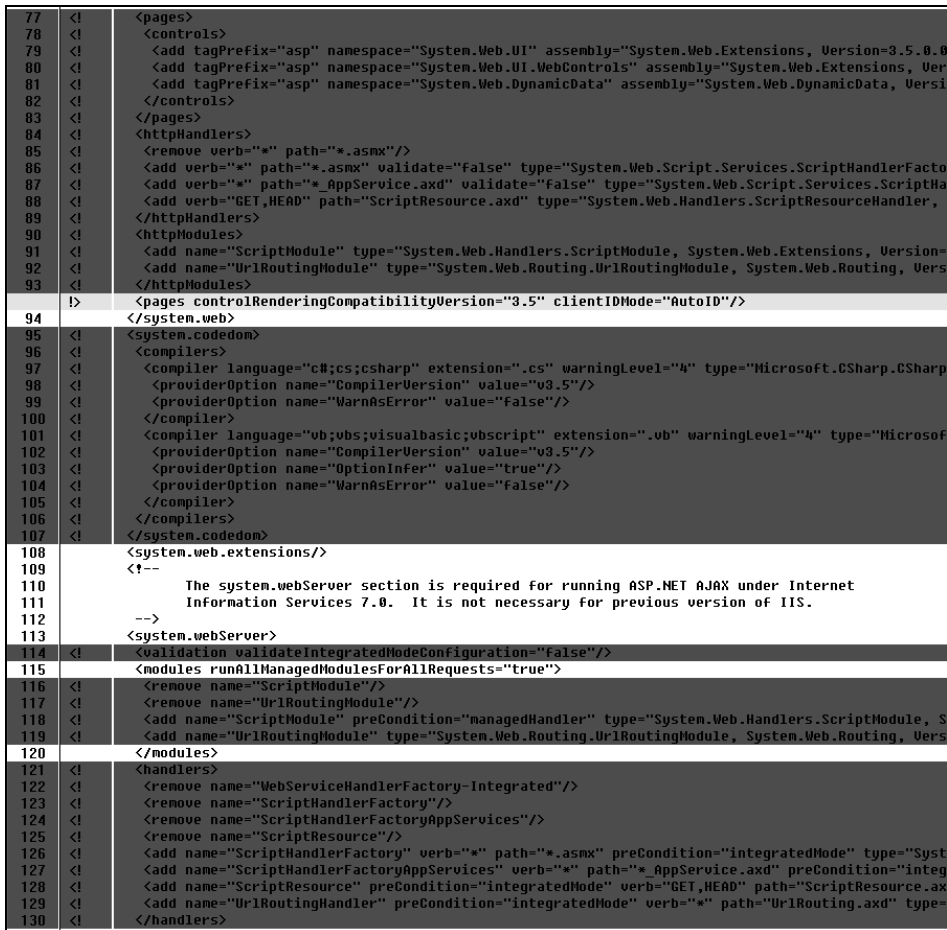


Abbildung 5.20 WinDiff beweist, dass der Visual Studio 2010-Migrationsassistent in den *web.config*-Dateien aufgeräumt hat

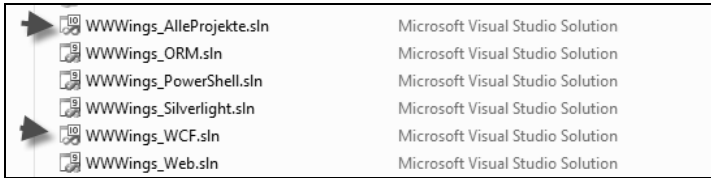


Abbildung 5.21 Die Dateinamenserweiterungen sind gleich geblieben, aber an den neuen Icons in blau und lila erkennt man Projekte, die mit Visual Studio 2010 erstellt oder dorthin konvertiert wurden

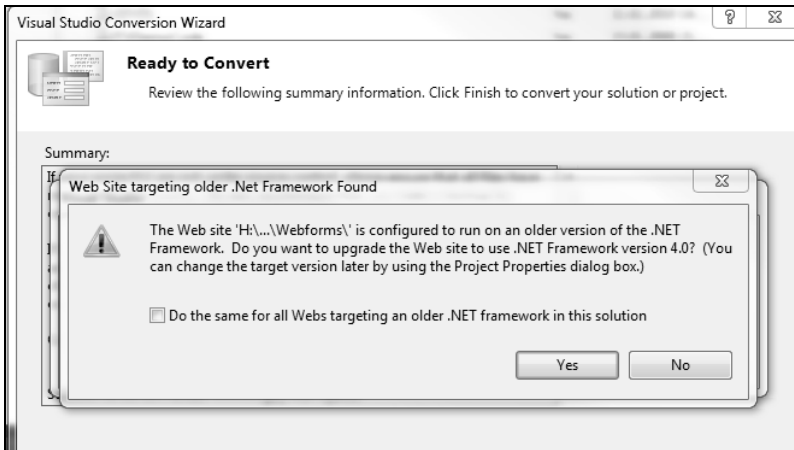


Abbildung 5.22 Nachfrage von Visual Studio 2010 bei der Umstellung eines Webprojekts auf .NET 4.0

Code-Editoren

Visual Studio 2010 bietet Code-Editoren mit zahlreichen Eingabehilfen an. Die Hilfeleistungen seitens des Code-Editors unterscheiden sich aber zwischen den .NET-Sprachen, so wie man es auch schon von früheren Visual Studio-Versionen kennt.

Farbdarstellung

Die Editoren bieten eine farbliche Unterscheidung verschiedener Programmcode-Elemente, z. B.:

- Schlüsselwörter der Sprache sind blau
- Kommentare sind grün
- Zeichenketten sind rot
- Bezeichner der Klassenbibliothek und eigene Bezeichner sind schwarz

TIPP

Die Farben sind über *Extras/Optionen/Umgebung/Schriftarten und Farben* konfigurierbar.

Hervorheben der aktuellen Zeile durch grauen Balken

In den optionalen Visual Studio 2010 Productivity Power Tools ist eine Erweiterung des Editors enthalten, die die Zeile, in der sich der Cursor befindet, grau hervorhebt (*Highlight Current Line*). Diesen Balken sieht man aber nur, solange kein Text ausgewählt ist.

```
public class Person
{
    public string Name;
    public string Straße;
    public Gemeinde Gemeinde;
}
```

Abbildung 5.23 Hervorhebung der aktuellen Zeile in VS 2010 mit Productivity Power Tools

Änderungsverfolgung

Visual Studio bietet seit Version 2005 durch farbliche Markierungen am linken Rand eine einfache Form der Änderungsverfolgung:

- Ein gelber Balken bedeutet, dass diese Zeile seit dem Öffnen der Datei verändert wurde
- Ein grüner Balken heißt, dass diese Zeile seit dem Öffnen der Datei verändert und die Veränderungen auch gespeichert wurden

Die Balken bleiben so lange erhalten, bis die Datei geschlossen wird. Ein grüner Balken kann wieder gelb werden, wenn die Zeile erneut geändert wird.

Zeilennummern und Zeilenumbruch

Ob Zeilennummern im Editor dargestellt und überlange Zeilen umgebrochen werden, lässt sich unter *Extras/Optionen/Text-Editor* pro Sprache oder für alle Sprachen gleichzeitig einstellen.

Änderung der Schriftgrößen durch Mausrad

Man sieht Visual Studio 2010 kaum an, dass es nun auf der Windows Presentation Foundation (WPF) basiert. WPF hat aber ermöglicht, dass man nun die Schriftgröße im Codeeditor mit der `[Strg]`-Taste und Mausrad nahtlos vergrößern kann. Das ist sehr vorteilhaft für Präsentationen oder wenn ein Kollege mal über die Schulter auf den Code schauen soll.

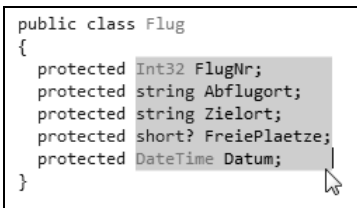
Auswahlfunktionen

Schon bisher kann man mit gedrückter Maustaste einen Bereich markieren. Durch Visual Studio 2010 sind zwei Möglichkeiten hinzugekommen:

- Durch schnelles dreimaliges Klicken wird die ganze Zeile ausgewählt. Voraussetzung sind auch hier die Visual Studio 2010 Productivity Power Tools. Man muss relativ schnell nacheinander klicken, auf jeden Fall schneller als in Microsoft Word, wo man ja seit Langem schon mit Dreifachklick einen Absatz markieren kann.

- Bisher kann man die Definition eines Bezeichners auffinden, indem man im Kontextmenü *Go to Definition* wählt oder **F12** drückt. Nun kann man alternativ – wenn die Visual Studio 2010 Productivity Power Tools installiert sind (!) – den Sprung über die Maus durchführen, wenn man gleichzeitig beim Links-Klick die **Strg**-Taste drückt. Wenn man nur die **Strg**-Taste drückt, sieht man bei selbstdefinierten Bezeichnern einen Hyperlink. Bei den Bezeichnern des .NET Framework erscheint kein Hyperlink, der Sprung ist aber nach dem Links-Klicken dennoch möglich.
- Durch Installieren der Visual Studio 2010 Productivity Power Tools übernimmt Visual Studio beim Kopieren von Text aus dem Visual Studio-Editor nun auch eine HTML-Version des formatierten Programmcodes. Beim Einfügen in einen HTML-Editor erhält man dann eine HTML-formatierte Version des Programmcodes.
- In Visual Studio 2010 kann man mit gedrückter **Alt**-Taste einen rechteckigen Bereich auswählen und in die Zwischenablage übernehmen.

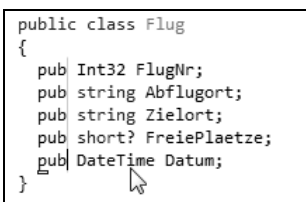
Die Rechteckauswahl macht zum Beispiel in der folgenden Bildschirmabbildung Sinn, wenn man die Variablendeklaration ohne *protected* kopieren möchte.



```
public class Flug
{
    protected Int32 FlugNr;
    protected string Abflugort;
    protected string Zielort;
    protected short? FreiePlaetze;
    protected DateTime Datum;
}
```

Abbildung 5.24 Rechteckauswahl (engl. Box Selection)

Noch interessanter ist, dass man einen rechteckigen Auswahlbereich auch ändern kann. In dem obigen Beispiel könnte man alle *protected*-Wörter markieren und müsste dann nur einmal *public* tippen, um alle zu ändern.



```
public class Flug
{
    pub Int32 FlugNr;
    pub string Abflugort;
    pub string Zielort;
    pub short? FreiePlaetze;
    pub DateTime Datum;
}
```

Abbildung 5.25 Während des Massenänderns von *protected* in *public* nach Rechteckauswahl

Angleichen von Zuweisungen

TIPP Wenn Sie bei installierten Productivity Power Tools einen Zuweisungsblock markieren und dann **STRG** + **ALT** + **'** drücken, wird Visual Studio alle Gleichheitszeichen untereinander anordnen, sodass der Codeblock übersichtlicher ist.

Verschieben einer Auswahl mit Cursortasten

Durch die Visual Studio 2010 Productivity Power Tools kann man einen ausgewählten Text mit den Tastenkombination `[Alt] + [↑]` und `[Alt] + [↓]` nach oben oder unten verschieben.

IntelliSense-Funktionen

Unter dem Begriff IntelliSense fasst Microsoft zahlreiche Eingabehilfen für Programmcode zusammen. Schon beim Eintippen des ersten Buchstabens eines neuen Words zeigt Visual Studio ein Auswahlmenü mit Sprachsyntaxelementen, Klassen und vordefinierten Codefragmenten.

Das Menü *Bearbeiten/IntelliSense* offenbart, dass C# mehr IntelliSense-Funktionen als VB besitzt.

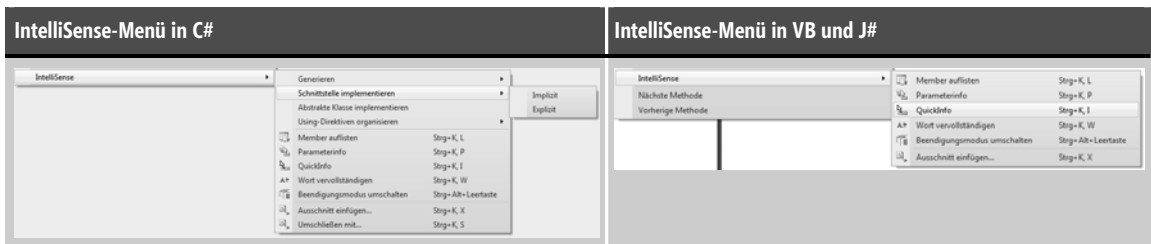


Tabelle 5.7 IntelliSense in den verschiedenen .NET-Sprachen

IntelliSense-Funktionen in Visual Studio .NET 2003 waren:

- Auflisten der Mitglieder eines Objekts (*Member auflisten / List Members*)
- Anzeige der Parameter einer Methode (*Parameterinfo / Parameter Info*)
- Anzeige eines Tooltip, der Typdeklaration bzw. Methodensignatur und oft auch einen kurzen Hinweis-text enthält (*Quickinfo / Quick Info*)
- Anzeige einer Liste der verfügbaren Befehlswörter, Typen und Mitglieder (*Wort vervollständigen / Complete Word*)

Neu seit Visual Studio Version 2005 sind folgende Funktionen:

- Einfügen von vordefinierten Codefragmenten aus einer erweiterbaren Quellcodebibliothek (*Ausschnitt hinzufügen / Insert Snippet*)
- Verpacken eines Codefragments in ein Blockkonstrukt (*Umschließen mit / Surround With*)
- Erzeugen eines Methodenrumpfs aus einem Methodenaufruf (*Methodenstub generieren / Generate Method Stub*)
- Einfügen der Rumpfe für eine Schnittstelle (*Schnittstelle implementieren / Implement Interface*)
- Einfügen der Rumpfe für eine abstrakte Klasse (*Abstrakte Klasse implementieren / Implement Abstract Class*)
- Nach Eingabe des Modifikators `override` bzw. `overrides` zeigt Visual Studio ein Auswahlmenü der überschreibbaren Methoden an und vervollständigt die Methodensignatur nach der Auswahl.

Neu seit Visual Studio Version 2008 sind folgende IntelliSense-Funktionen:

- IntelliSense für JavaScript-Quellcode (in SP1 nochmals verbessert)

- IntelliSense für LINQ-Befehle
- IntelliSense für eingebettete XML-Fragmente (nur Visual Basic)
- Beim Drücken der **[Strg]**-Taste wird das IntelliSense-Fenster transparent, sodass man den Quellcode darunter erkennen kann
- Während der Eingabe reduziert sich die Auswahl im IntelliSense-Fenster (nur Visual Basic)

Neben den im *IntelliSense*-Menü angezeigten Funktionen bietet Visual Studio noch einige etwas versteckte Funktionen, die durch die **[F12]**-Taste aktiviert werden und daher als *Tabular IntelliSense* bezeichnet werden:

- Generierung einer Ereignisbehandlungsroutine für ein Ereignis
- Einfügen von Snippets nach Eingabe bestimmter Schlüsselwörter

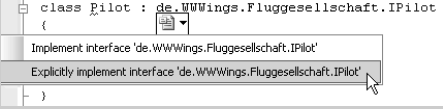

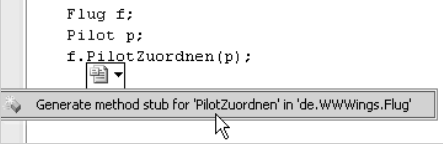


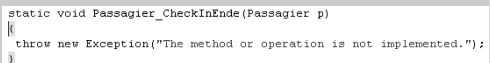
Funktion	Ihre Aktion	Was Visual Studio für Sie erledigt
Schnittstelle implementieren/ Abstrakte Klasse implementieren (Implement Interface/ Implement Abstract Class)	Geben Sie nach einer Klassendefinition hinter dem Doppelpunkt eine Schnittstelle oder abstrakte Klasse an. Öffnen Sie das Smarttag der Schnittstelle der abstrakten Klasse oder wählen Sie <i>Implement</i> im <i>IntelliSense</i> -Menü. 	<pre>class Pilot : de.WWWings.Fluggesellschaft.IPilot { #region IPilot Members DateTime de.WWWings.Fluggesellschaft.IPilot.FlugscheinZeit { get { throw new Exception("The method or operation is not implemented."); } set { throw new Exception("The method or operation is not implemented."); } } string de.WWWings.Fluggesellschaft.IPilot.FlugscheinTyp { get { throw new Exception("The method or operation is not implemented."); } } }</pre>
Umschließen mit (Surround With)	Markieren Sie eine oder mehrere Zeilen. Wählen Sie im Kontextmenü oder im IntelliSense-Menü <i>Surround With</i> . 	<pre>public long Add(Passagier P) { try { long pnr = this.NaechstePassagierNummer(); base.Add(pnr, P); return pnr; } catch (Exception) { throw; } }</pre>
Methodenstub generieren (Generate Method Stub)	Geben Sie einen Methodenaufruf für eine Methode ein, die nicht existiert. Öffnen Sie das Smarttag der Methode oder wählen Sie <i>Generate Method Stub</i> im <i>IntelliSense</i> -Menü. 	Visual Studio fügt einen Methodenrumpf für die Methode in die entsprechende Klasse ein, sodass die Anwendung kompiliert werden kann. <pre>public partial class Flug { public void PilotZuordnen(Pilot p) { throw new Exception("The method or operation is not implemented."); } }</pre>
Generierung einer Ereignisbehandlungsroutine für ein Ereignis	Geben Sie nach einem Ereignisnamen += ein und drücken Sie zweimal [F12] . 	Ergebnis nach dem ersten Drücken von [F12] :  Ergebnis nach dem zweiten Drücken von [F12] : 

Tabelle 5.8 Hinweise zur Nutzung verschiedener IntelliSense-Funktionen in Visual Studio

Neu seit Visual Studio Version 2010 sind folgende IntelliSense-Funktionen:

- Eine zentrale Neuerung in Visual Studio 2010 bemerkt man schon nach den ersten getippten Buchstaben: Die IntelliSense sucht nicht mehr wie bisher nur Wortanfänge, sondern führt eine Teilstringsuche aus. Gibt man **Consol.Col** ein, bietet Visual Studio nun die Attribute *BackgroundColor* und *ForegroundColor* sowie die Methode *ResetColor* an. Oder Sie wollen einen Fehler auslösen. Sie wissen, dass alle Fehlerklassen auf dem Wort *Exception* enden. Bisher war dabei die IntelliSense gar nicht behilflich. Nun gibt es eine Liste aller erreichbaren Fehlerklassen.
- Die Eingabe von **Console.WL** bietet *Console.WriteLine* und *Console.WindowLeft* (Suche anhand der Anfangsbuchstaben!) an.
- Für die testgetriebene Entwicklung gibt es einen neuen IntelliSense-Modus. Mit der Tastenkombination `[Strg] + [Alt] + []` versetzt man Visual Studio in einen Modus, in dem die Entwicklungsumgebung nach dem Drücken der `[]`-Taste nicht mehr die Eingabe zu vervollständigen versucht, sondern in Hinblick darauf, dass es gewollt ein neuer Bezeichner sein könnte, der die angefangene Zeichenkette wie eingegeben stehen lässt.

Blockvervollständigung

Der Visual Basic-Editor schont die Fingerkuppen des Entwicklers schon länger dadurch, dass er beim Eingeben eines Blockkonstrukts (Sub, Function, For, Do etc.) automatisch das passende Blockende (End, Next) mit einfügt. Bei einigen Fehlern macht der Code-Editor Verbesserungsvorschläge (siehe Abbildung 5.26).



Abbildung 5.26 Fehlerbehebungsvorschlag im Editor für Visual Basic

TIPP

Um in C# in den Genuss zu kommen, dass auch die schließenden geschweiften Klammern automatisch eingefügt werden, muss man die Productivity Power Tools installieren.

Generierung von bisher nicht deklarierten Klassen und Klassenmitgliedern bei ihrer ersten Verwendung

Eine wichtiges Thema bei der Verbesserung des Editors in Visual Studio 2010 war das Test Driven Development (TDD), bei dem man Klassen und Klassenmitglieder aufruft, die man noch gar nicht deklariert hat. Schon seit Visual Studio 2005 kann man Methodenaufrufe für nicht vorhandene Methoden eingeben und sich dann von der Entwicklungsumgebung eine passende Methode generieren lassen (*Generate Method Stub* im Kontextmenü). Nun kann man auch Klassen und Attribute auf diese Weise deklarieren.

Bei Klassen besteht die Wahl, eine *Standardklasse* (private Klasse im gleichen Projekt und Projektordner wie der aktuelle Standort) zu erzeugen oder über das Dialogfeld *Generate New Type* eine individuellere Auswahl

zu treffen. Bei Attributen hat man die Wahl zwischen einem Field oder einem Property, wobei Letzteres als so genanntes *Automatic Property*, also ohne explizites Field erzeugt wird. Dieses in C# 3.0 eingeführte prägnante Sprachkonstrukt gibt es nun ab Version 2010 auch für Visual Basic.

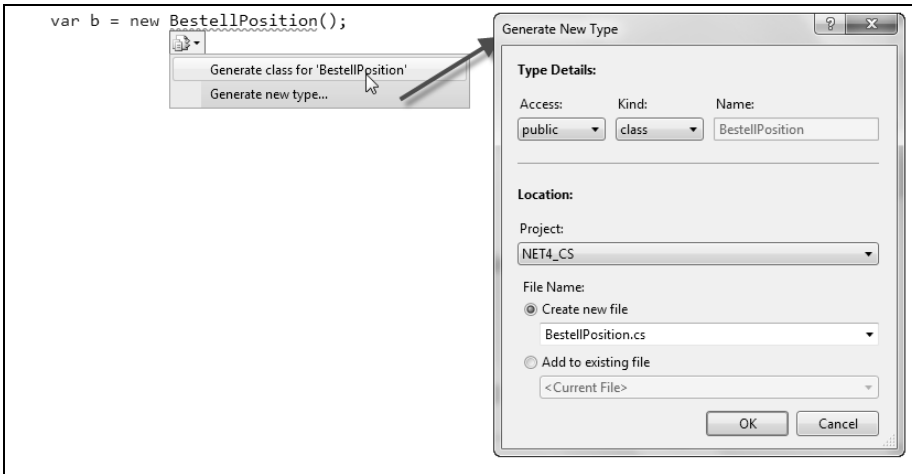


Abbildung 5.27 Erzeugen einer neuen Klasse aus ihrer Verwendung heraus


Codebibliothek (Code Snippets)

Visual Studio bietet die Möglichkeit, vordefinierte Codefragmente in die Codedateien einzufügen. Die Funktion wird aktiviert durch den Kontextmenüeintrag *Ausschnitt hinzufügen* im Code-Editor oder durch das Menü *Bearbeiten/IntelliSense/Ausschnitt hinzufügen*. Die Auswahl in VB ist so viel reichhaltiger als in C#, dass die Snippets in zahlreichen Gruppen organisiert sind. Viele Snippets enthalten Platzhalter, die mit der -Taste direkt angesprungen werden können. Früher hießen die *Snippets* noch *Expansion*.

HINWEIS Auch kurios bleibt, dass C# weiterhin bedeutend weniger von den *Snippets* genannten Codefragmenten besitzt als Visual Basic. In der Vergangenheit musste der Entwickler weitere Code-Snippets für C# einzeln herunterladen und das gilt leider auch für die neue Version [<http://msdn.microsoft.com/en-us/vstudio/aa718338.aspx>]. Neu in Visual Studio 2010 ist, dass es Code-Snippets auch für HTML, ASP.NET und JavaScript gibt. Die Liste der Code-Snippets ist wie bisher auch schon erweiterbar.

Code-Snippets in C#	Code-Snippets in VB
<p>Ausschnitt einfügen: Visual C# > </p> <ul style="list-style-type: none"> enum equals exception for foreach forr if indexer interface <p>Codeausschnitt für Ausnahme Verknüpfung: Exception</p>	<p>Konnektivität, Sicherheit, Workflow > Konnektivität und Netzwerk > </p> <ul style="list-style-type: none"> Andere Computer prüfen Bestimmen, ob das Netzwerk verfügbar ist Daten mithilfe von My Computer Netzwerk hochladen Daten über HTTP hochladen Daten von einem anderen Anschluss lesen E-Mail-Nachricht erstellen Relativen URI in absoluten URI konvertieren SerialPort verwenden, um eine Telefonnummer zu wählen Serial-Anschlüsse auflisten <p>Bestimmt, ob die angegebene Website reagiert, Verknüpfung: config</p>

Tabelle 5.9 Code-Snippets in C# und Visual Basic

TIPP Sie können über das Menü *Extras/Codeausschnittmanager (Tools/Code Snippet Manager)* eigene Snippets einpflegen und dadurch eine eigene Codefragment-Datenbank unterhalten. Die eigenen Snippets speichert Visual Studio im Benutzerprofil im Ordner */Meine Dokumente/Visual Studio 2010/Code Snippets*. In C# und J# können Snippets auch durch die Eingabe von Kürzeln (z. B. mbox, tryf und ctor) mit anschließendem zweimaligen  eingefügt werden.

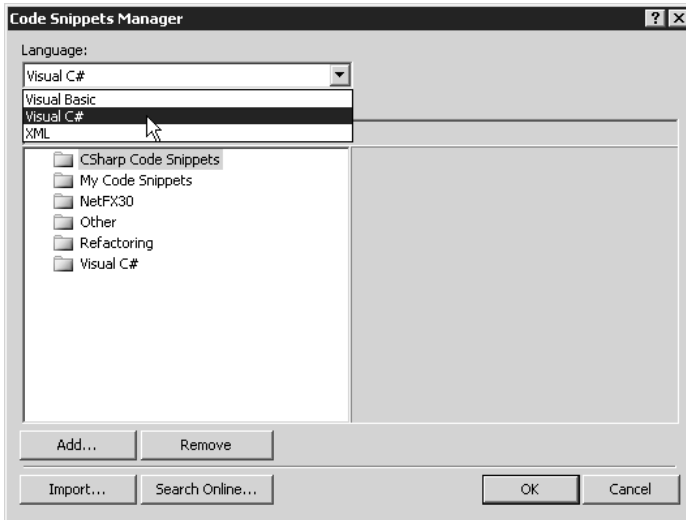








Abbildung 5.28 Code Snippet Manager

Coderegionen

In den Visual Studio-Code-Editoren kann der Entwickler die Implementierung von einzelnen Klassen und Unterrouinen durch die Zeichen  und , ähnlich wie bei den Verzeichnissen im Windows Explorer verbergen. Es bleibt dann nur der Klassen- bzw. Unterrouinenrumpf stehen mit einem Verweis, dass die Implementierung verborgen ist. Mit der Compiler-Direktive `#Region...#End Region` kann der Entwickler selbst eigene zuklappbare Code-Abschnitte definieren.

HINWEIS Bisher konnte man Namensräume, Klassen und Methoden einklappen sowie Bereiche, die man mit `#Region...#End Region` markiert hat. Nun kann man auch mit der Tastenfolge  gefolgt von  eine beliebige Zeilenfolge einklappen. Aufheben kann man dies wieder durch  gefolgt von . Alternativ kann man das Kontextmenü *Outlining* verwenden. Die Einklappungsinformation speichert Visual Studio in der *.suo*-Datei, die benutzerspezifische Konfigurationsinformationen enthält und im Standard nicht im Team Foundation Server gespeichert wird. Daher gelten diese Einklappungen nur für den einen Benutzer auf einem PC.

Aufgeklappte Region	Zugeklappte Region
<pre>#region Konstruktoren // ===== Konstruktor 1 public Flug(string FlugNr, string AbflugOrt, string ZielOrt) { this.FlugNr = FlugNr; this.AbflugOrt = AbflugOrt; this.ZielOrt = ZielOrt; if (Flug.Fluege.ContainsKey(this.FlugNr.ToString())) Flug.Fluege.Remove(this.FlugNr.ToString()); Flug.Fluege.Add(this.FlugNr.ToString(), this); } // ===== Konstruktor 2 public Flug(string FlugNr, string AbflugOrt, string ZielOrt, DateTime Datum) : this(FlugNr, AbflugOrt, ZielOrt) { this.Datum = Datum; } } #endregion</pre>	<div>Konstruktoren</div>

Tabelle 5.10 Regionen im Editor

TIPP

Im C#-Editor existiert mit der Funktion *Umschließen mit (Surround With)* eine einfache Möglichkeit, ein Codefragment in eine Region zu verpacken.

Refactoring (Umgestalten)

Visual Studio bietet ein paar hilfreiche Funktionen zum nachträglichen Umgestalten von Programmcode (Refactoring) – allerdings im Standardlieferungsumfang nur für die Sprache C#. Für Visual Basic ist ein kostenloses Add-on verfügbar (siehe unten).

HINWEIS

Die Refactoring-Funktionen sind in Visual Studio 2010 genauso schwach wie zuvor. Ebenso hat sich nichts daran geändert, dass die Refactoring-Funktionen für Visual Basic nicht zum Standardlieferungsumfang von Visual Studio gehören. Zu erwarten wäre gewesen, dass Microsoft wieder über einen Vertrag mit der Firma Developer Express einen Teil deren Refactoring-Funktionen einkauft und kostenfrei an die Visual Basic-Nutzer weitergibt. Dies ist jedoch bis zum Redaktionsschluss (5.10.2010) nicht erfolgt.

Refactoring für C#

Die über das Menü *Umgestalten (Refactor)* verfügbaren Funktionen für C# zeigt die nachstehende Tabelle.

Befehl	Verfügbar für	Aktionen
Umbenennen (Rename)	Typ, Klassenmitglied	Umbenennen eines Typs und eines Typmitglieds. Alle Codezeilen im Projekt, die den Typ nutzen, werden automatisch geändert. Umbenennen ist kein Suchen und Ersetzen, sondern ein »intelligentes« Umbenennen, das die Namensraumzugehörigkeit berücksichtigt. Auch Code in referenzierenden Projekten wird geändert, wenn die Projekte in der gleichen Projektmappe geöffnet sind.
Feld enkapseln (Encapsulate Field)	Field	Erzeugen einer <i>Getter</i> - und <i>Setter</i> -Methode für ein vorhandenes einfaches Attribut in Form eines <i>field</i> ; ändert die Sichtbarkeit für das ursprüngliche Mitglied von <i>Public</i> auf <i>Private</i>
Methode extrahieren (Extract Method)	Codefragment	Auslagern des Codefragments in eine Methode. Alle in das Codefragment eingehenden Variablen werden automatisch zu Methodenparametern.
Schnittstelle extrahieren (Extract Interface)	Klasse	Erzeugen einer Schnittstelle mit auswählbaren Mitgliedern der Klasse

Befehl	Verfügbar für	Aktionen
Lokale Variable auf Parameter hinaufstufen (Promote Local Variable to Parameter)	Variablendeklaration mit Initialisierung in einer Methode	Variable wird zum Parameter der Methode; Initialisierung wird verworfen
Parameter entfernen (Remove Parameters)	Methoden, Konstruktoren, Indexer, Delegat	Entfernen eines Parameters aus dem Methodenrumpf. Bei allen Aufrufen wird der Parameter auch entfernt.
Parameter neu anordnen (Reorder Parameters)	Methoden, Konstruktoren, Indexer, Delegat	Änderung der Parameterreihenfolge. Bei allen Aufrufen wird die Parameterreihenfolge auch geändert.
Organize Usings	Using-Block	Entfernen überflüssiger <i>using</i> -Befehle und/oder Sortieren der <i>using</i> -Befehle

Tabelle 5.11 Refactoring-Methoden in C#

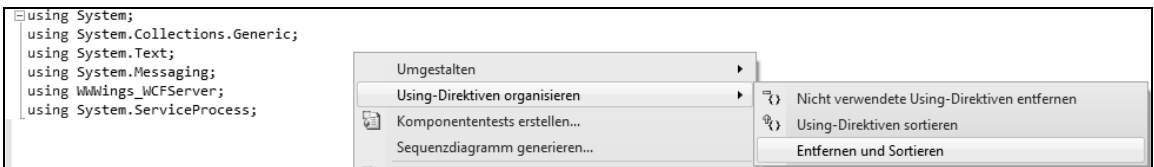


Abbildung 5.29 Organize Usings

HINWEIS Eine »intelligente« *Umbenennen*-Funktion steht auch in Visual Basic zur Verfügung. Wählen Sie dazu im Code-Editor auf der Deklaration des Clients im Kontextmenü den Eintrag *Umbenennen* (*Rename*).

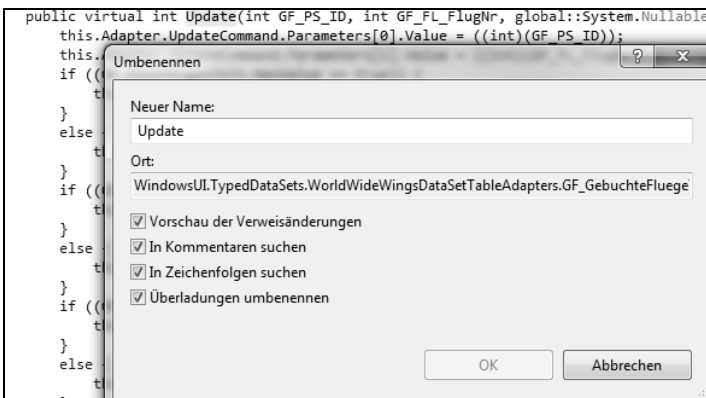


Abbildung 5.30 Umbenennen einer Methode

Refactoring für Visual Basic

Refactoring-Funktionen gibt es in der Grundausstattung weder in Visual Studio 2005 noch in Visual Studio 2008 oder 2010. Aufgrund des Protestes der Visual Basic-Entwickler, die sich durch die fehlenden Refactoring-Funktionen benachteiligt fühlten, hat Microsoft Anfang 2005 einen Vertrag mit der Firma Developer Express Inc. geschlossen, sodass deren Produkt *Refactor! for Visual Basic* allen Visual Studio 2008-Nutzern in einer funktionsreduzierten Version kostenlos zur Verfügung steht [MSDN15].

TIPP Nach der Installation dieser Erweiterung stehen in Visual Basic mehr Refactoring-Funktionen als in C# zur Verfügung.

Codeformatierung

Für alle Programmiersprachen und auch Dokumentenformate (HTML, CSS, XML, XSD, XSLT, etc.) bietet Visual Studio automatisches Einrücken des Codes. Die Art und Tiefe der Einrückung kann über *Extras/Optionen/Text-Editor* für alle Sprachen gleich oder pro Sprache gesteuert werden.

Über *Bearbeiten/Erweitert/Dokument formatieren* und *Bearbeiten/Erweitert/Auswahl formatieren* kann ein Dokument nachträglich eingerückt werden.

Kommentare

Über ein Symbol in der Standardsymbolleiste oder über den Menüpunkt *Bearbeiten/Erweitert* kann ein Codefragment in einen Kommentar umgewandelt bzw. in Code zurückgewandelt werden. Visual Basic bietet zudem im Kontextmenü eines Typs oder eines Typmitglieds den Eintrag *Kommentar einfügen*, um einen Kommentar für die XML-Code-Kommentierung einzufügen.

Suchfunktion

Eine Suchfunktion im Quellcode gab es natürlich auch bisher schon (unter *Edit/Find and Replace*). Dabei konnte man aber immer nur zur nächsten Fundstelle springen. Visual Studio 2010 besitzt ein zusätzliches Suchfenster (*Edit/Navigate To*), das alle Fundstellen zu einem Begriff anzeigt (siehe Bildschirmabbildung). Mit einem Mausklick springt man die Fundstelle an.

TIPP

Auch Camel-Casing-Anfangsbuchstabensuche ist möglich. So findet *KS* zum Beispiel auch *KundenStatus*.

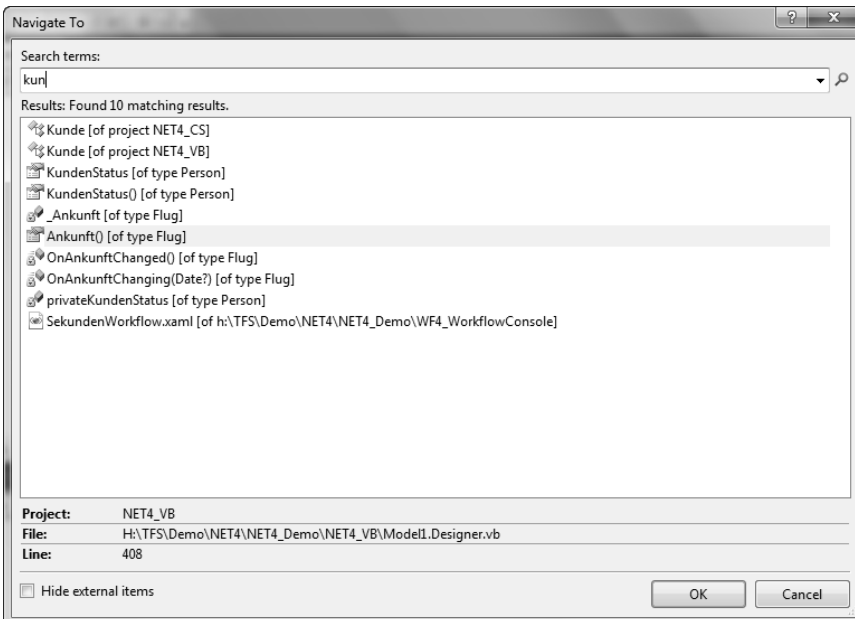


Abbildung 5.31 Navigate To-Fenster in Visual Studio 2010

Aufrufhierarchie und Sequenzdiagramme

Beim Markieren eines Bezeichners hebt Visual Studio 2010 alle weiteren Vorkommen dieses Bezeichners in der aktuellen Datei durch graue Hinterlegung hervor. Außerdem gibt es als Alternative zu der falschen Liste, die bisher *Find all References* produziert, nun auch eine hierarchische Ansicht (*View Call Hierarchy*), in der man sich durch den gesamten Aufrufbaum hangeln kann.

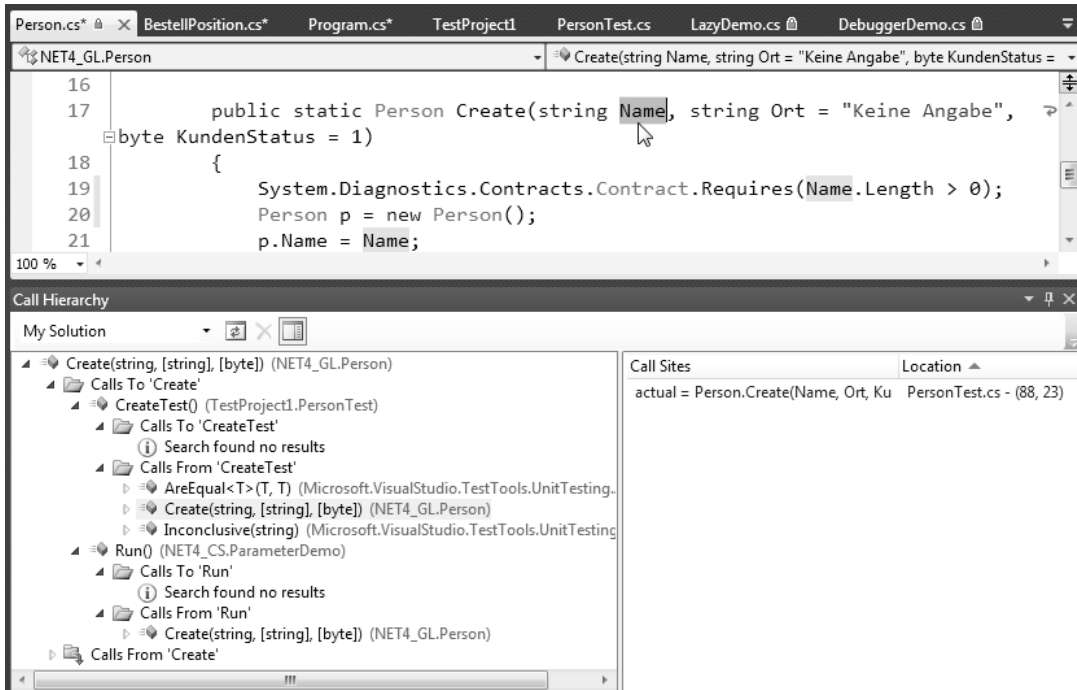


Abbildung 5.32 Code-Hervorhebung und Aufrufhierarchieansicht

Alternativ kann man sich mit der Funktion *Generate Sequence Diagramm* (ab Ultimate-Variante) ein UML-Sequenzdiagramm (Dateityp `.sequencediagram`) anzeigen lassen, das die Funktionsweise einer Routine visualisiert (siehe Abbildung 5.33).

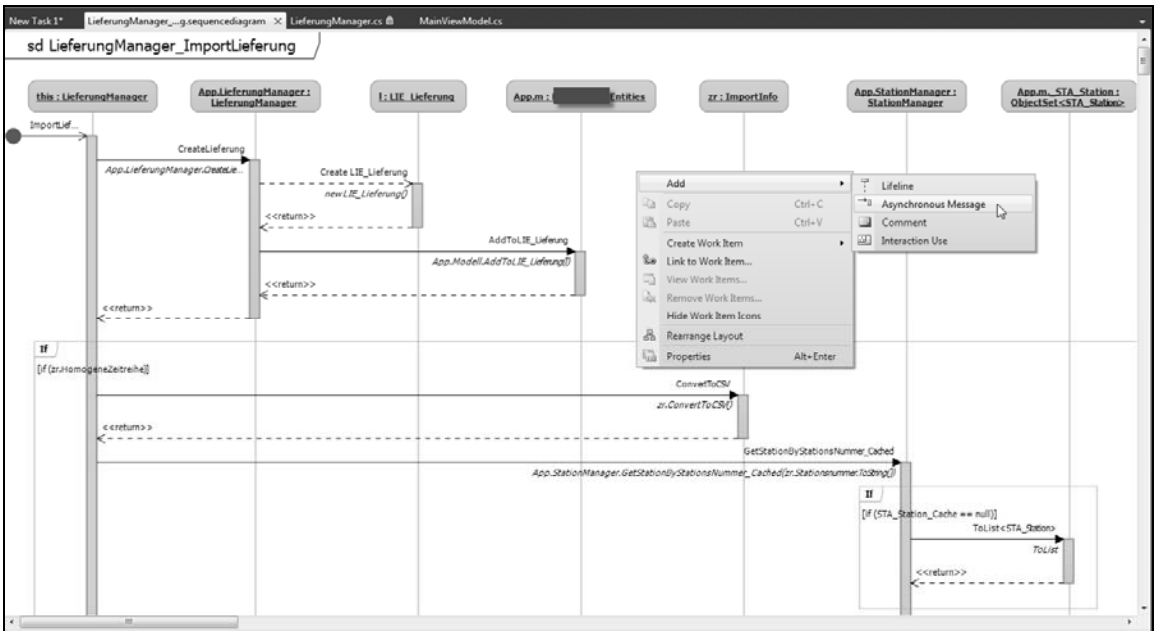


Abbildung 5.33 Generiertes Sequenzdiagramm in Visual Studio 2010

Vertikale Hilfslinien (Editor Guidelines)

Die Möglichkeit zum Setzen vertikaler Linien im Codeeditor gehört zu den versteckten Geheimnissen, die es seit Visual Studio 2002 gibt. Vorher musste man zum Setzen der Hilfslinien einen Eintrag in der Registrierungsdatenbank vornehmen, z.B. `RGB(139,0,0) 4, 11` im Eintrag `Guides` unter `[HKEY_CURRENT_USER]\Software\Microsoft\VisualStudio\8.0\Text Editor` bedeutete, dass eine vertikale rote Hilfslinie bei den Spalten 4 und 11 gezogen werden soll.

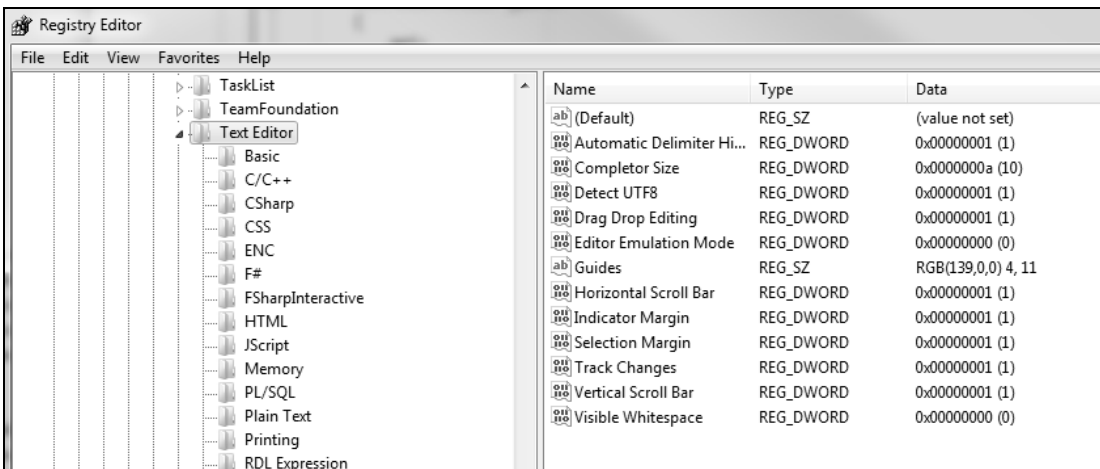


Abbildung 5.34 Aktivieren der vertikalen Hilfslinien in der Windows-Registrierungsdatenbank

Mit den Visual Studio 2010 Productivity Power Tools kann man diesen Eintrag nun komfortabel über das Kontextmenü des Editors setzen.

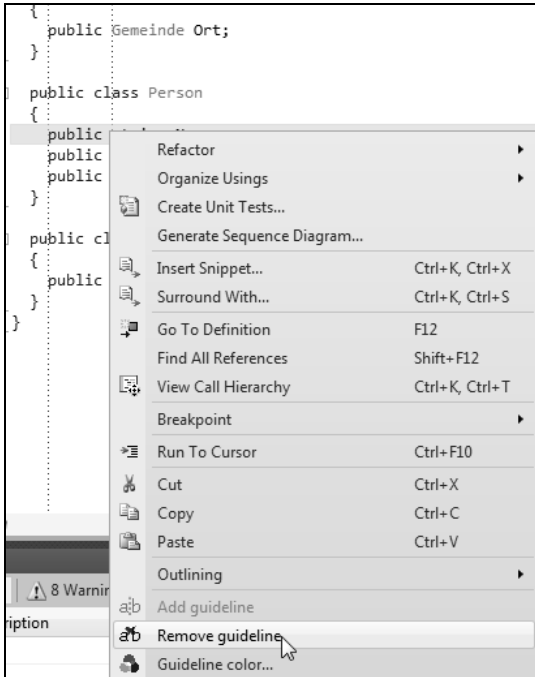


Abbildung 5.35 Verwalten der vertikalen Hilfslinien direkt im Kontextmenü des Codeeditors

Hintergrundkompilierung

Sehr nützlich ist die Hintergrundkompilierung für Visual Basic-Code: Schon während der Eingabe, jeweils nach Verlassen einer Codezeile, prüft die Entwicklungsumgebung auf Syntax- und Typfehler. Eine geschlängelte blaue Linie zeigt dem Entwickler sofort an, dass hier etwas nicht stimmt. Parallel dazu erzeugt Visual Studio einen Eintrag im Fenster *Fehlerliste*. In einigen Fällen werden über Smarttags Fehlerkorrekturvorschläge angeboten (siehe Bildschirmabbildung).

Der Entwickler wird nicht mehr wie bei der automatischen Syntaxprüfung in Visual Studio 6.0 durch Dialogfelder gestört. Während in den ersten Versionen von C# kaum Hintergrundkompilierung existiert, ist dies in den letzten Versionen fast so gut wie in Visual Basic geworden.

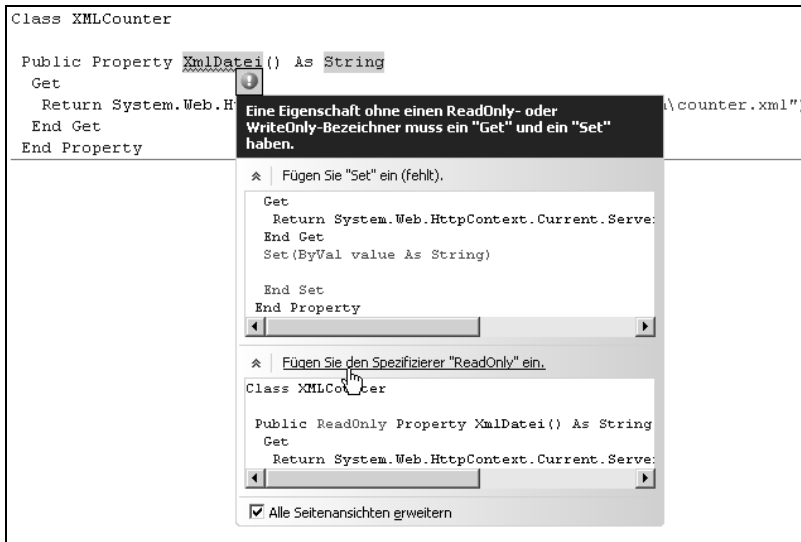


Abbildung 5.36 Fehlerbehebungsvorschläge in Visual Studio

XML-Editor

Visual Studio bietet einen XML-Editor mit IntelliSense-Funktion. Wenn es kein Schema gibt, dann schlägt der Editor nur die Startelemente vor und vervollständigt die schließenden Tags. Wenn ein oder mehrere Schemata in den Eigenschaften des XML-Dokuments hinterlegt sind, bietet Visual Studio die in dem aktuellen Kontext verfügbaren Elemente an. Man kann in dem *Properties*-Fenster unter *Schemas* ein so genanntes *Schema Set* mit mehreren Schemas zusammenstellen.



Abbildung 5.37 Eingabeunterstützung ohne Schema

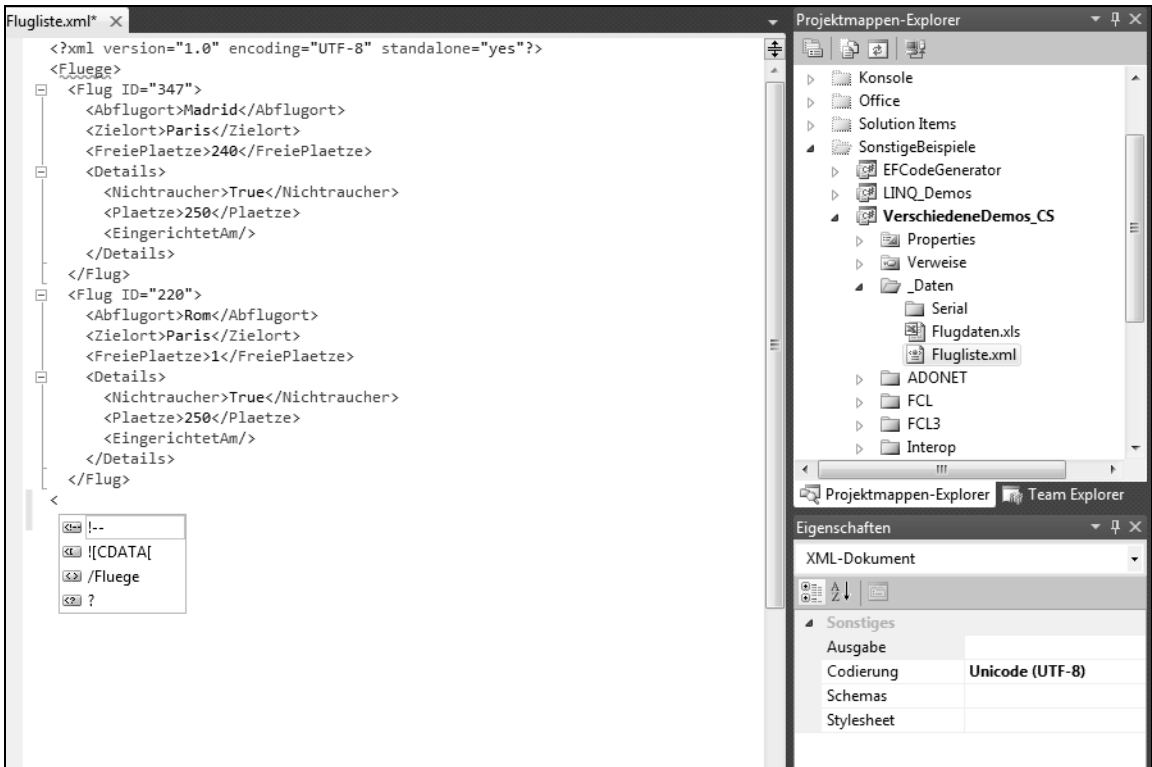


Abbildung 5.38 Eingabeunterstützung mit Schema

Wenn man sich in einem XML-Dokument befindet, steht der zusätzliche Menüpunkt **XML** zur Verfügung. Hier hat man folgende Optionen:

- Festlegen der Schemata (Punkt *Schemas*)
- Erzeugen eines Schemas aus einem XML-Dokument durch Schemaableitung (*Create Schema*)
- Transformation: Ausführung eines XSLT-Dokuments für ein bestimmtes XML-Dokument (*Show XSLT Output*). Diese Option steht auf einem XML-Dokument zur Verfügung, wobei Visual Studio dann nach einer *.xsl*-/ *.xslt*-Datei fragt. Die Funktion steht auch zur Verfügung auf einer *.xsl*-/ *.xslt*-Datei, wobei Visual Studio dann nach einer XML-Datei fragt. Alternativ kann man die Verknüpfung fest hinterlegen in der Eigenschaft *Stylesheet* im *Properties*-Fenster für ein XML-Dokument bzw. *Input* bei einer Stylesheet-Datei.
- Debugging von XSLT-Dateien: siehe Abschnitt »XSLT-Debugging«

Grafische Editoren (Designer)

Visual Studio bietet grafische Editoren für Windows Forms, Windows Presentation Foundation (WPF), Silverlight, Windows Workflow Foundation (WF), ASP.NET-Webforms/HTML-Dokumente, Bitmaps (*.bmp*, *.cur*, *.ico*), Klassendiagramme, XML-Ressourcendateien (*.resx*), XSLT-Dateien (*.xslt*), XML-Schemata (*.xsd*) und Datenmodelle (typisierte DataSets und Entity Framework-Modelle).

HINWEIS Für Silverlight, den kleinen Web-Bruder von WPF, enthält Visual Studio 2010 erstmals überhaupt einen Designer. Bisher musste man alle Silverlight-XAML-Tags händisch erfassen und konnte Visual Studio nur zu einer Voransicht nötigen, aber grafisch keine Änderungen vornehmen. Microsoft verweist die Entwickler immer gerne auf das Produkt Expression Blend, das die Gestaltung mit WPF und Silverlight beherrscht. Neben den zusätzlichen Lizenzkosten schreckt aber auch die an den Bedürfnissen gelernter Designer orientierte Benutzeroberfläche von Blend ab.

Auch der WPF-Designer wurde verbessert.

Designer für Windows Forms, WPF und Webforms

Visual Studio 2010 bietet grafische Designer sowohl für Windows Forms und WPF als auch für ASP.NET-Webforms-Oberflächen.

Vergleich der Designer für Windows Forms, WPF und Webforms

Zwar gleichen sich die Designer für Windows Forms, Windows Presentation Foundation und Webforms oberflächlich, das Ergebnis könnte jedoch verschiedener kaum sein: Der Webforms-Editor erzeugt HTML-Code mit Serversteuerelementen. Der Windows Forms-Editor hingegen setzt jedes Element auf dem Bildschirm in eine Folge von Codezeilen in der für das Projekt gewählten Programmiersprache um. Der WPF-Designer erzeugt XAML-Code.

Zur Übernahme von Benutzeroberflächen zwischen Web und Windows besteht folglich (ohne Zusatzwerkzeuge) keine Chance. Die Möglichkeiten zur Trennung von Code und Gestaltung in verschiedenen Dateien haben sich schon seit Visual Studio Version 2005 gegenüber der Vorgängerversion verändert: Während unter Visual Studio .NET 2002/2003 bei Webforms immer getrennt wurde und bei Windows Forms nicht, besteht nun bei Webforms die Wahl und bei Windows Forms erfolgt immer eine Trennung. Auch WPF trennt Code im Layout immer. Das in der Tabelle genannte *Ein-Datei-Modell* für WPF kann man nur manuell erreichen.

	Gestaltung	Eigener Code (Ereignisbehandlungsroutinen)
Windows Forms	Formularname.Designer.cs (.vb, .cs, .jsl etc.)	Formularname.cs (.vb, .cs, .jsl etc.)
WPF – Ein-Datei-Modell (Inline Code)	Fenstername.xaml	Code in .xaml-Datei
WPF – Hintergrundcode-Modell	Fenstername.xaml	Fenstername.xaml.xs
Webforms – Ein-Datei-Modell (Inline Code)	HTML + Serversteuerelemente in .aspx-Datei	Code in .aspx-Datei
Webforms – Hintergrundcode-Modell	HTML + Serversteuerelemente in .aspx-Datei	.aspx.cs, .aspx.vb

Tabelle 5.12 Trennung von Code und Gestaltung in Windows Forms, WPF/Silverlight und ASP.NET Webforms

Eine zweigeteilte Ansicht, in der man Quellcode und Designer gleichzeitig sieht (*Split View*), bietet Visual Studio für WPF/Silverlight und Webforms an. Bei beiden Designern kann man zwischen einer horizontalen oder einer vertikalen Teilung wählen. Nur beim WPF-Designer lässt sich frei wählen, was in welcher Hälfte erscheinen soll. Während man beim WPF-Designer die Einstellungen durch Symbole im Designer vornehmen kann, muss man die Ansicht zwischen horizontal und vertikal bei Webforms unter *Extras/Optionen/HTML Designer* umschalten.

HINWEIS Details zum Einsatz der Designer erfahren Sie in den Kapiteln zu Windows-Oberflächen mit Windows Forms, Windows Presentation Foundation (WPF) und ASP.NET (dieses Zusatzkapitel können Sie als PDF auf dem Leser-Portal herunterladen).

Werkzeuggeste

Die Werkzeuggeste (Toolbox) bietet verschiedene Registerkarten, jeweils angepasst auf den aktuell geöffneten Designer. Leider kann man in Visual Studio 2010 immer noch nicht in der Werkzeuggeste suchen.

TIPP Um den Inhalt der Werkzeuggeste zu erweitern, wählen Sie *Choose Items* im Kontextmenü der Werkzeuggeste. In dem Dialogfeld werden dann alle Steuerelemente angeboten, die sich im Global Assembly Cache (GAC) sowie allen in der Registrierungsdatenbank unter `HKEY_CURRENT_USER\SOFTWARE\Microsoft\NETFramework\AssemblyFolders` verzeichneten Dateisystemordner befinden. Alternativ kann man in dem Dialogfeld über *Browse* auch direkt eine Assembly-Datei im Dateisystem auswählen.

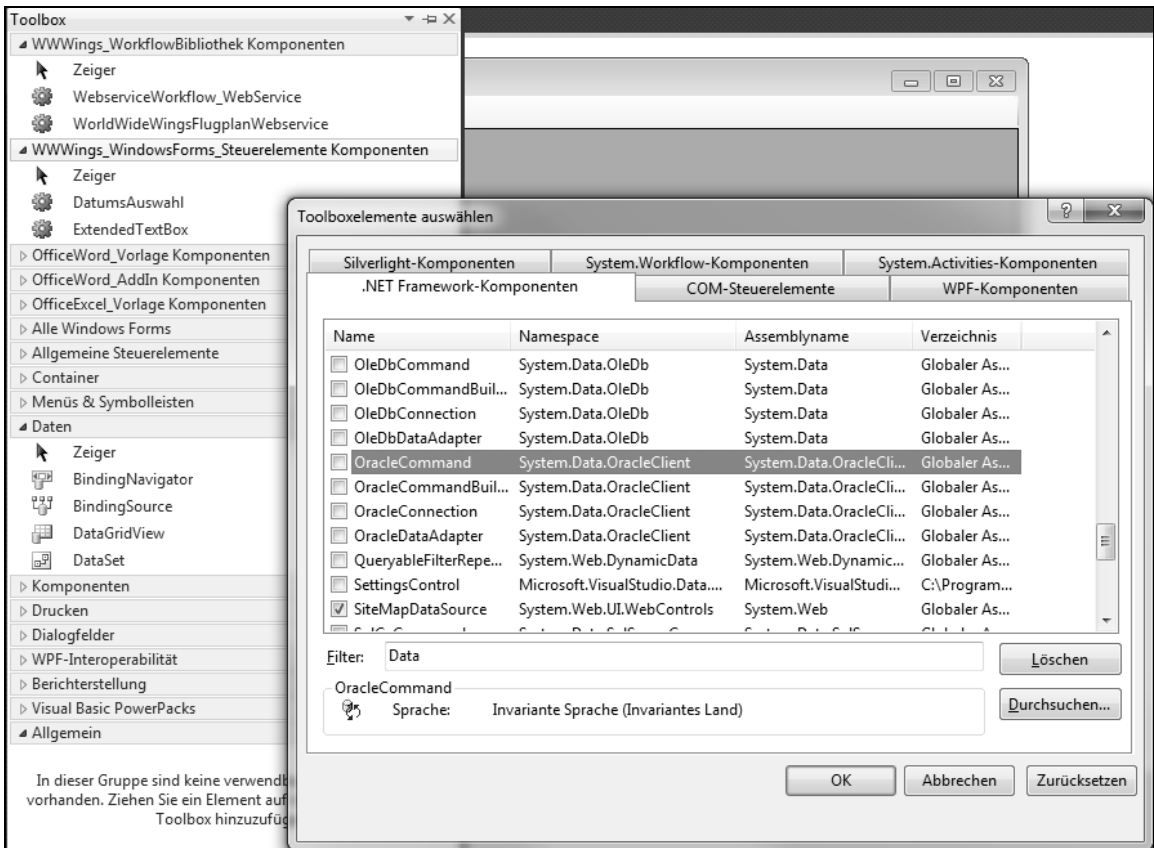


Abbildung 5.39 Konfiguration der Werkzeuggeste in Visual Studio 2010

Designer für nichtvisuelle Komponenten

Auch für nichtvisuelle .NET-Komponenten – Elementvorlage *Komponentenklasse* (*Component Class*) – gibt es einen Designer, der als ein Seiten füllender *Component Tray* für die Aufnahme von Entwurfszeit-Steuer-elementen verstanden werden kann.

Eine Komponentenklasse besteht wie ein Formular aus drei Dateien: aus einer Codedatei für den Entwickler (*component.cs/.vb/.jsl*), einer Codedatei für den vom Designer generierten Code (*component.designer.cs/.vb/.jsl*) sowie einer Ressourcendatei (*.resx*). Die erzeugte Klasse ist abgeleitet von `System.ComponentModel.Component`.

Der Komponentenklassendesigner erlaubt auch, visuelle Windows Forms-Steuerelemente einzufügen. Dies ist in der Regel jedoch nicht sinnvoll.

HINWEIS Bei der ersten Ankündigung von .NET im Juli 2000 sprach Microsoft einmal von der *Orchestrierung* von Anwendungen mit Visual Studio. Viele Beobachter hatten erwartet, dass man in Visual Studio den Verbindungscode (Glue Code) für Softwarekomponenten grafisch erzeugen könnte. Davon ist aber leider auch in Visual Studio 2010 nichts zu sehen. Am ehesten findet man diese Gedanken in der Windows Workflow Foundation und in Microsoft BizTalk. Mit dem derzeitigen Versionsumfang macht der Komponentenklassendesigner wenig Sinn. Einen Schritt in Richtung *Orchestrierung* geht Microsoft erst mit der Windows Workflow Foundation (WF). Vorher gab es eine Art *Orchestrierung* nur im Microsoft BizTalk Server.

XSD-Designer

In Visual Studio 2010 startet Microsoft auch einen neuen Versuch für eine grafische Bearbeitung von XML-Schema-Dateien (XSD). Bereits in früheren Visual Studio-Versuchen gab es da Werkzeuge, die aber wieder herausgenommen wurden.

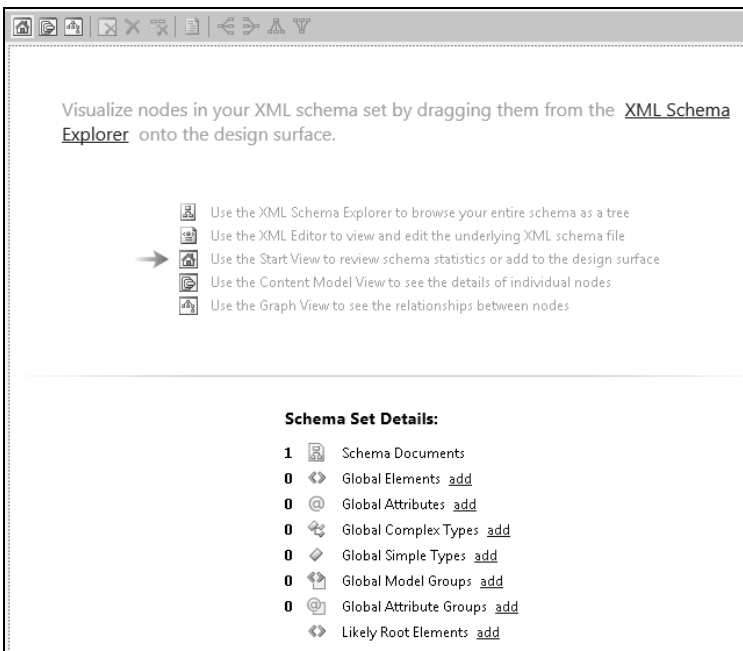


Abbildung 5.40 Die Startseite des XSD-Designers gibt Informationen über die Möglichkeiten

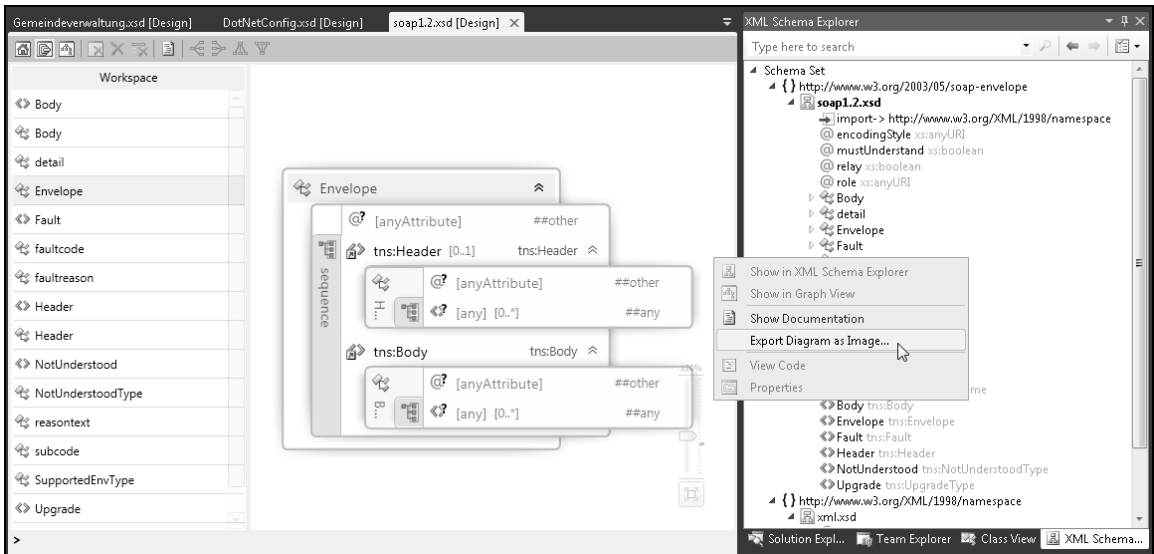


Abbildung 5.41 Ansicht *Content Model* im XSD-Designer (am Beispiel des XML-Schema für SOAP 1.2)

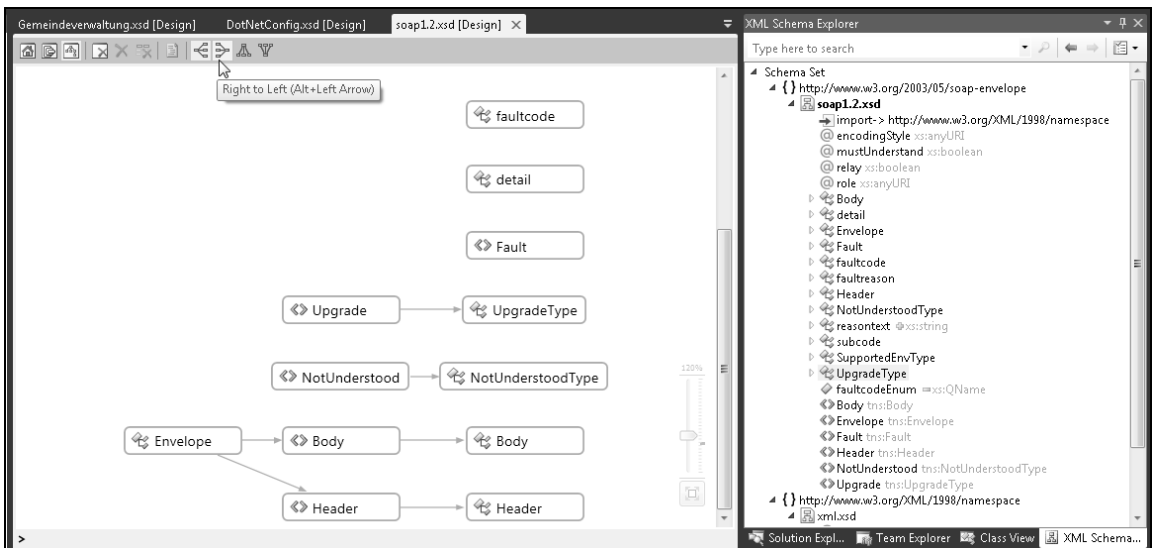


Abbildung 5.42 Ansicht *Graph* im XSD-Designer (am Beispiel des XML-Schema für SOAP 1.2)

Seit Visual Studio 2008 Service Pack 1 ist mit dem Projektelement *XML to Schema* eine weitere Möglichkeit geschaffen worden, XSD-Dateien erzeugen zu lassen. Zur Wahl steht die Erzeugung aus einer lokalen XML-Datei, einer Webressource oder aus dem Inhalt der Zwischenablage.

HINWEIS Bisher steht *XML to Schema* kurioserweise nur für Visual Basic-Projekte zur Verfügung.

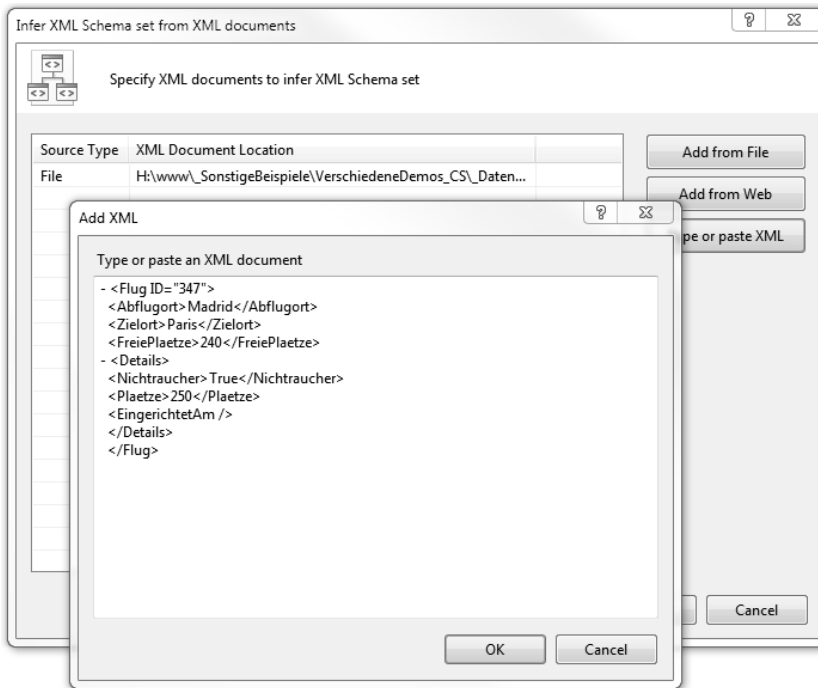


Abbildung 5.43 XSD-Erzeugung aus XML-Dateien mit dem Projektelement XML to Schema

Designer für Workflows

Ein weiterer grafischer Designer erstellt Workflows für die Windows Workflow Foundation (WF). Die bisherige Implementierung der Laufzeitumgebung und des Designers war aber nicht zufriedenstellend, sodass Microsoft in .NET 4.0/Visual Studio 2010 alles neu implementiert hat. Der neue Designer arbeitet zwar tatsächlich etwas flüssiger, die Bedienung ist aber immer noch umständlich. Details erfahren Sie im Kapitel zu »Windows Workflow Foundation 4.0«.

Weitere Fenster

Dieser Abschnitt bespricht einige weitere nützliche Funktionen von Visual Studio 2010.

Aufgabenliste

Die *Aufgabenliste* (Menü *Ansicht/Aufgabenliste*) ermöglicht die automatische oder manuelle Aufnahme von Stichpunkten. Die manuelle Aufnahme erfolgt einfach über das Fenster selbst (Symbol *Benutzeraufgabe erstellen*). Automatisch aufgenommen werden Kommentare aus dem Quellcode, die bestimmte Schlüsselwörter enthalten. Vordefiniert sind die Schlüsselwörter *TODO*, *UNDONE*, *HACK* und *UnresolvedMergeConflict*. Einen Kommentar, der mit diesen Schlüsselwörtern beginnt, fügt Visual Studio automatisch der Aufgabenliste hinzu. Weitere Schlüsselwörter kann man über *Extras/Optionen/Umgebung/Aufgabenliste* definieren.

HINWEIS Kompilierungsfehler und Warnungen werden seit Visual Studio 2005 nicht mehr in der Aufgabenliste, sondern in dem getrennten Fenster *Fehlerliste* (*Error List*) angezeigt.

Suche über Visual Studio-Funktionen

Über den Quellcode konnte man immer schon in Visual Studio suchen. Visual Studio selbst wird aber immer mächtiger und damit unübersichtlicher. Die Productivity Power Tools bringen aber eine weitere hilfreiche Suche mit: Eine Suche über Visual Studio-Funktionen (Menüpunkte, Kontextmenüeinträge, Vorlagen, Optionen).

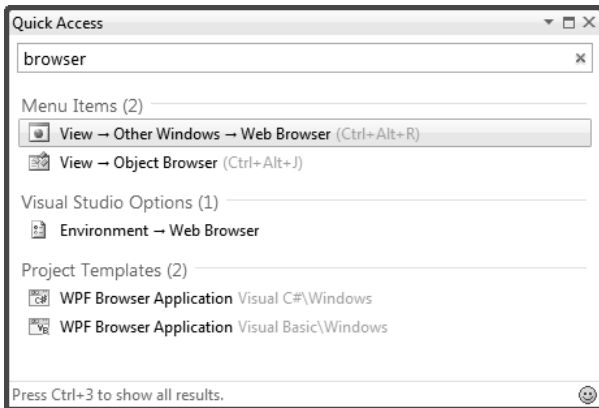


Abbildung 5.44 Suche nach Visual Studio-Funktionen, die *Browser* im Namen tragen

Codedefinitionsfenster (Code Definition Window)

Das Fenster *Codedefinition* zeigt zu jedem Typ bzw. jedem Typmitglied die Definition an. Typen bzw. Typmitglieder aus Projekten, die mit zur Projektmappe gehören, werden im kompletten Quelltext angezeigt. Für Typen bzw. Typmitglieder aus kompilierten Assemblys werden nur Rümpfe und Kommentare angezeigt.

Server Explorer

Der *Server Explorer* zeigt neben dem bereits besprochenen Ast *Datenverbindungen* (*Data Connections*) auch Systemkomponenten (z. B. Dienste, Leistungsindikatoren, -Klassen der Windows Management Instrumentation (WMI), Ereignisprotokolle) eines beliebigen Rechners im LAN an und bietet die grafische Instanziierung per Drag & Drop bzw. Codegenerierung im Kontextmenü. Das bisherige WMI-Add-in für den Server Explorer ist nun fester Bestandteil.

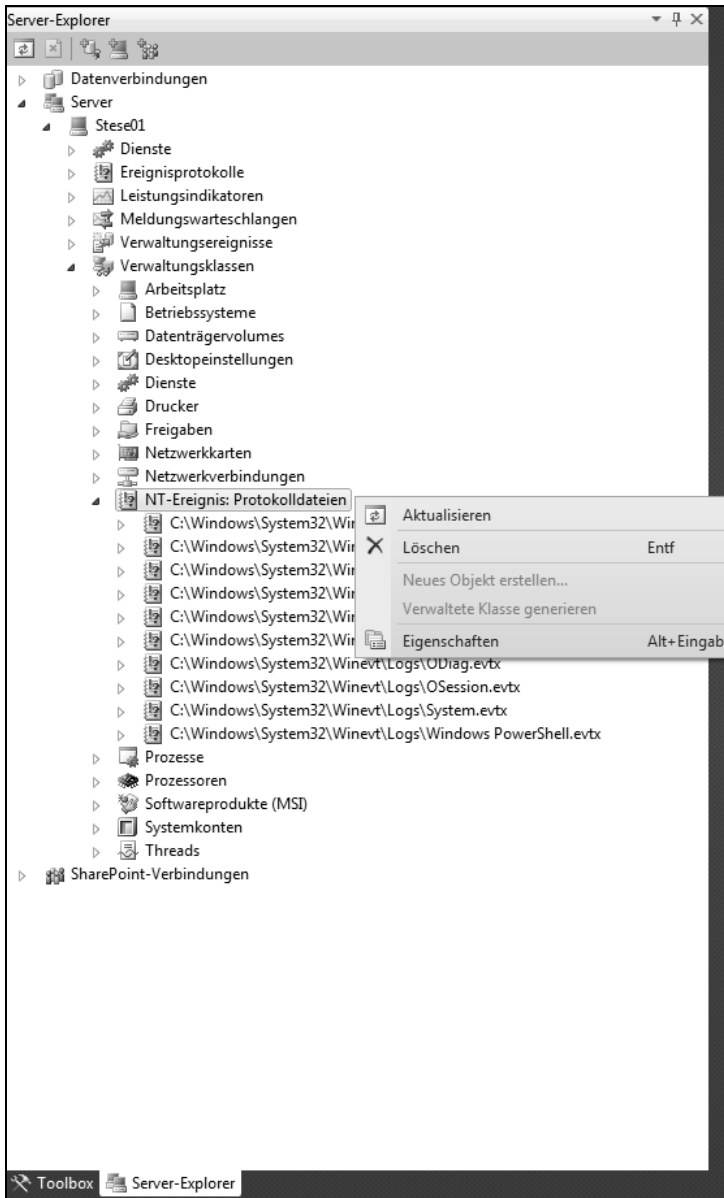


Abbildung 5.45 Generierung einer .NET-Wrapper-Klasse für eine WMI-Klasse im Server Explorer

Objektbrowser und Klassenansicht (Object Browser / Class View)

Das Fenster *Objektbrowser* (engl. *Object Browser*) ermöglicht es, die Typen und Typmitglieder einer Assembly anzuzeigen. Nach Namensräumen sortiert werden neben den Typen in dem geöffneten Projekt auch alle referenzierten Komponenten sowie die Komponenten der .NET-Klassenbibliothek. Die Anzeige kann auf die verwendeten Komponenten begrenzt werden. Alternativ kann die Anzeige nach Komponenten gruppiert werden (Symbol *Object Browser Settings*).

Das Fenster *Klassenansicht* (engl. *Class View*) bietet eine ähnliche Sicht – allerdings nur für das geöffnete Projekt und mit mehr Funktionen. Die Typen sind nach Namensräumen angeordnet, unabhängig davon, in welcher Datei die Implementierung liegt. Zentrale Funktion ist die Möglichkeit, die Implementierung eines Typs oder eines Mitglieds durch einen Doppelklick auf den Namen direkt anzuspringen. Das Kontextmenü bietet weitere hilfreiche Funktionen:

- *Definition durchsuchen (Browse Definition)* springt direkt zur entsprechenden Stelle im Objektbrowser

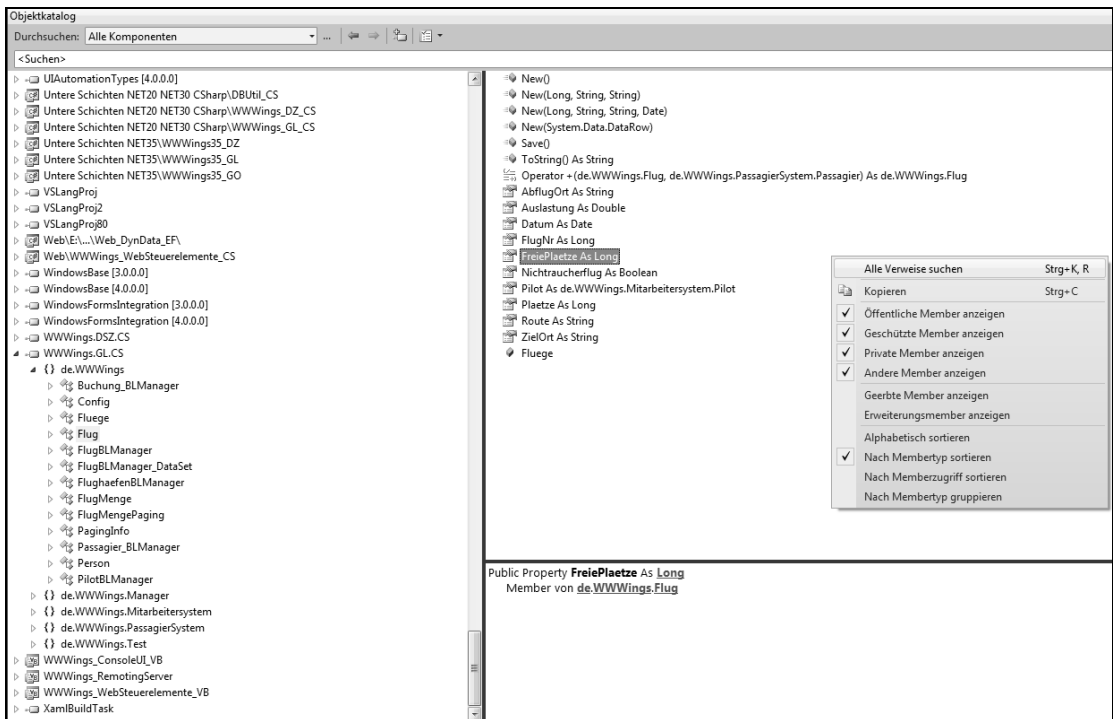


Abbildung 5.46 Objektbrowser in Visual Studio

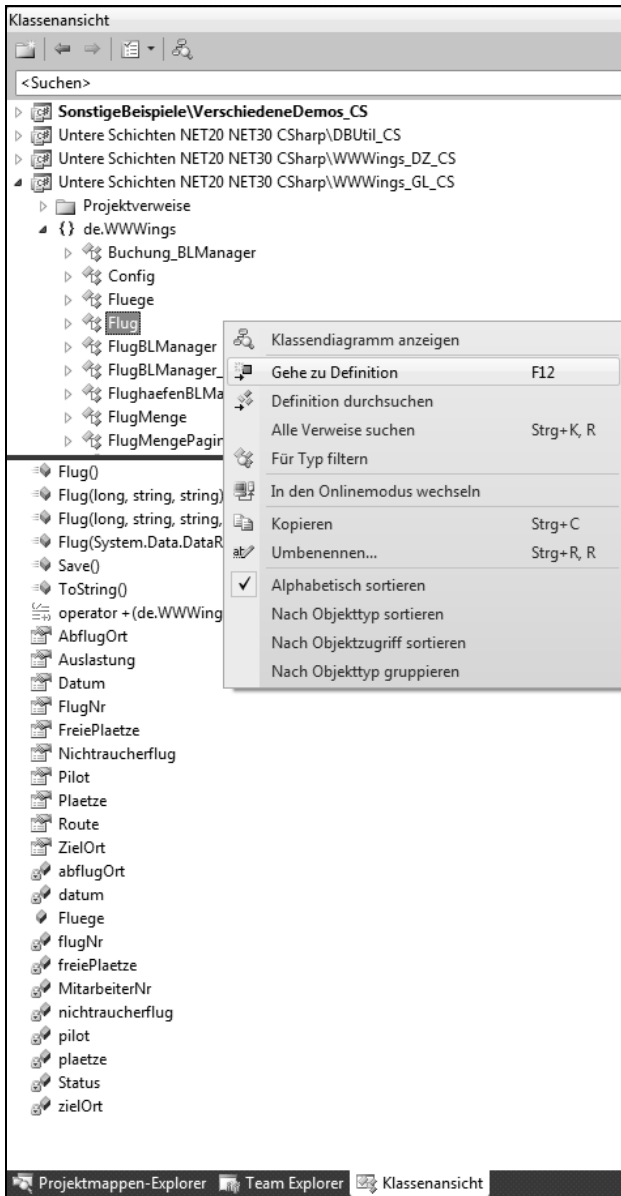


Abbildung 5.47 Klassenansicht in Visual Studio

- *Alle Verweise suchen (Find all References)* zeigt eine Liste aller Codestellen, die auf einen Typ bzw. ein Mitglied verweisen
- *Umbenennen (Rename)* bietet die Möglichkeit, einen Typ umzubenennen und alle nutzenden Codestellen dabei anzupassen

Modellierung

Microsoft war bisher nie stark im Bereich der Modellierung. Frühere Versionen von Visual Studio konnten im Zusammenspiel mit Visio aus UML-Klassendiagrammen Programmcode generieren und Programmcode in Klassendiagrammen visualisieren, beides aber als Einbahnstraße. Ein Round-Trip-Engineering war nicht möglich. Im Jahr 2004 hatte Microsoft dann angekündigt, die eigene UML-Unterstützung bei Version 1.3 einzufrieren und stattdessen domänenspezifische Modellierungssprachen zu entwickeln. Allerdings war in Visual Studio Team System (VSTS) 2005 und 2008 davon kaum etwas zu sehen und mit Ausnahme des (nicht-UML-konformen) Klassendiagrammdesigners gab es auch kaum Anwender. Die Modellierungswerkzeuge waren der bei weitem schwächste Teil von Visual Studio Team System. In dem VSTS-Nachfolger "Visual Studio 2010 Ultimate" ist dies besser geworden.

HINWEIS Für Klassendiagramme gibt es in Visual Studio 2010 insgesamt drei Designer, die alle leider anders funktionieren.

- Designer für Klassendiagramme (seit Visual Studio 2005) – siehe unten
 - Entity Framework-Designer (seit Visual Studio 2008 SP1) – siehe Kapitel 12 zum Entity Framework
 - UML-Klassendiagramm-Designer (seit Visual Studio 2010) – siehe unten
-

Designer für Klassendiagramme (seit Visual Studio 2005)

Während in Visual Studio .NET 2002/2003 die einzige Möglichkeit zur visuellen Gestaltung von Klassendiagrammen in der Nutzung von Microsoft Visio-UML-Diagrammen bestand, bietet Visual Studio seit Version 2005 einen eigenen Klassendiagrammdesigner. Die Visual Studio-Klassendiagramme sind zwar UML-ähnlich, unterstützen aber keineswegs den vollen UML-Standard.

Die Klassendiagramme bieten eine visuelle Repräsentation von C#-, Visual Basic- oder J#-Programmcode. Dabei kann der Entwickler sowohl bestehenden Programmcode visualisieren als auch Programmcode durch visuelle Modellierung erzeugen.

Der Klassendesigner bietet echtes Round-Trip-Engineering, d. h., jede Änderung im Diagramm wird in den Code übernommen und jede Änderung im Code in das Diagramm. Möglich wird dies, indem der Code der einzige Speicher für die Klasseninformationen ist. Round-Trip-Engineering ist daher möglich ohne die Gefahr von Änderungskonflikten. In der .cd-Datei, die Visual Studio für jedes Klassendiagramm anlegt, werden lediglich das Diagrammlayout (Position und Größe der Kästchen) sowie Kommentare abgelegt.

Der Klassendiagrammdesigner bietet eine Werkzeugleiste mit den Elementen *Klasse (Class)*, *Enumeration (Enum)*, *Schnittstelle (Interface)*, *Abstrakte Klasse (Abstract Class)*, *Struktur (Struct)*, *Delegat (Delegate)*, *Vererbung (Inheritance)*, *Zuordnung (Association)* und *Kommentar (Comment)*. Beim Fallenlassen der sechs Erstgenannten fragt Visual Studio nach dem gewünschten Speicherort des Typs. Alternativ kann ein bestehender Typ aus der *Klassenansicht (Class View)* oder eine Datei aus dem *Projektmappen-Explorer (Solution Explorer)* per Drag & Drop in das Diagramm eingebaut werden. Wenn in einer Datei mehrere Typen realisiert sind, fügt der Designer alle enthaltenen Typen ein. In dem Klassendiagramm können direkt Klassenmitglieder ergänzt und verändert werden. Außerdem können einige Refactoring-Funktionen unmittelbar aus dem Diagramm heraus aufgerufen werden.

Der Klassendiagrammdesigner unterstützt alle .NET-Konstrukte einschließlich Generics und partielle Klassen. Der Klassendesigner ist für C++ erst ab Version 2008 und nur eingeschränkt verfügbar.

TIPP Die Mitglieder eines Typs können im Diagramm selbst (*Erweitern/Reduzieren* auf einem Typ-Rechteck) oder in einem separaten Fenster (*Ansicht/Weitere Fenster/Klassendetails*) angezeigt werden.

Ein Beispiel für ein Visual Studio-Klassendiagramm haben Sie bereits im Kapitel 2 »World Wide Wings – Das mehrschichtige Fallbeispiel in diesem Buch« gesehen. Zu Gunsten anderer Informationen wird hier auf den erneuten Abdruck verzichtet.

UML-Diagramme in Visual Studio 2010

In Visual Studio 2010 beugt sich Microsoft nun dem Kundenwunsch und bietet die UML-Diagrammtypen *Activity*, *Use Case*, *Logical Class*, *Component* und *Sequence* an. Zudem gibt es einen nicht UML-konformen Diagrammtyp *Layer*. Zur Nutzung der Diagramme muss man zunächst ein Projekt vom Typ *Modeling Project* anlegen.

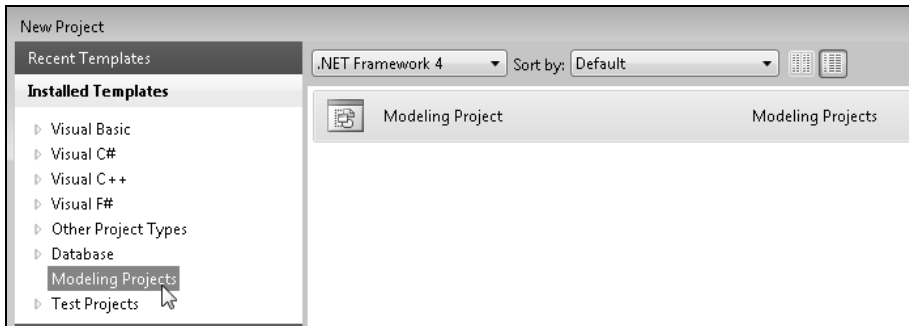


Abbildung 5.48 Anlegen eines Modeling Projects

Dann hat man die in der folgenden Bildschirmabbildung gezeigten Diagramme zur Wahl.

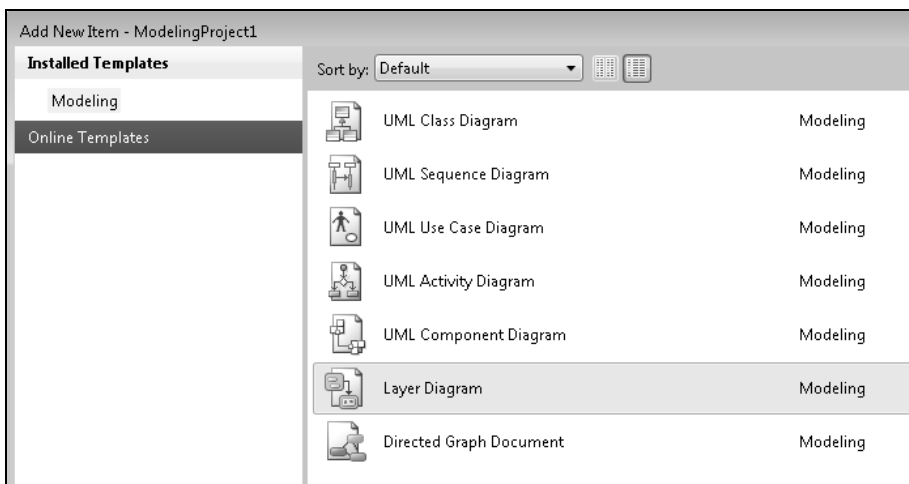


Abbildung 5.49 Diagrammarten in Visual Studio 2010

In der RTM-Version wurden allerdings UML und die anderen Diagrammtypen nur zum Malen mitgeliefert. Codegenerierungen aus UML-Klassendiagrammen sowie die Erzeugung von Klassendiagrammen aus Programmcode bekommt man mit dem *Visual Studio 2010 Visualization and Modeling Feature Pack*, das im Juni 2010 als kostenfreie Erweiterung erschienen ist.

Das *Visual Studio 2010 Visualization and Modeling Feature Pack* enthält folgende Funktionen für UML-Diagramme:

- Programmcodegenerierung aus UML-Klassendiagrammen
- Erzeugung von UML-Klassendiagrammen aus Programmcode
- Import von XMI 2.1-Dateien (für Klassendiagramme, Sequenzdiagramme und Use Case-Diagramme)
- Verknüpfung von TFS Work Items mit Modellen
- Erzeugen von Layer-Diagrammen aus bestehendem C- und C++-Code
- Erzeugen von Abhängigkeitsgraphen für C/C++-Projekte und ASP.NET-Webprojekte

Zum Erzeugen eines UML-Klassendiagramms aus bestehendem Programmcode muss man den Architecture Explorer (Menu View) aufrufen und dann die Klassen auf die Diagrammoberfläche ziehen. Die nachstehende Bildschirmabbildung zeigt ein UML-Klassendiagramm, das vier Klassen enthält. Wie man sieht, wurde in dem Diagramm zwar die Vererbungsbeziehung zwischen *Besitzer* und *Person* angelegt, nicht aber die vielfältigen Assoziationen zwischen den Klassen *Haus*, *Gemeinde* und *Besitzer*.

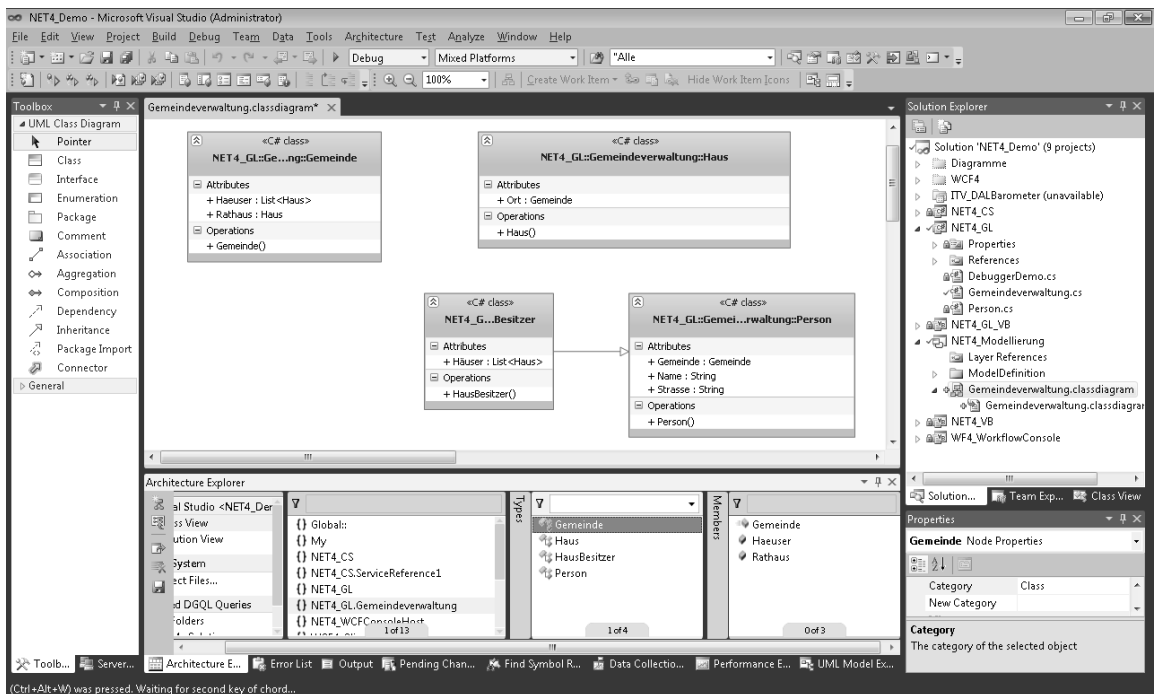


Abbildung 5.50 UML-Modellierung in Visual Studio 2010 Ultimate (hier: Klassendiagramm)

Oslo, »M« und Quadrant

Einige Jahre lang hat Microsoft an einer eigenen Modellierungsstrategie im Rahmen einer Gesamtstrategie für Service-orientierte Architekturen (SOA) unter dem Codenamen *Oslo* entwickelt. Codename für die Modellierungssprache war »M« und dazu gab es ein passendes Werkzeug *Quadrant*. Das Oslo-Projekt ist zum Teil in Produkte gemündet (z.B. Windows Application Server *AppFabric* und WCF 4.0).

Im September 2010 hat Microsoft bekannt gegeben, einen Teil des Projekts – insbesondere das geplante Modell-Repository (SQL Server Data Modeling) und das Werkzeug Quadrant – zu beerdigen [DBOX01]. Lediglich »M« wird es in einer veränderten Form vielleicht irgendwann geben.

Architektur- und Codeanalyse

Codeanalyse-Funktionen gibt es in Visual Studio seit der Version 2005. Sie wurden in Visual Studio 2010 erweitert durch grafische Darstellungen.

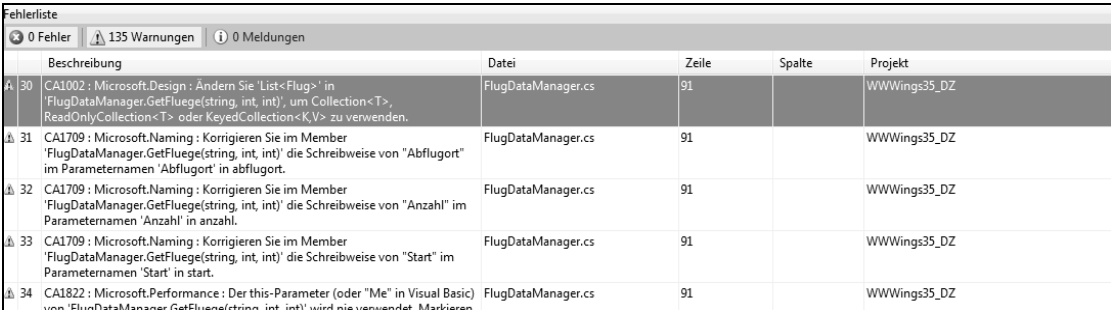
Statische Codeanalyse

Im Rahmen der statischen Codeanalyse erfolgt eine Überprüfung des Quellcodes in Hinblick auf die Einhaltung von Codierungsrichtlinien sowie die Codekomplexität.

Microsoft hat bereits für das .NET Framework 1.0 Richtlinien für die Benennung und Verwendung von Typen und Mitgliedern erlassen [MSDN13] und dazu passend ein Prüfungswerkzeug (FxCop) angeboten. FxCop gibt es weiterhin als kostenlose eigenständige Windows-Anwendung [MSDN37].

Wenn Sie eine Visual Studio-Version besitzen, die die statische Codeanalyse unterstützt, ist die Funktionalität von FxCop direkt in die Entwicklungsumgebung integriert:

- Im Menü *Erstellen* bzw. im Kontextmenü eines Projekts findet man den Befehl *Codeanalyse ausführen*. Die Codeanalysemeldungen erscheinen in der Fehlerliste.
- In den Projekteigenschaften auf der Registerkarte *Codeanalyse* kann festgelegt werden, welche Regeln zur Anwendung kommen sollen und ob Regelverstöße zu Warnungen oder Fehlermeldungen führen.



Fehlerliste					
0 Fehler 135 Warnungen 0 Meldungen					
	Beschreibung	Datei	Zeile	Spalte	Projekt
30	CA1002 : Microsoft.Design : Ändern Sie 'List<Flugs>' in 'FlugDataManager.GetFluege(string, int, int)', um Collection<T>, ReadOnlyCollection<T> oder KeyedCollection<K,V> zu verwenden.	FlugDataManager.cs	91		WWWings35_DZ
31	CA1709 : Microsoft.Naming : Korrigieren Sie im Member 'FlugDataManager.GetFluege(string, int, int)' die Schreibweise von "Abflugort" im Parameternamen 'Abflugort' in abflugort.	FlugDataManager.cs	91		WWWings35_DZ
32	CA1709 : Microsoft.Naming : Korrigieren Sie im Member 'FlugDataManager.GetFluege(string, int, int)' die Schreibweise von "Anzahl" im Parameternamen 'Anzahl' in anzahl.	FlugDataManager.cs	91		WWWings35_DZ
33	CA1709 : Microsoft.Naming : Korrigieren Sie im Member 'FlugDataManager.GetFluege(string, int, int)' die Schreibweise von "Start" im Parameternamen 'Start' in start.	FlugDataManager.cs	91		WWWings35_DZ
34	CA1822 : Microsoft.Performance : Der this-Parameter (oder "Me" in Visual Basic) von 'FlugDataManager.GetFluege(string, int, int)' wird nie verwendet. Markieren	FlugDataManager.cs	91		WWWings35_DZ

Abbildung 5.51 Liste der Warnungen nach einer Codeanalyse

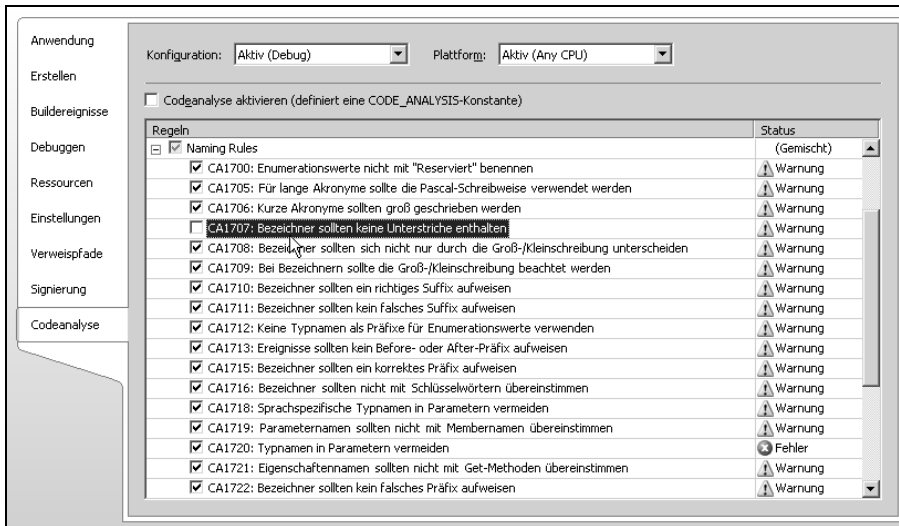


Abbildung 5.52 Konfiguration der Codeanalyse

HINWEIS Ein wichtiger Unterschied zwischen FxCop und der in Visual Studio integrierten Codeanalysefunktion besteht darin, dass FxCop nur die kompilierte Assembly analysiert, während die Codeanalysefunktion von Visual Studio den Quellcode kennt und daher ein direktes Anspringen der Quellcodestellen aus dem Fenster *Fehlerliste* ermöglicht.

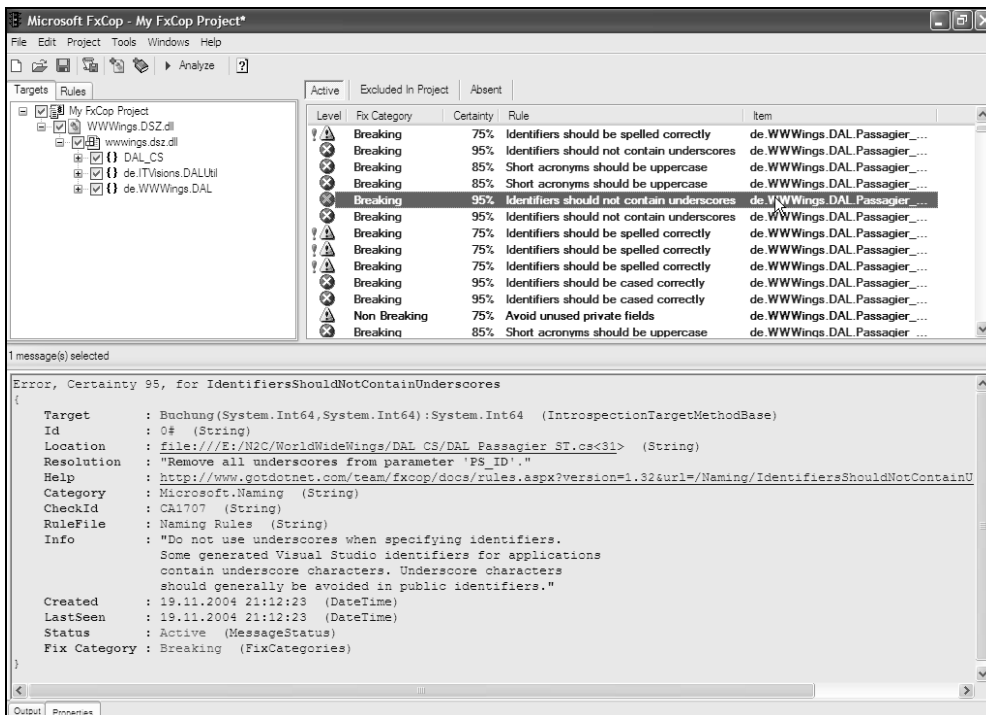


Abbildung 5.53 FxCop für .NET

Code-Kennzahlen

Neu seit Visual Studio 2008 ist die Code-Kennzahlen-Funktion (Code Metrics): Visual Studio liefert auf Wunsch für jede Komponente, Klasse oder Methode Daten über die Anzahl der Codezeilen, die Anzahl der Abhängigkeiten zwischen Klassen und die Vererbungstiefe sowie einen *Maintainability Index* von 0 bis 100, wobei die Entwicklungsumgebung Werte zwischen 0 und 9 als sehr schlecht, 10-19 als bedenklich und Werte über 20 als unbedenklich ansieht (siehe [FXCOP01]).

Nach dem Aufruf der Analyse über das Menü *Analyse/Calculate Code Metrics* kann man die Ergebnisse pro Assembly, Namensraum, Klasse und Methode betrachten.

Hierarchie	Wartbarkeitsindex	Zyklomatische Komplexität	Vererbungstiefe	Klassenkopplung	Codezeilen
Untere Schichten NET20 NET30 CSharp\DBUtil_CS (Debug)	81	48	1	31	155
Untere Schichten NET20 NET30 CSharp\WWWings_DZ_CS (Debug)	79	107	1	37	263
Untere Schichten NET20 NET30 CSharp\WWWings_GL_CS (Debug)	90	277	4	83	384
{ } de.WWWings	89	136	2	56	190
Buchung_BLManger	88	11	1	6	20
Config	95	3	1	1	3
Fluege	89	11	2	14	13
Flug	88	31	2	19	43
FlugBLManger	80	29	2	24	45
FlugBLManger_DataSet	93	5	1	2	5
FlughaefenBLManger	75	5	1	9	9
FlugMenge	100	1	2	2	1
FlugMengePaging	84	9	1	10	16
IPersonenKlasse	100	7	0	0	0
ObjektStatus	100	0	1	0	0
PagingInfo	71	1	2	1	6
Passagier_BLManger	89	8	1	7	12
AnzahlPassagiere() : int	95	1		1	1
HoleAlle() : PassagierMenge	84	1		4	2
HolePassagier(long) : Passagier	78	1		4	3
HolePassagierNummer(string) : long	76	1		1	3
HolePassagierNummer(string, string) : long	92	1		1	1
[NeuerPassagier(long, string, string, string, byte, DateTime) : void	1	1		2	1
NeuerPassagier(string, string) : void	1	1		0	0
Passagier_BLManger()	1	1		0	1
Person	13		1	3	15
PilotBLManger	2	1	1	1	2
de.WWWings.Fluggesellschaft	6	0	0	2	0
de.WWWings.Manager	4	0	0	0	0
de.WWWings.Mitarbeitersystem	95	4	1	3	4
de.WWWings.MitarbeiterSystem	96	25	3	10	26
de.WWWings.ORM	73	25	1	18	49
de.WWWings.PassagierSystem	93	66	4	35	94
de.WWWings-Test	81	11	1	7	21

Abbildung 5.54 Auswertung der Code-Analyse

Architecture Explorer

Mit dem Architektur Explorer (ab Visual Studio 2010, siehe folgende Abbildung) kann ein Entwickler bestehenden Quellcode erforschen und die Abhängigkeiten visualisieren. Die entstandenen Diagramme speichert der Architecture Explorer in einer XML-Datei im Format Directed Graph Markup Language (DGML).

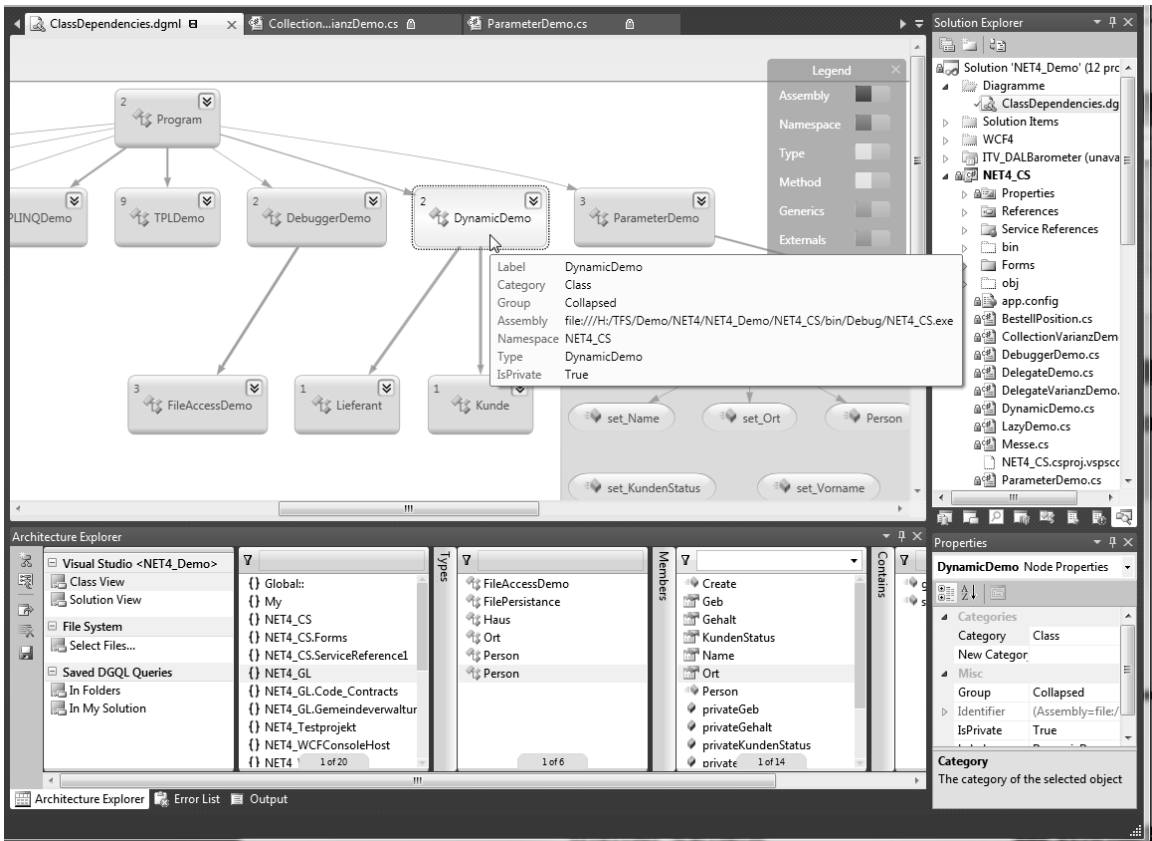


Abbildung 5.55 Der Architecture Explorer erlaubt die Visualisierung der vorhandenen Implementierung

Arbeit mit Datenbanken

Die gesamte Umgebung für die Arbeit mit Daten wurde seit Visual Studio Version 2005 stark erweitert. Zur Verwaltung von Daten existieren nunmehr zwei Fenster: *Server Explorer* und *Datenquellen (Data Sources)* sowie das Menü *Data*.

ACHTUNG Alle Datenbankverwaltungswerkzeuge in Visual Studio 2010 arbeiten nur noch mit Microsoft SQL Server 2005/2008 (inkl. R2) zusammen. Microsoft hat die Unterstützung für SQL Server 2000 in Visual Studio eingestellt [MSFOR01].

Datenverbindungen (Data Connections)

Das Fenster *Server Explorer (Ansicht/Server Explorer)* zeigt im Ast *Datenverbindungen* den Inhalt aller eingebundenen Datenbanken an. Über das Kontextmenü dieses Astes können neue Datenbanken hinzugefügt werden (*Verbindung hinzufügen*). Für das Microsoft SQL Server-DBMS wird auch angeboten, neue Datenbanken direkt aus Visual Studio zu erstellen. Allerdings hat man dabei keine Einstellungsmöglichkeiten für die Datenbankgröße.

Für Microsoft SQL Server-Datenbanken werden in Visual Studio seit Version 2005 mehr Verwaltungsfunktionen angeboten als in früheren Versionen von Visual Studio. Insbesondere ist es möglich, Datenbankdiagramme zu erzeugen, Abfragen zu definieren (auch grafisch) und die Funktion der SQL CLR (Stored Procedures, Trigger, Typen und Funktionen in .NET-Sprachen) zu nutzen.

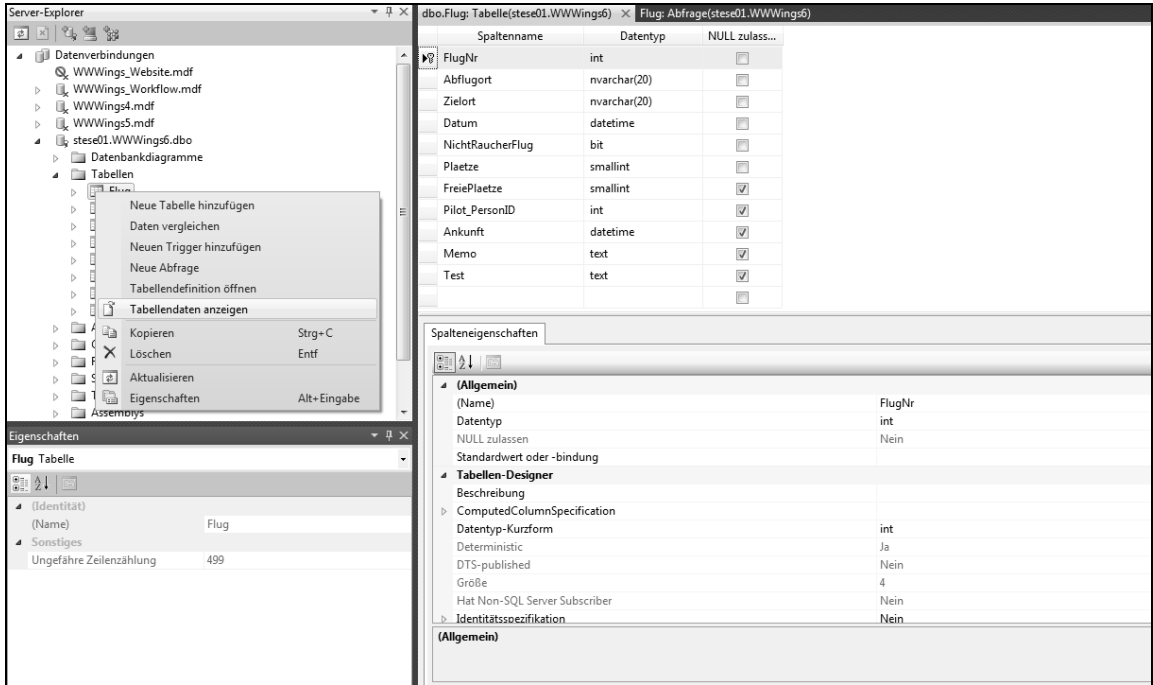


Abbildung 5.56 Datenverbindungen im Server Explorer

T-SQL-Editor

In Visual Studio-Projekten kann man Dateien mit der Erweiterung *.sql* anlegen (entweder über *Add new Item/SQL file* oder Menü *Data/Transact-SQL-Editor/New Query Connection*). Visual Studio bietet dann einen T-SQL-Editor mit Farbhervorhebung und IntelliSense.

TIPP IntelliSense für T-SQL funktioniert dann am besten, wenn man mit einem Microsoft SQL Server 2008 oder höher direkt verbunden ist. Ohne eine Verbindung zu einer Datenbank sind keine Vorschläge für Tabellen und Spaltennamen (wie in folgender Bildschirmabbildung gezeigt) möglich.

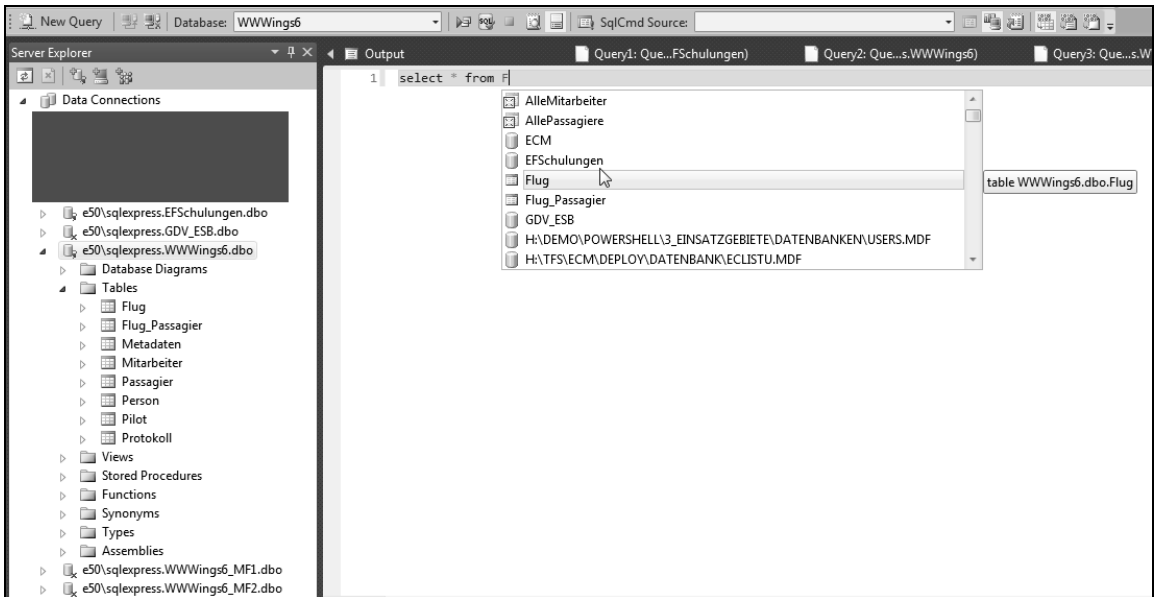


Abbildung 5.57 T-SQL-Editor in Visual Studio 2010

Datenquellen (Data Sources)

Das Fenster *Datenquellen (Data Sources)* zeigt alle im Projekt vorhandenen typisierten DataSets sowie Webservices und Geschäftsobjekte, die explizit als Datenlieferanten hinzugefügt wurden. Der Obermenüpunkt *Daten* enthält die Unterpunkte *Datenquellen anzeigen (Show Data Sources)*, wodurch das Fenster *Datenquellen* eingeblendet wird, sowie *Neue Datenquelle hinzufügen (Add new Data Source)*, was einen Assistenten zum Hinzufügen einer Datenquelle startet.

Die folgenden Bildschirmabbildungen zeigen das Vorgehen zum Erzeugen eines typisierten DataSets. Im ersten Schritt ist auszuwählen, ob eine Datenbank, ein Webservice oder ein Geschäftsobjekt hinzugefügt werden soll.

HINWEIS Das Fenster *Datenquellen (Data Sources)* existiert nicht im VWD. Hier können Datenquellen über die Datenquellensteuerelemente eingefügt werden. Der Assistent für Datenquellensteuerelemente ist teilweise dem *Data Source Configuration Wizard* ähnlich.

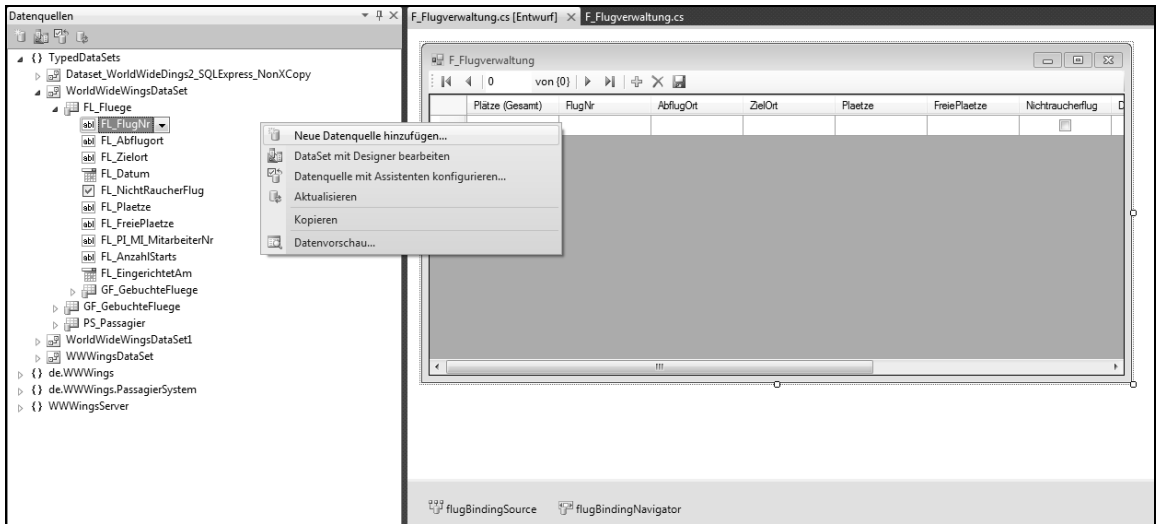


Abbildung 5.58 Datenquellen-Fenster

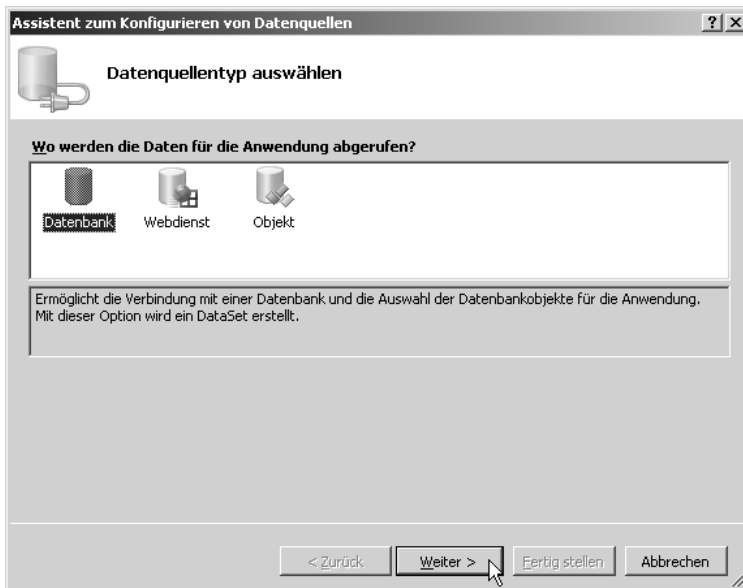


Abbildung 5.59 Auswahl der Art der Datenquelle

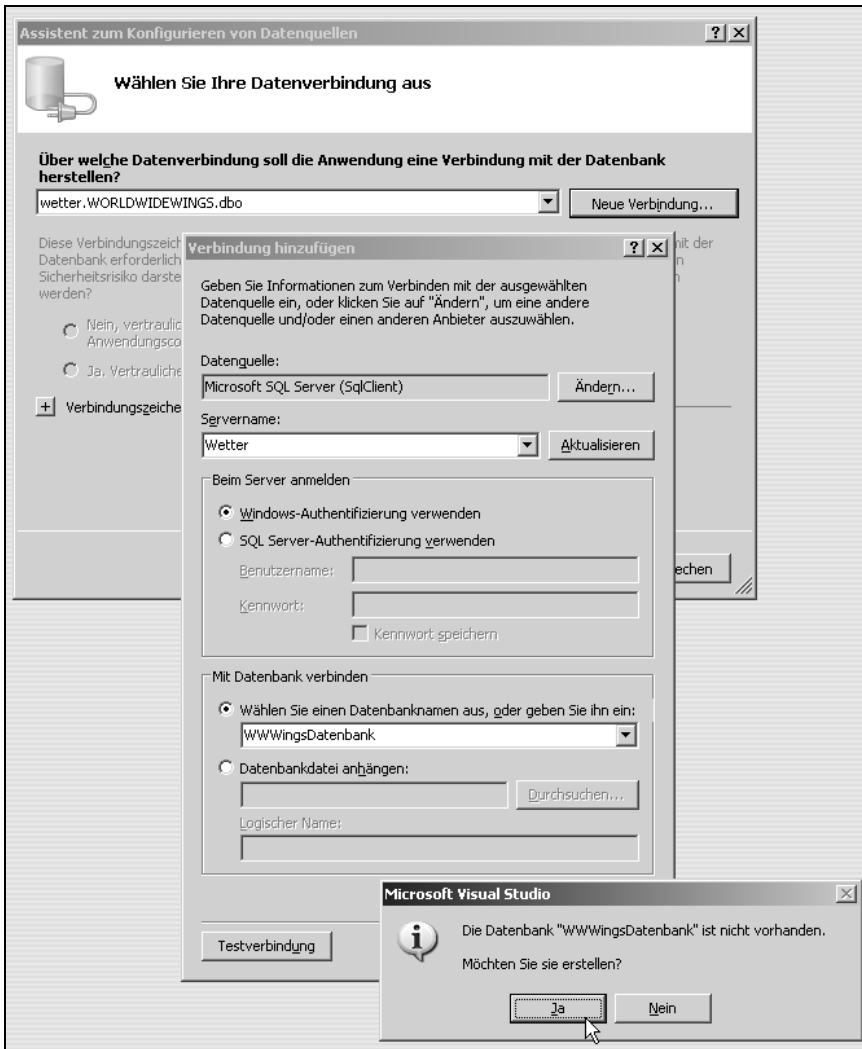


Abbildung 5.60 Auswahl einer Datenbank

Nach Abschluss der Erzeugung einer Verbindungszeichenfolge bietet der Assistent noch an, die Verbindungszeichenfolge in der XML-basierten Anwendungskonfigurationsdatei abzulegen.



Abbildung 5.61 Speicherung der Verbindungszeichenfolge

Im letzten Schritt fordert der Assistent die Information an, welche Daten in das typisierte DataSet aufgenommen werden sollen.

Objektrelationales-Mapping-Designer

Der Objektrelationale-Mapping-Designer für das ADO.NET Entity Framework wird im Kapitel 12 zu »Objektrelationales Mapping (ORM)« besprochen. Darüber hinaus gibt es noch einen ORM-Designer für LINQ to SQL, der in dieser Auflage des Buchs nicht mehr besprochen wird.

Datenbankverwaltungswerkzeuge

In Visual Studio 2010 Premium und Ultimate gibt es spezielle Werkzeuge für die Verwaltung von Datenbanken:

- **Schema Comparison** Vergleich zweier Datenbankschemata
- **Data Comparison** Vergleich des aktuellen Datenbestandes zweier Datenbanken
- **Refactoring für Datenbankobjekte** Umbenennen von Datenbankobjekten (nur verfügbar in der Schemaansicht eines Datenbankprojekts, siehe unten)
- **Datenbanktests** (Siehe Abschnitt »Testen«)

Außerdem gibt es eigene Datenbankprojekte, mit denen man Microsoft SQL Server-Datenbanken unabhängig von anderen Visual Studio-Projekten betrachten und verändern kann.

Schemavergleich

Für einen Schemavergleich legt man zunächst die beiden zu vergleichenden Datenbanken fest. Das Vergleichsergebnis zeigt auch das Schema *Update Script*, mit dem man die Zieldatenbank an die Quelldatenbank angleichen kann.

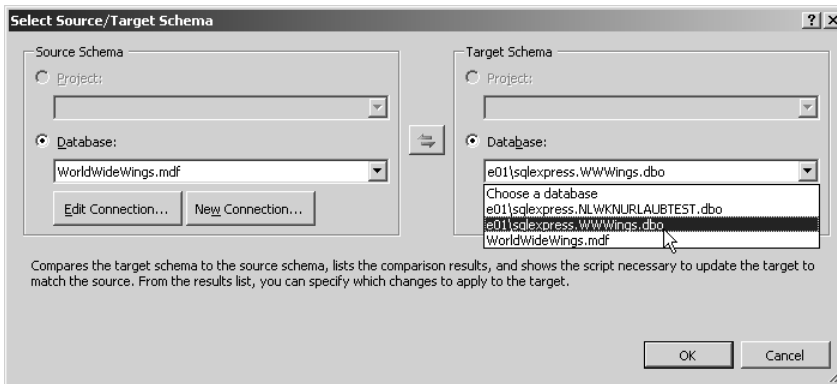


Abbildung 5.62 Festlegung zweier Datenbanken für den Schemavergleich

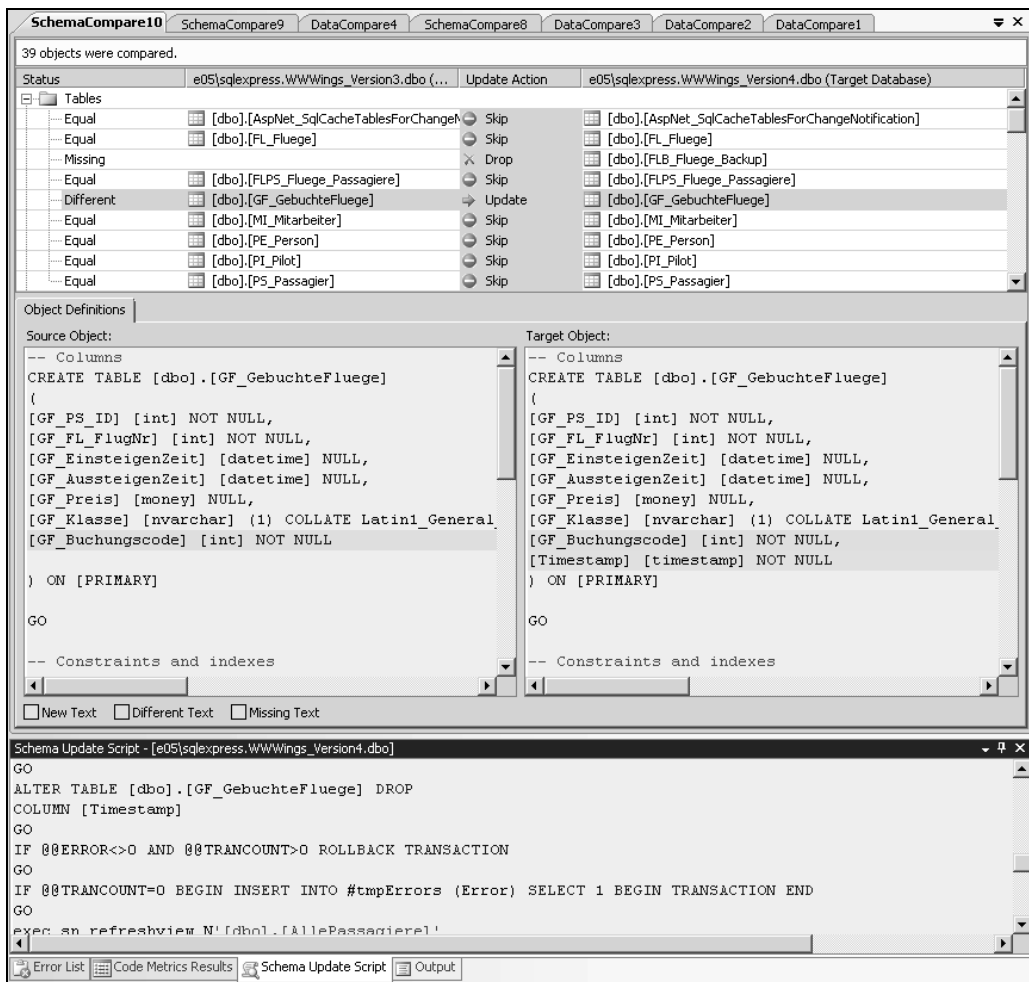


Abbildung 5.63 Ergebnis des Schemavergleichs

Datenvergleich

Auch beim Datenvergleich legt man im ersten Schritt die Datenbanken fest. Im zweiten Schritt wählt man die Datenbankobjekte (Tabellen und Sichten) aus, die verglichen werden sollen. Die Auswertung zeigt dann die hinzugefügten, gelöschten und geänderten Datensätze an. Auf Wunsch kann man ein Aktualisierungsskript (*Data Update Script*) generieren lassen, das die Zieldatenbank an die Quelldatenbank angleicht.

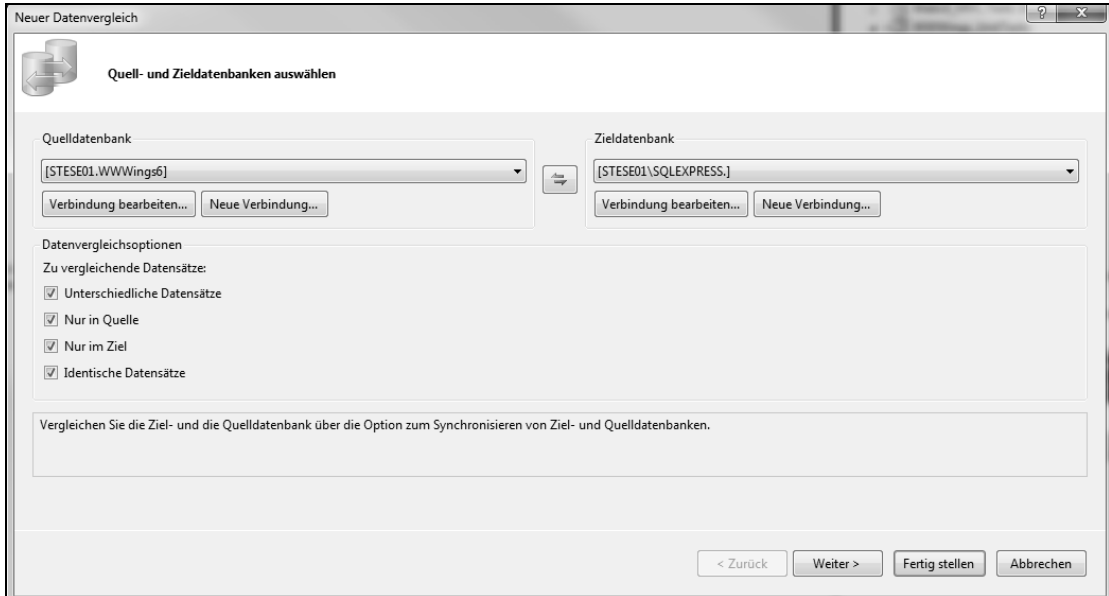


Abbildung 5.64 Festlegung zweier Datenbanken für den Datenvergleich

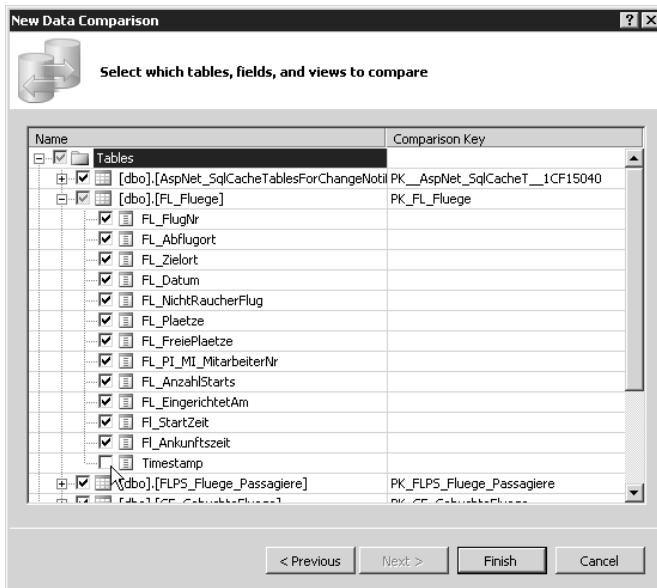


Abbildung 5.65 Festlegung der Vergleichsobjekte

9 tables and/or views were compared.

Source: e05\sqlexpress.WWWings_Version3.dbo Target: e05\sqlexpress.WWWings_Version4.dbo

Object (check to include in update)	Different Records	Only in Source	Only in Target	Identical Records
Tables				
[dbo].[AspNet_SqlCacheTablesForChange1 (Update 1)]	0	0	0	0
[dbo].[FL_Fluege]	4 (Update 4)	4 (Add 4)	0	678
[dbo].[FLPS_Fluege_Passagiere]	0	0	0	0
[dbo].[GF_GebuchteFluege]	0	0	0	0
[dbo].[MI_Mitarbeiter]	0	0	0	8
[dbo].[PE_Person]	0	0	0	92
[dbo].[PI_Pilot]	0	0	0	2
[dbo].[PS_Passagier]	0	0	0	5
[dbo].[sysdiagrams]	0	0	0	1
Views				
Table/View Combination				

Different Records (4) Only in Source (4) Only in Target (0) Identical Records (678)

4 records that exist on both source and target contain different data; 4 records will be updated on target (e05\sqlexpress.WWWings_Version4.dbo).

Update	FL_FlugNr	FL_Abflugort	FL_Abflugort	FL_Zielort	FL_Zielort	FL_Datum	FL_Datum
<input checked="" type="checkbox"/>	103	Berlin	Berlin	Frankfurt	Frankfurt	16.07.2008 04:34:25	16.07.20
<input checked="" type="checkbox"/>	106	Berlin	Berlin	München	München	28.03.2008 15:51:25	28.03.20
<input checked="" type="checkbox"/>	113	Berlin	Berlin	Rom	Rom	10.09.2008 11:06:25	10.09.20
<input checked="" type="checkbox"/>	126	Berlin	Berlin	Paris	Paris	19.07.2008 04:55:25	19.07.20

Data Update Script - [.\sqlexpress.WWWings_Version4]

```

UPDATE [dbo].[FL_Fluege] SET [FL_FreiePlaetze]=79 WHERE [FL_FlugNr]=103
UPDATE [dbo].[FL_Fluege] SET [FL_FreiePlaetze]=25 WHERE [FL_FlugNr]=106
UPDATE [dbo].[FL_Fluege] SET [FL_FreiePlaetze]=190 WHERE [FL_FlugNr]=113
UPDATE [dbo].[FL_Fluege] SET [FL_FreiePlaetze]=137 WHERE [FL_FlugNr]=126
UPDATE [dbo].[AspNet_SqlCacheTablesForChangeNotification] SET [changeId]=33442 WHERE [tableName]='fl_flu
INSERT INTO [dbo].[FL_Fluege] ([FL_FlugNr], [FL_Abflugort], [FL_Zielort], [FL_Datum], [FL_NichtRaucherFlu
INSERT INTO [dbo].[FL_Fluege] ([FL_FlugNr], [FL_Abflugort], [FL_Zielort], [FL_Datum], [FL_NichtRaucherFlu
INSERT INTO [dbo].[FL_Fluege] ([FL_FlugNr], [FL_Abflugort], [FL_Zielort], [FL_Datum], [FL_NichtRaucherFlu
INSERT INTO [dbo].[FL_Fluege] ([FL_FlugNr], [FL_Abflugort], [FL_Zielort], [FL_Datum], [FL_NichtRaucherFlu
ALTER TABLE [dbo].[FLPS_Fluege_Passagiere] ADD CONSTRAINT [FK_FLPS_Fluege_Passagiere_FL_Fluege] FOREIGN K
ALTER TABLE [dbo].[FLPS_Fluege_Passagiere] ADD CONSTRAINT [FK_FLPS_Fluege_Passagiere_PS_Passagier] FOREIG
ALTER TABLE [dbo].[GF_GebuchteFluege] ADD CONSTRAINT [FK_GF_GebuchteFluege_FL_Fluege] FOREIGN KEY ([GF_FL
ALTER TABLE [dbo].[GF_GebuchteFluege] ADD CONSTRAINT [FK_GF_GebuchteFluege_PS_Passagier] FOREIGN KEY ([GF_FL
ALTER TABLE [dbo].[FL_Fluege] ADD CONSTRAINT [FK_FL_Fluege_PI_Pilot] FOREIGN KEY ([FL_PI_MI_MitarbeiterNr
ALTER TABLE [dbo].[MI_Mitarbeiter] ADD CONSTRAINT [FK_MI_Mitarbeiter_MI_Mitarbeiter] FOREIGN KEY ([MI_Vor

```

Error List Code Metrics Results Data Update Script Output

Abbildung 5.66 Ergebnis des Vergleichs und das generierte Aktualisierungsskript

Datenbankprojekte

Mit einem Datenbankprojekt (Anzulegen über New Project/Database Projects) kann man ein Datenbankschema unabhängig von einer Instanz eines Microsoft SQL Server verwalten. Datenbankprojekte bieten zwei Ansichten: Die Projektmappenansicht und die Schemaansicht (siehe folgende Bildschirmabbildungen). In der Schemaansicht ist auch die Refactoring-Funktion *Rename* zum Umbenennen von Datenbankobjekten verfügbar.

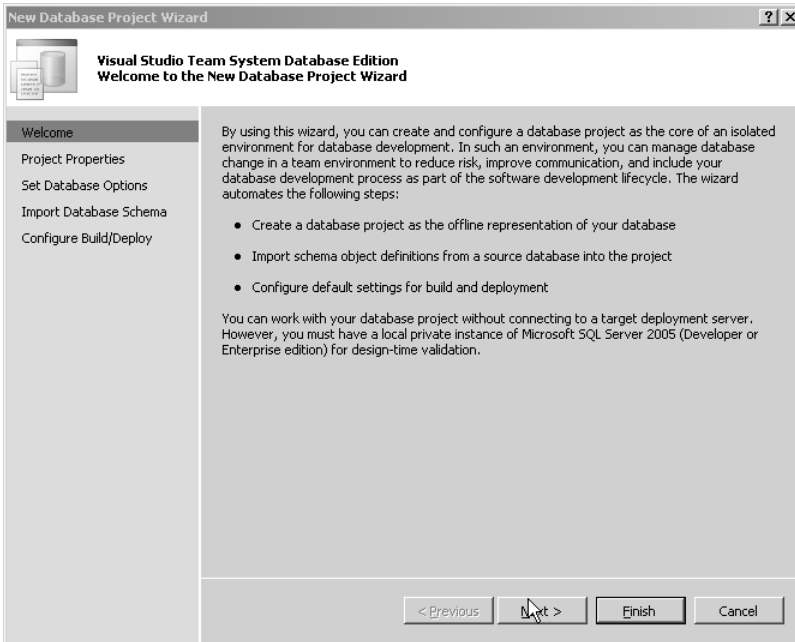


Abbildung 5.67 Anlegen eines Datenbankprojekts mit dem Assistenten. Dabei kann man ein Schema einer bestehenden Datenbank importieren

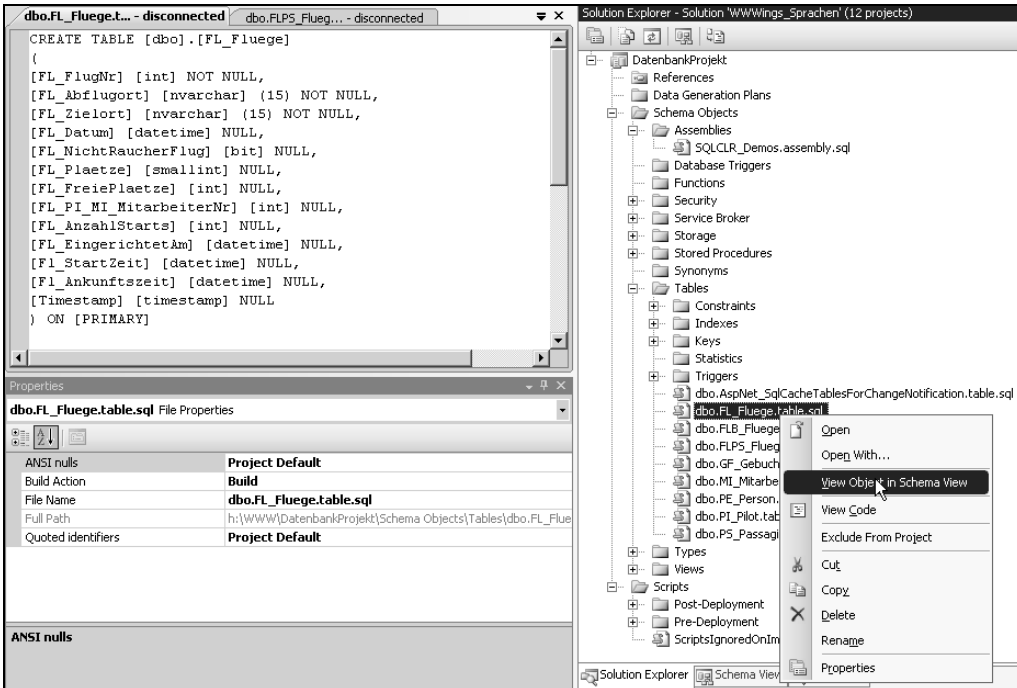


Abbildung 5.68 Ansicht eines Datenbankschemas in der Projektmappenansicht

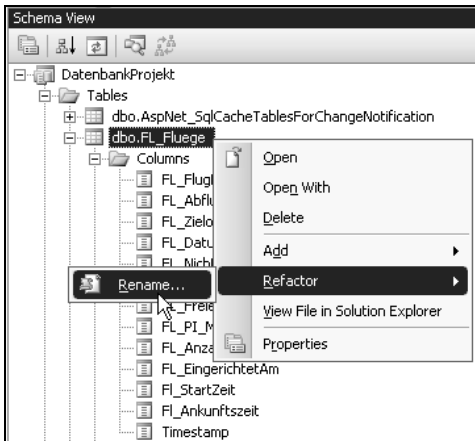


Abbildung 5.69 Ansicht eines Datenbankschemas in der Schemaansicht – inklusive Refactoring-Funktionen

Kompilierung und Ausführung

Dieser Abschnitt beschäftigt sich mit den Möglichkeiten zur Kompilierung und Ausführung von .NET-Anwendungen innerhalb der Entwicklungsumgebung.

Übersetzungskonfigurationen

Visual Studio unterstützt verschiedene Übersetzungskonfigurationen mit verschiedenen Einstellungen (z. B. ob Debug-Informationen generiert werden oder nicht). Standardmäßig existieren die Konfigurationen *Debug* und *Release*.

HINWEIS Unterschiedliche Übersetzungskonfigurationen werden nicht von Projekten nach dem Websitemodell unterstützt. Um diese Funktionen zu nutzen, müssen Sie bei ASP.NET-Anwendungen entweder auf das Web Deployment-Add-In oder das Webanwendungsmodell zurückgreifen.

Über das Auswahlménú *Projektmappenplattformen* (*Solution Platform*) steuert der Entwickler die Plattformoptionen im CLR-Header (siehe Abschnitt zu 64-Bit-.NET im Kapitel 4 »Grundkonzepte des .NET Framework 4.0«). Dabei tauchen die technischen Bezeichnungen *ILOnly* und *32-Bit-Required* dort leider nirgendwo auf. Visual Studio nimmt Bezug auf die Prozessorarchitektur: Die Optionen *AnyCPU* und *x86* erzeugen PE32-Dateien, während *x64* und *Itanium* PE32+ generieren. *x86* setzt das Flag *32-Bit-Required*. *ILOnly* wird automatisch gesetzt, wenn es nur CIL-Code in der Assembly gibt. Folglich ist *AnyCPU* die flexibelste Option mit der größten Plattformunabhängigkeit.

Fehlerliste

Nach dem Übersetzungsvorgang findet man im Fenster *Fehlerliste* folgende Arten von Fehlern:

- Kompilierungsfehler im Code in Hintergrundcodedateien, eigenständigen Codedateien und Inline-Code in ASPX-Dateien

- Validierungsfehler der Inhalte der ASPX-Seiten (wenn dies unter *Extras/Optionen/Texteditor/HTML/Validierung*) aktiviert ist. Die Validierungsart (z. B. XHTML 1.0/1.1, HTML 4.01 oder Internet Explorer 6.0) wählen Sie durch das Auswahlfeld *Zielschema für die Validierung* in der Symbolleiste *Formatierung* aus.
- Strukturfehler in *web.config*-Dateien
- Bezug auf nicht vorhandene User Controls
- Verstöße gegen die Codierungsrichtlinien von Microsoft (wenn in den Projekteigenschaften in der Rubrik *Erstellen* die Option *Codeanalyse aktivieren* angewählt ist)
- Verstöße gegen die Richtlinien für barrierefreie Websites gemäß Content Accessibility Guidelines (WCAG) des W3C und der US Access Board Sektion 509 [ACBO01] (wenn die Validierung in den Projekteigenschaften in der Rubrik *Erstellen/Eingabehilfvalidierung* aktiviert ist)

HINWEIS

Es passiert nicht selten, dass ein Fehler zweimal in der Fehlerliste erscheint und zwar einmal mit dem Verweis auf die Codedatei, in der der Fehler passiert, und einmal mit dem Verweis auf die vom ASP.NET Compiler automatisch daraus generierte Datei. Wie die nachfolgende Bildschirmabbildung zeigt, kann es vorkommen, dass die Fehlermeldung einmal in Deutsch und einmal in Englisch erscheint.

Fehlerliste					
34 Fehler 3 Warnungen 0 Meldungen					
	Beschreibung	Datei	Zeile	Spalte	Projekt
10	'Application' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	App_Web_wgmbxwie.3.vb	105		
4	'CB2' is ambiguous.	App_Web_wgmbxwie.3.vb	61		
2	'Context' is not a member of 'CB2'.	App_Web_wgmbxwie.3.vb	46		
3	'Context' is not a member of 'CB2'.	App_Web_wgmbxwie.3.vb	52		
18	'CreateResourceBasedLiteralControl' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	CB_Manuell.aspx	1		
13	'Form1' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	App_Web_wgmbxwie.3.vb	130		
8	'GetWrappedFileDependencies' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	App_Web_wgmbxwie.3.vb	96		
17	'InitializeCulture' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	CB_Manuell.aspx	1		
7	'ReadStringResource' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	App_Web_wgmbxwie.3.vb	92		
9	'Server' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	App_Web_wgmbxwie.3.vb	99		
11	'Session' is not a member of 'ASP.seitenmodelle_cb_manuell.aspx'.	App_Web_wgmbxwie.3.vb	113		
35	"Application" ist kein Member von "ASP.seitenmodelle_cb_manuell.aspx".	CB_Manuell.aspx	23	3	H:_DEV\WFBuch\
33	"CB2" ist nicht eindeutig.	CB_Manuell.aspx	18	3	H:_DEV\WFBuch\
29	"Context" ist kein Member von "CB2".	CB_Manuell.aspx	18	3	H:_DEV\WFBuch\
31	"Context" ist kein Member von "CB2".	CB_Manuell.aspx	18	3	H:_DEV\WFBuch\
36	"Session" ist kein Member von "ASP.seitenmodelle_cb_manuell.aspx".	CB_Manuell.aspx	23	3	H:_DEV\WFBuch\
5	Class 'seitenmodelle_cb_manuell.aspx' must implement 'ReadOnly Property IsReusable()' As Boolean for interface 'System.Web.IHttpHandler'. Implementing property must have matching 'ReadOnly' or 'WriteOnly' specifiers.	App_Web_wgmbxwie.3.vb	62		
6	Class 'seitenmodelle_cb_manuell.aspx' must implement 'Sub ProcessRequest(context As HttpContext)' for interface 'System.Web.IHttpHandler'.	App_Web_wgmbxwie.3.vb	62		
34	Der Name "Response" wurde nicht deklariert.	CB_Manuell.aspx	21	2	H:_DEV\WFBuch\

Abbildung 5.70 Eine Fehlermeldung, die falsch ist und gleich zweifach vorkommt. Im Code gibt es tatsächlich nur eine Klasse CB2.

Microsoft Build (MSBuild)

Während in Visual Studio .NET 2002/2003 der Übersetzungsvorgang weitestgehend eine *Black Box* war, liefert Microsoft seit dem .NET Framework 2.0 ein eigenes Build-Werkzeug (*msbuild.exe*), das auch von Visual Studio, seit Version 2005 zur Übersetzung der Projekte verwendet wird. MSBuild ist Teil des .NET Framework Redistributable (ab Version 2.0). Die von Visual Studio erzeugten Projektdateien sind gültige Eingabedateien für MSBuild, welche alle notwendigen Einstellungen enthalten, sodass eine Übersetzung der Projekte auch ohne die Installation von Visual Studio möglich ist.

TIPP Sie können Visual Studio-Projekte übersetzen, ohne die Entwicklungsumgebung selbst zu starten, indem Sie das mit dem .NET Framework ausgelieferte Kommandozeilenwerkzeug *msbuild.exe* direkt aufrufen. Die Visual Studio-Projektdateien und Projektmappendateien sind gültige Eingabedateien für *msbuild.exe*.

Beispiel	Erläuterung
<code>MSBuild.exe WWWings_ConsoleUI_CS.csproj /t:Clean;Build /p:Configuration=Debug</code>	Löschen des Ausgabezeichnisses und Übersetzen des angegebenen Projekts im Modus <i>Debug</i>
<code>MSBuild.exe WWW_Web.sln /t:Rebuild /p:Configuration=Release</code>	Neuübersetzen der angegebenen Projektmappe im Modus <i>Release</i>

Tabelle 5.13 Beispiele zum Einsatz von MSBuild

Neu in Visual Studio seit Version 2005 ist die Funktion *Projektmappe bereinigen* (*Clean Solution*) im *Erstellen*-Menü bzw. die Funktion *Bereinigen* im Kontextmenü der Projekte. Hiermit werden alle Ausgabedateien der Projekte gelöscht.

Debugger

Visual Studio unterstützt das Debugging sowohl von lokalen als auch von entfernten .NET-Anwendungen. Dabei vergessen Entwickler oft, dass Remote Debugging nur funktioniert, wenn auf dem Rechner, auf dem die zu testende Anwendung, aber kein Visual Studio läuft, ein kleiner Gegenpart zum Debugger installiert wird. Auf der Installations-CD / -DVD von Visual Studio gibt es dazu den Ordner */Remote Debugger* mit der Datei *rdbgsetup.exe*. Diese Installationsroutine muss vor dem ersten Debug-Versuch auf dem entfernten System ausgeführt werden.

TIPP Auch der Debugger hat in Visual Studio 2010 einige kleine, feine Verbesserungen sowie eine große Verbesserung (siehe »IntelliTrace«) erhalten.

Start des Debuggers

Der Debugger startet auf folgenden Wegen:

- Die Anwendung erzeugt eine Ausnahme
- Die Anwendung erreicht einen gesetzten Haltepunkt
- Die Anwendung führt den Befehl `System.Diagnostics.Debugger.Break()` oder einen vergleichbaren sprachspezifischen Befehl (z. B. *Stop in Visual Basic*) aus

Bei welchen Ausnahmen der Debugger startet, kann über das Menü *Debuggen/Ausnahmen* (*Debug/Exception*) konfiguriert werden. Haltepunkte können durch einen Klick in die graue Leiste neben der Zeilennummer oder über *Debuggen/Neuer Haltepunkt* gesetzt und mit Bedingungen versehen werden.

Setzen eines Haltepunktes

Einen Haltepunkt setzt man mithilfe der Taste **[F9]** oder mit einem Mausklick in die graue Leiste vor den Zeilennummern im Editor.

Ab Visual Studio 2010 kann man einem Haltepunkt Bezeichnungen (*Labels*) zuordnen. Ein Haltepunkt kann mehrere Labels besitzen. Haltepunkte kann man importieren und exportieren. Die Exportfunktion findet man im Kontextmenü der Haltepunktliste links im Editor. Die Importfunktion findet man in dem Fenster *Breakpoints* hinter einem Symbol verborgen.

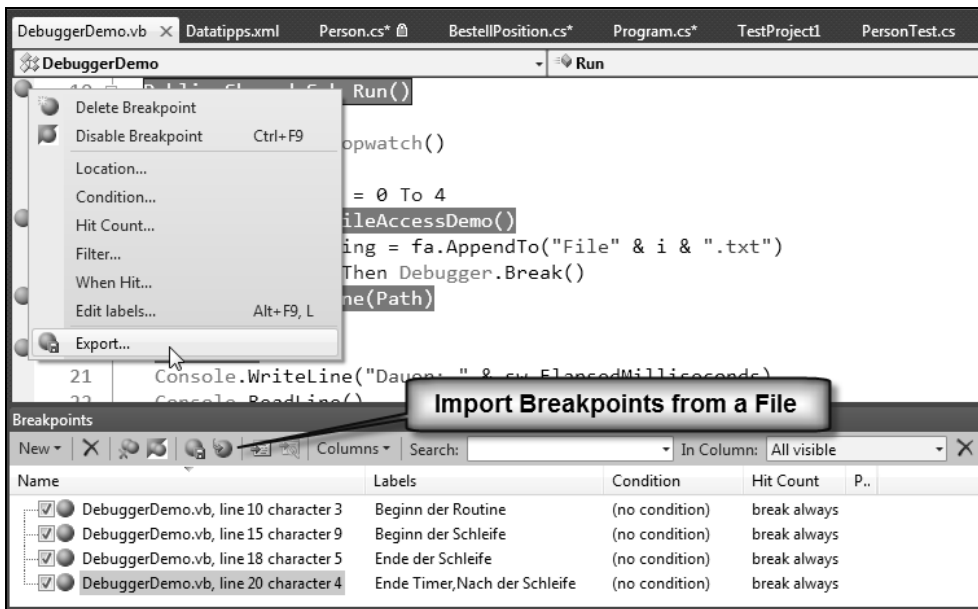


Abbildung 5.71 Setzen eines Haltepunkts

Funktionen im Haltemodus

Visual Studio bietet im Haltemodus folgende Funktionen:

- Durchlaufen der Befehle im Einzelschrittmodus
- Fortsetzen der Anwendung
- Anzeige und Manipulation des aktuellen Werts von Variablen direkt im Codefenster
- Anzeige von Variablenwerten im Fenster *Locals*
- Anzeige der Aufrufreihenfolge der Methoden (*Debuggen/Fenster/Aufrufliste* – engl. *Call Stack*)

- Anzeige der laufenden Threads (*Debuggen/Fenster/Threads*)
- Anzeige der laufenden Prozesse (*Debuggen/Fenster/Prozesse*)
- Verändern des Quelltextes (in bestimmten Grenzen)

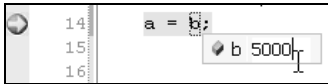


Abbildung 5.72 Veränderung eines Variablenwerts im Haltemodus

Zu den Verbesserungen in Visual Studio 2010 zählen die Debugger Data Tipps. Schon bisher konnte man während einer Debugger-Sitzung mit der Maus Variablen überfahren, um ihren aktuellen Wert anzusehen. Wenn man aber die Maus wegbewegt hat, war die Anzeige sofort verschwunden. Nun kann man über eine Heftnadel (siehe Abbildung) die Anzeigen »festheften«. Dieser Heftvorgang gilt nicht nur für die aktuelle Sitzung, sondern übersteht auch einen Neustart von Visual Studio. Man kann diese *Debugger Data Tipps* sogar exportieren (Menü *Debug/Export Data Tipps*) und auf einem anderen System wieder importieren (Import Data Tipps). Da man außerdem zu jedem Debugger Data Tipp noch einen Kommentar schreiben kann, eignet sich die Funktion auch dazu, Debugging-Informationen an andere Teammitglieder weiterzugeben.

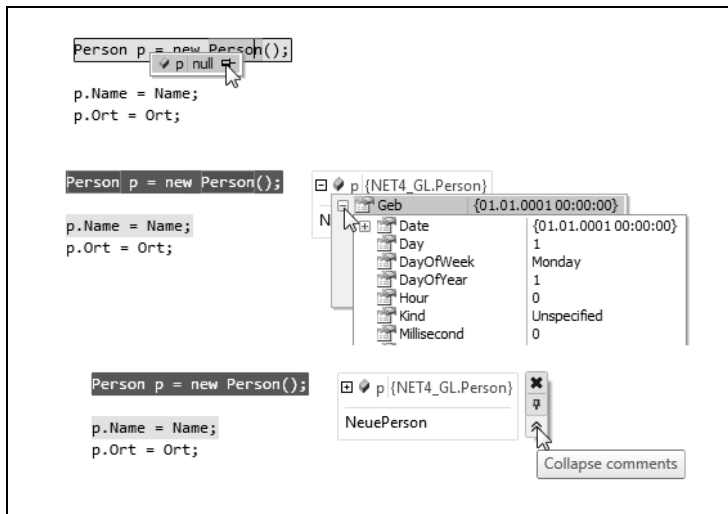


Abbildung 5.73 Debugger Data Tipps

Bearbeiten und Fortsetzen (Edit & Continue)

Die Möglichkeit, den Quelltext während des Debugging-Vorgangs zu verändern und dann die Anwendung fortzusetzen (Edit & Continue – kurz E&C), ist eine »Neuheit« in Visual Studio seit Version 2005. Diese Funktion war in Visual Studio 6.0 verfügbar und wurde in Visual Studio .NET 2002/2003 von vielen Entwicklern vermisst.

HINWEIS

E&C ist nicht verfügbar im 64-Bit-Modus. Für E&C muss man auf 64-Bit-Systemen im WOW arbeiten, indem man als Plattform x86 wählt!

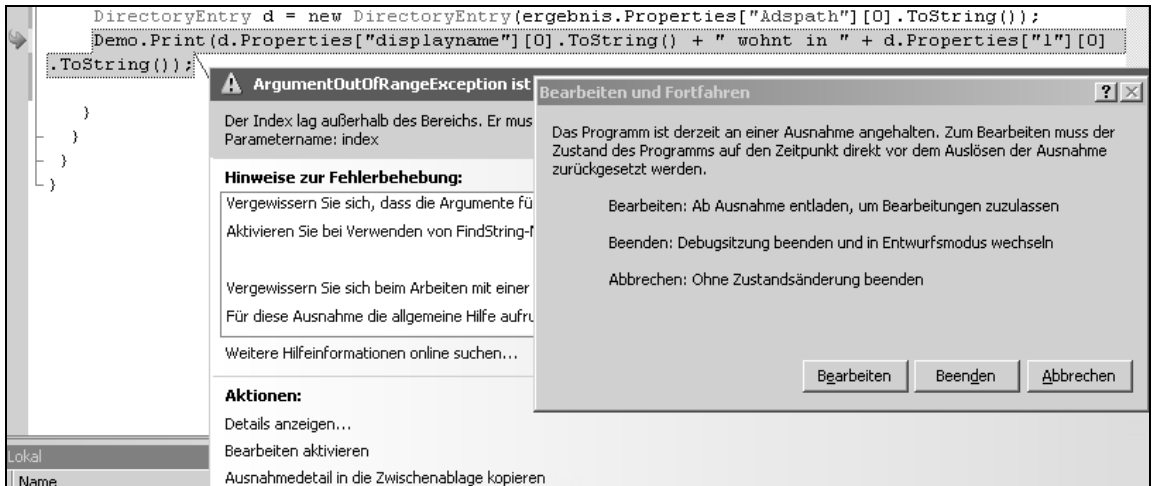


Abbildung 5.74 Einsatz von Edit & Continue für C# in Visual Studio

Nicht alle Änderungen erlauben eine Fortsetzung. Beispiele für Änderungen, nach denen keine Fortsetzung erfolgen kann, sind:

- Ändern des Namens einer Klasse
- Hinzufügen von Feldern zu einer Klasse
- Hinzufügen oder Entfernen von Methoden
- Hinzufügen oder Entfernen von Methodenparametern

Steuerung der Debugger-Anzeige

Beim Betrachten von aktuellen Variableninhalten während des Debuggings ist es oft nicht einfach, den eigentlichen Inhalt der Variablen zu erfassen, wenn es sich um eine komplexe Datenstruktur wie beispielsweise ein Hashtable-Objekt oder ein DataSet-Objekt handelt.

Visual Studio bietet mehrere Hilfsinstrumente, um die Anzeige während des Debugging-Vorgangs zu beeinflussen:

- Durch die Annotation `DebuggerDisplayAttribute` kann festgelegt werden, welche Daten die Entwicklungsumgebung zu einer Variablen anzeigt
- Durch `DebuggerBrowsableAttribute` kann ein Datenmitglied vor dem Debugger versteckt werden
- Durch `DebuggerTypeProxyAttribute` kann ein Typ als ein anderer angezeigt werden. Üblicherweise wird eine abgeleitete Klasse hier so wie ihre Basisklasse dargestellt.
- Durch *Debugger Visualizer* kann für komplexe Variableninhalte eine adäquate Darstellung angeboten werden. In Visual Studio sind solche Visualizer für XML, HTML, DataSets und Datatables vorhanden. Ein Typ zeigt durch `DebuggerVisualizerAttribute` an, wie er sinnvoll dargestellt werden kann. Eigene Visualizer kann man mit Klassen realisieren, die `IDebugVisualizer` implementieren. Die Klasse `Microsoft.VisualStudio.DebuggerVisualizers.DialogDebuggerVisualizer` enthält dafür ein Muster. Visual Studio bietet eine Elementvorlage *Debugger Visualizer* an.

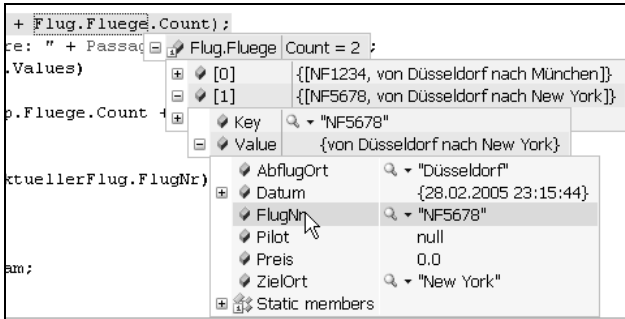


Abbildung 5.75 Anzeige einer SortedList im Debugger

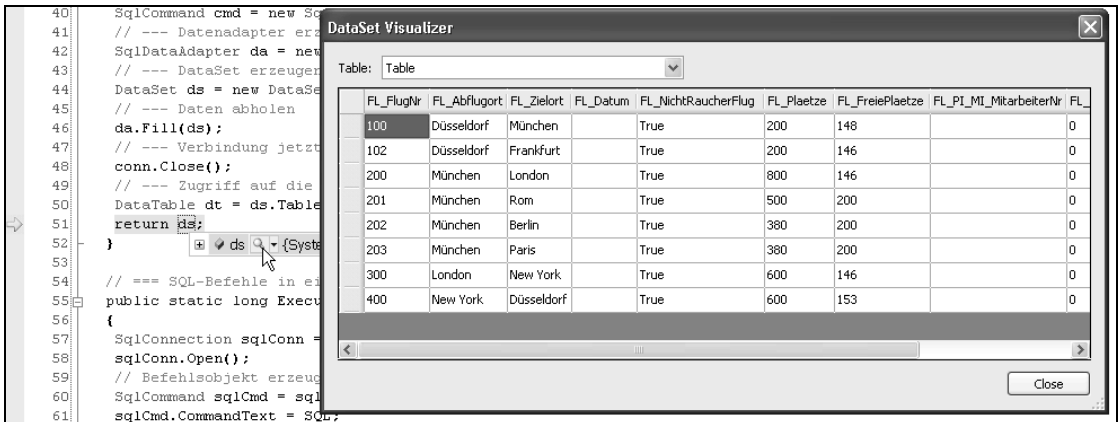


Abbildung 5.76 Data Visualizer für ein DataSet

Anwendung auf eigene Klassen

Das folgende Anwendungsbeispiel zeigt die Nutzung des DebuggerDisplayAttribute für die Klasse Flug und das Datenmitglied Pilot.

```
[System.Diagnostics.DebuggerDisplay("Flug Nr={FlugNr}")]
public partial class Flug //: System.EnterpriseServices.ServicedComponent
{
    ...
    [System.Diagnostics.DebuggerBrowsable(System.Diagnostics.
    DebuggerBrowsableState.Collapsed)]
    public Single Preis;
    [System.Diagnostics.DebuggerDisplay("Pilot={Pilot.GanzerName}")]
    public de.WWWings.MitarbeiterSystem.Pilot Pilot;
}
```

Listing 5.1 Anwendung des DebuggerDisplayAttribute

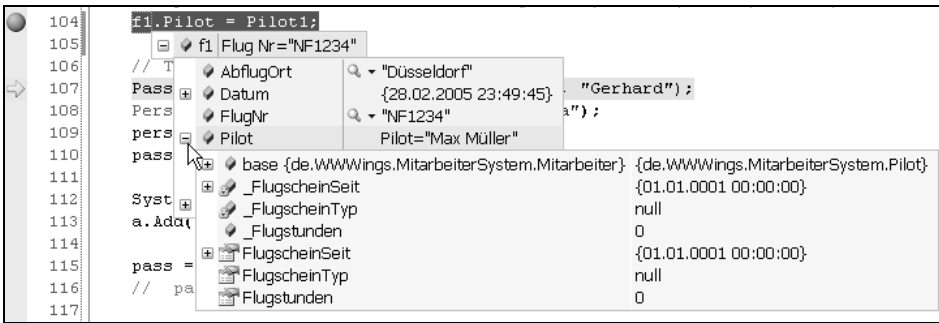


Abbildung 5.77 Anzeige der Klasse Flug im Debugger

Direktfenster (Intermediate Window)

Das *Direktfenster* (*Intermediate Window*) erlaubt es, während des Debuggings beliebige Befehle im aktuellen Kontext auszuführen. Durch ein vorangestelltes Fragezeichen können Ausgaben erzeugt werden. Das *Direktfenster* bietet auch IntelliSense (siehe Abbildung).

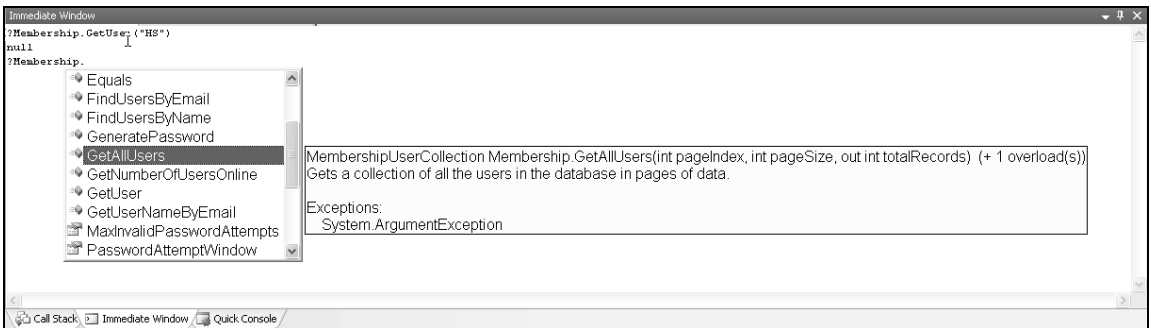


Abbildung 5.78 Das Direktfenster

Objekttestcenter (Object Test Bench, OTB)

Das Objekttestcenter (engl. *Object Test Bench*, *OTB*) ist ein Instrument zum Debuggen und Testen von Anwendungen innerhalb von Visual Studio. Es ermöglicht das Instanzieren von Klassen und den Aufruf von Methoden in Objekten, ohne expliziten Aufrufcode implementieren und die Anwendung starten zu müssen. Aus der Ansicht *Klassenansicht* (*Class View*) oder aus einem Klassendiagramm heraus kann der Entwickler eine Instanz der Klasse erzeugen, die dann grafisch in der Object Test Bench angezeigt wird. Im Kontextmenü des grafischen Objekts stehen daraufhin alle Methoden des Objekts zum direkten Aufruf zur Verfügung. Statische Methoden können unmittelbar aus der Klassenansicht heraus aufgerufen werden.

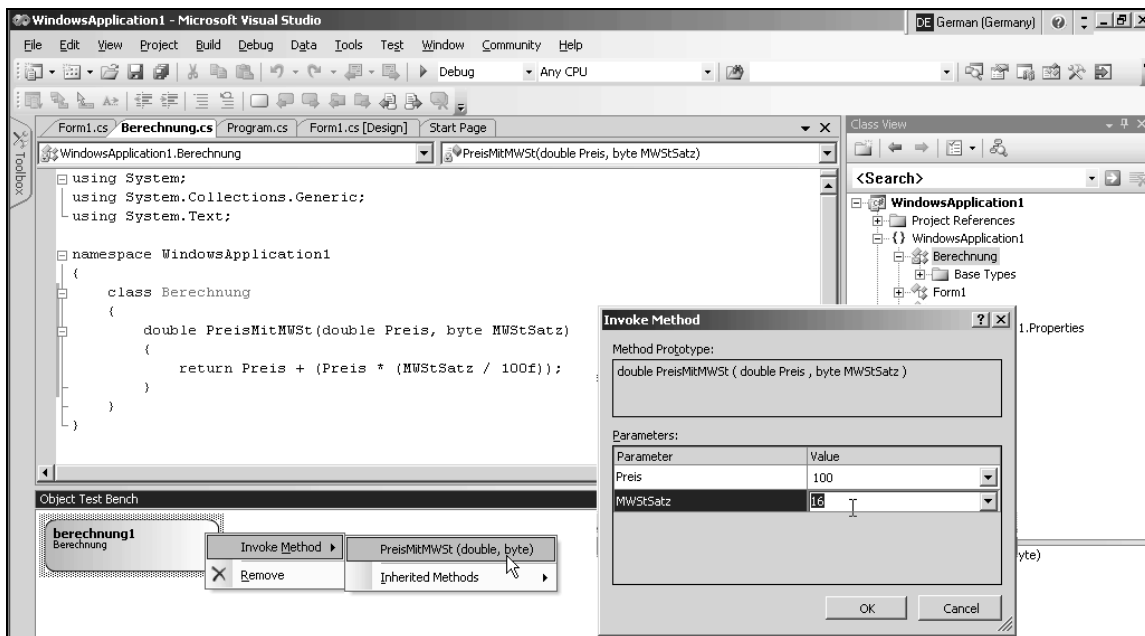


Abbildung 5.79 Objekttestcenter (Object Test Bench)

Debugging des Quellcodes des .NET Framework

Am 16. Januar 2008 hat Microsoft unter dem Titel »*.NET Reference Source Project*« den Quellcode einiger .NET-Bibliotheken zur Einsicht innerhalb des Visual Studio-Debuggers freigegeben. Anders als im Rahmen des Rotor-Projekts kann man nicht C#-Quelldateien herunterladen, sondern Visual Studio-Debugger-Dateien (.pdb), die man nur im Rahmen eines Debugging-Vorgangs in Visual Studio nutzen kann. Beim Aufruf einer Property oder einer Methode in einer der unterstützten .NET-Bibliotheken kann man im Rahmen eines Debugging-Vorgangs »hineinschreiten« (Taste **F11**) oder wie im Menü *Debug/Step Into* wie in selbstentwickelten Programmcode. Visual Studio zeigt dann den Quellcode und die von den Microsoft-Entwicklern hinterlegten Kommentare an.

ACHTUNG Das Betrachten des Quellcodes der von Microsoft geschriebenen .NET-Bibliotheken funktioniert nur mit einem Internet-Zugang. Zwar werden die Debugger-Dateien lokal zwischengespeichert (dieses Verzeichnis legt man unter *Tools/Options/Debugging/Symbols* fest), aber Visual Studio nimmt immer Kontakt zu den Servern bei Microsoft auf. Ohne einen Online-Zugang bekommt man den Quellcode nicht zu sehen.

Voraussetzung ist die Konfiguration von Visual Studio:

- Unter *Tools/Options/Debugging/General* muss *Enable Just My Code* deaktiviert werden
- Unter *Tools/Options/Debugging/General* muss *Enable Source Server Support* aktiviert werden
- Unter *Tools/Options/Debugging/Symbols* muss der Symbolserver <http://referencesource.microsoft.com/symbols> eingetragen werden

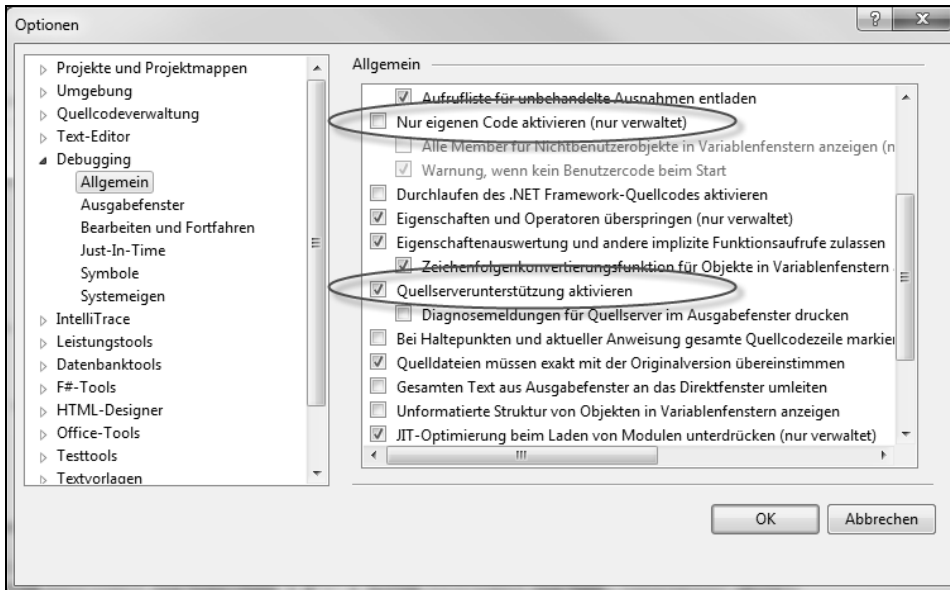


Abbildung 5.80 Einstellung für das Debugging des .NET Framework-Quellcodes

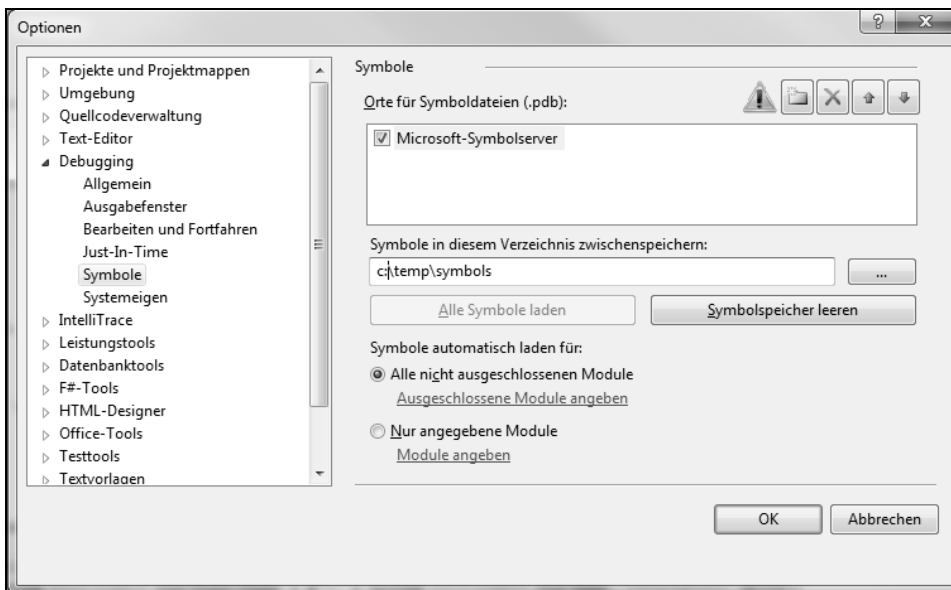


Abbildung 5.81 Einstellung für den .NET Framework-Quellcodeserver

Beim Debugging muss man dann jeweils an einem Haltepunkt die Symboldateien für die gewünschten .NET-Assemblies anfordern. Dies erfolgt mit dem Befehl *Load Symbols* entweder im Fenster *Call Stack* oder *Modules*. Da nur *Modules* alle notwendigen Assemblies zeigt, ist dies meist der bessere Ort. Der Entwickler muss selber entscheiden, welche Assemblies er laden will. Im Zweifel findet man die Information über die Assembly, in der sich eine Klasse befindet, in der MSDN-Dokumentation zu der Klasse.

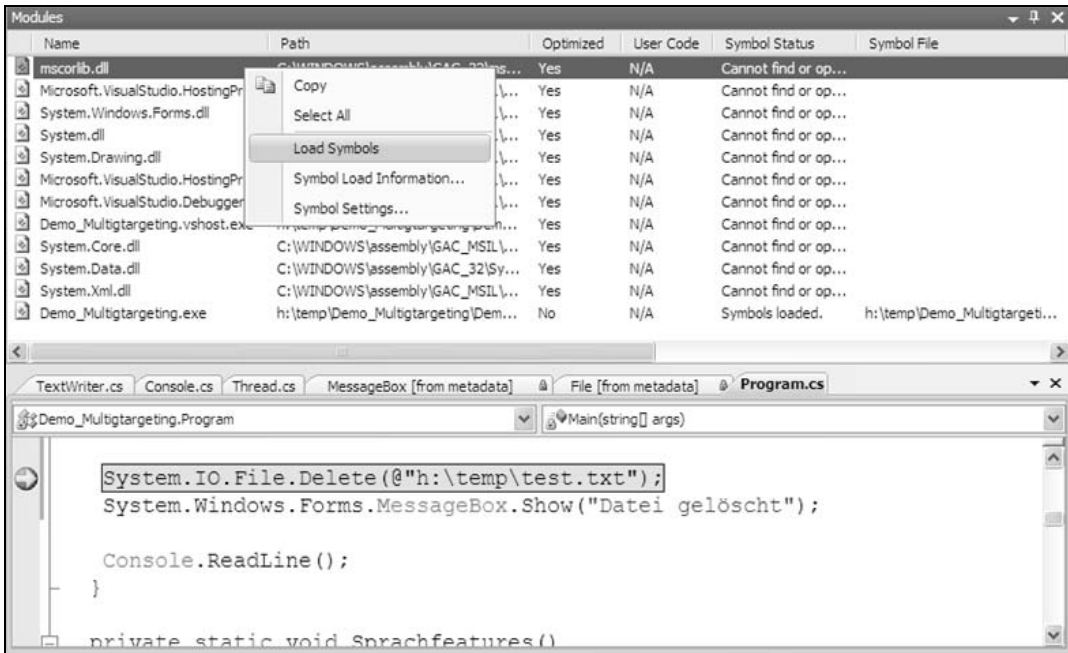


Abbildung 5.82 Laden des Quellcodes für die mscorlib.dll, damit die Methode Delete() in der Klasse System.IO.File im Quellcode betrachtet werden kann

Beim Laden der Symboldatei muss man die Bedingungen von Microsoft akzeptieren.

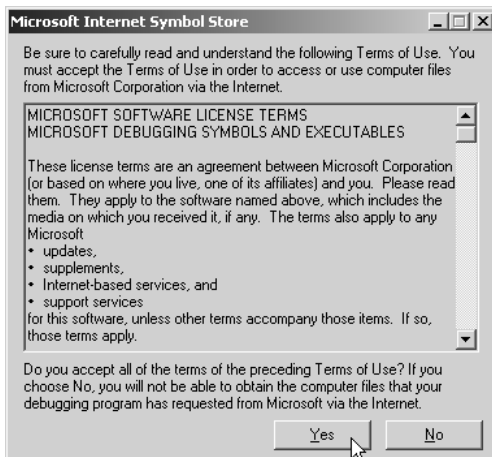


Abbildung 5.83 Man muss Microsofts Bedingungen akzeptieren

Sobald die entsprechende Assembly von dem Redmonder Symbolserver geladen ist, erscheinen die entsprechenden Zeilen im Call Stack-Fenster nicht mehr in grau, sondern in schwarz. Sodann funktioniert *Debug/Step Into* (Taste **F11**).

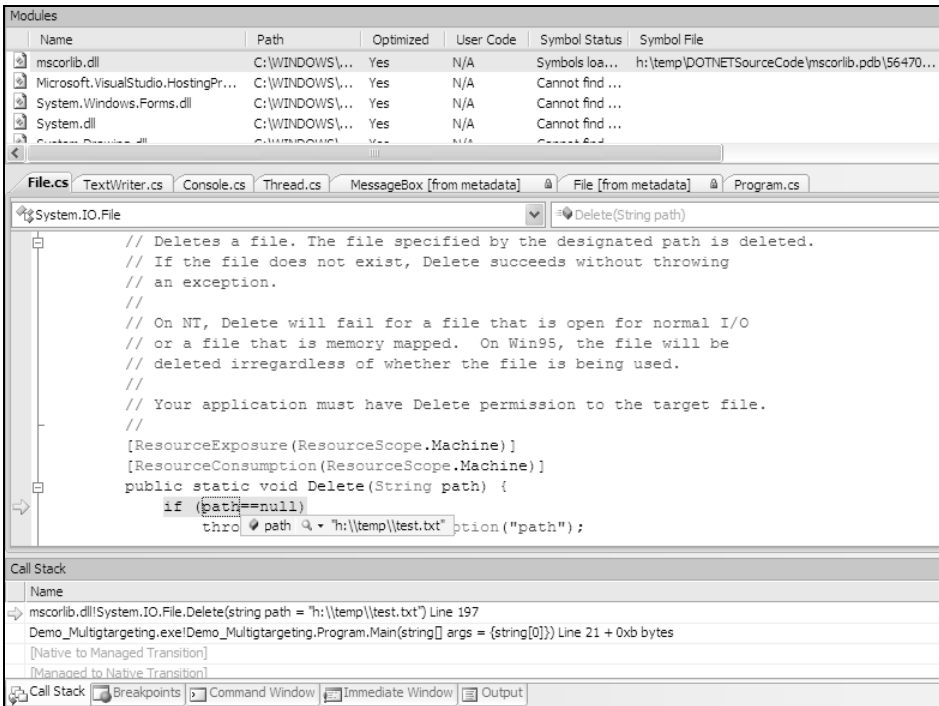


Abbildung 5.84 Debugging der Methode Delete()

WICHTIG

Man muss *Load Symbols* nach einem Neustart des Debuggers jeweils erneut für jede Bibliothek ausführen. Die PDB-Dateien werden aber in einem lokalen Verzeichnis zwischengespeichert und müssen daher nicht immer wieder von dem Symbolserver geladen werden.

Man kann derzeit auch nicht alle *pdb*-Dateien auf einmal laden, sondern immer nur einzeln anfordern.

HINWEIS

Weiterhin verfügbar ist auch der Rotor-Quellcode in Form von C#-Quelldateien im Rahmen der Shared Source Common Language Infrastructure. Allerdings enthält Rotor nicht alle der im .NET Reference Source Project verfügbaren Bibliotheken.

IntelliTrace

Die große neue Funktion im Debugger ist IntelliTrace. Während der Beta-Phase war der Name dieser Funktion noch sprechender: Historical Debugger. IntelliTrace zeichnet während einer Debugger-Sitzung laufend Daten auf. Dies sind zum einen Interaktionen mit dem System (z.B. Mausklicks, Dateisystemzugriffe, Registryzugriffe, Starten von Threads, Datenbankzugriffe, Datenbindung, Laden von XML-Dokumenten) und zum anderen Methodenaufrufe innerhalb des Programmcodes. Die erste Abbildung zeigt die Debugger-Situation in der Ereignisansicht (Events View) und die zweite Abbildung in der Aufrufansicht (Call View). In beiden Ansichten kann man an die einzelnen Zwischenstationen zurückspringen und dort auch konkrete Werte der Vergangenheit sehen. Leider bezieht sich diese Werteanzeige immer nur auf Methodenparameter. Die Inhalte lokaler Variablen, der Instanzen-Variablen und statischer Variablen werden nicht gespeichert; wie bisher sieht man von ihnen nur den Zustand am aktuellen Haltepunkt.

HINWEIS Leider gibt es die IntelliTrace-Funktion aber nur dann, wenn man in Ultimate-Version von Visual Studio 2010 investiert.

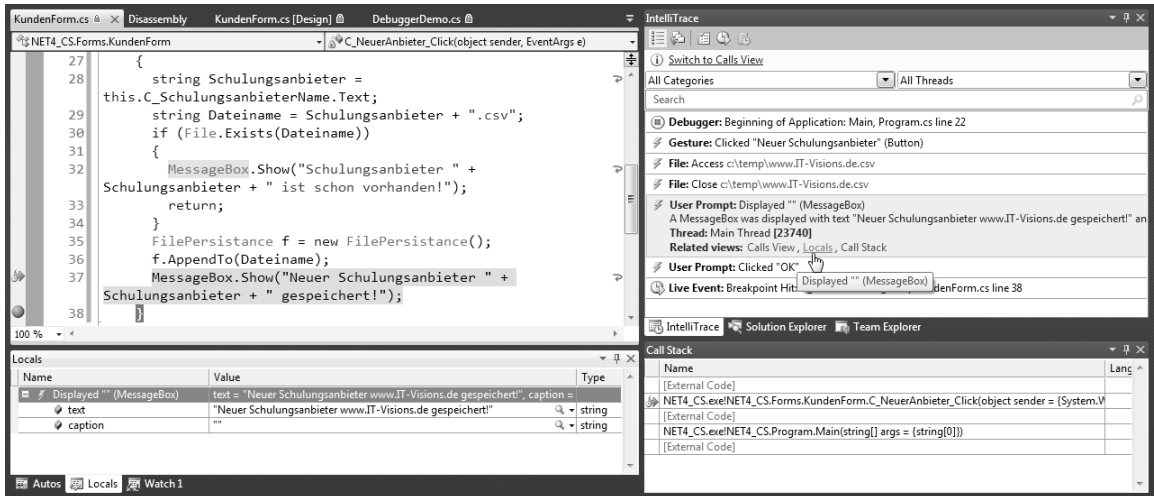


Abbildung 5.85 IntelliTrace-Ansicht für Ereignisse (Events View)

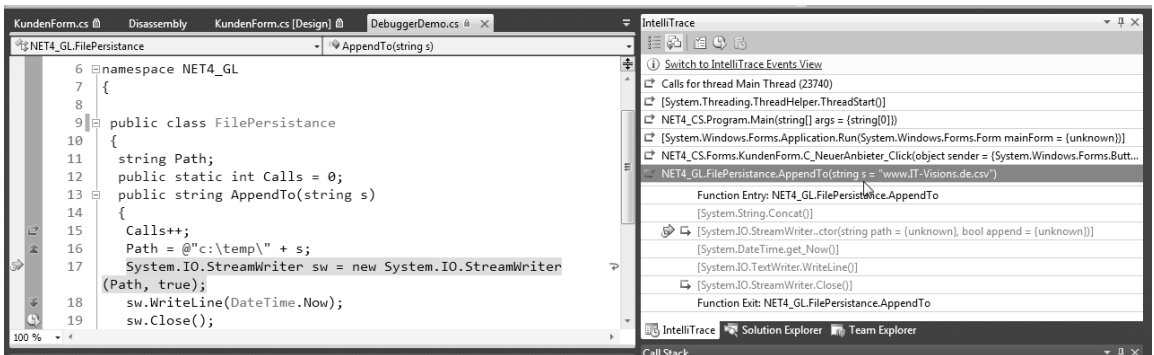


Abbildung 5.86 IntelliTrace-Ansicht für Ereignisse (Events View)

Interessant bei IntelliTrace ist, dass man diese Daten auch außerhalb von Visual Studio mit dem Kommandozeilenwerkzeug *intellitrace.exe* aufzeichnen kann. Die entstandenen *.tdlog*-Dateien können dann in Visual Studio geöffnet werden, um wieder zu der gleichen Ansicht zu kommen, die man beim Debugging in Visual Studio gehabt hätte. Einigen Entwicklern funkeln nun schon die Augen bei dem Gedanken, das Werkzeug auf Kundensystemen, bei denen ihre Anwendung nicht korrekt läuft, einzusetzen.

Einschränkungen auf 64-Bit-Systemen

Drei Debugging-Funktionen von Visual Studio stehen auf 64-Bit-Systemen nicht zur Verfügung:

- gemischtes Debuggen von Managed Code und Native Code
- Ändern & Weitermachen (Edit & Continue) beim Debuggen
- IntelliTrace

Durch den lokalen Einsatz von Remote Debugging erreicht Microsoft aber, dass man auch den 64-Bit-Just-in-Time-Compiler aus dem 32-Bit-Visual Studio heraus entlassen kann.

XSLT-Debugging

Mit Haltepunkten, Einzelschrittmodus und der Anzeige der aktuellen Variableninhalte kann man die Ausführung einer XSL-Transformation studieren. Den Debugger erreicht man über *Debug XLST* im Menü *XML*, wenn man sich in einer XML-Datei (.xml) oder Transformationsdatei (.xsl oder .xslt) befindet. Wenn es keine im Properties-Fenster in den Eigenschaften *Stylesheet* bzw. *Input* hinterlegte Verknüpfung zwischen einem XML-Dokument und einer Transformationsdatei gibt, fragt Visual Studio nach, welche Datei verwendet werden soll.

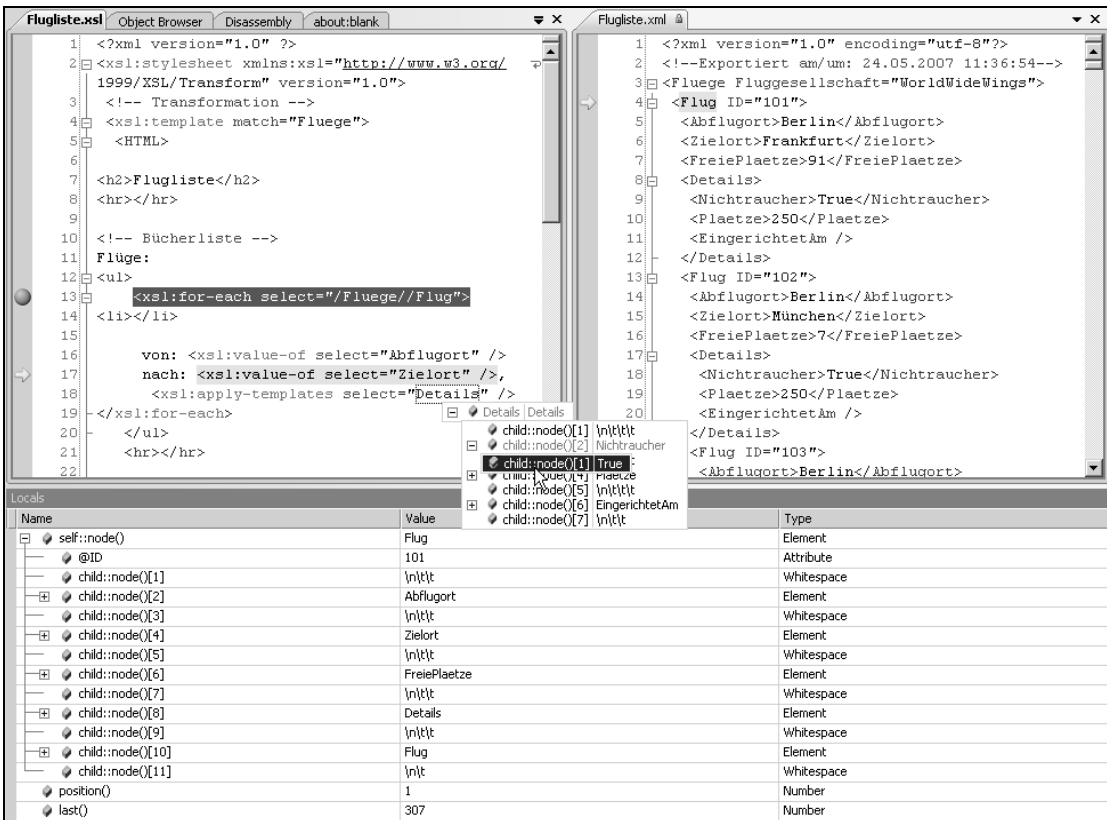


Abbildung 5.87 Arbeit mit dem XSLT-Debugger

Testen

Testfunktionen gab es bereits in Visual Studio 2005, aber nur in den Team-Editionen. In Visual Studio 2008 machte Microsoft zumindest die Basisfunktionen des Unit Testing einer breiteren Masse zugänglich, indem man diese in die Professional-Variante integrierte.

In Visual Studio 2010 sind nun die Testfunktionen in der Premium- und Ultimate-Variante erheblich erweitert, insbesondere durch Oberflächentests. Oberflächentests konnte man in Visual Studio 2005/2008 nur für Webanwendungen erstellen, wobei hier der HTTP-Datenstrom aufgezeichnet und wiedereingespielt wurde. Neu in Visual Studio 2010 ist die Aufzeichnung von Maus- und Tastaturaktionen in einer beliebigen Windows-Anwendung (*UI Tests*).

HINWEIS

Einige erweiterte Funktionen (z.B. Codeabdeckung, Lasttests, Oberflächentests) gibt es nur in den teuren Varianten (vgl. Gegenüberstellung der Varianten zu Beginn dieses Kapitels).

Testarten

Visual Studio 2010 Professional bietet folgende Testformen:

- **Unit Tests** Testen von beliebigen Klassen. Tests können aus vorhandenem Code generiert werden. Getestet werden können Klassen in folgenden Anwendungsarten: Desktop, Konsole, Compact Framework und ASP.NET.
- **Geordnete Tests** Ausführung einer bestimmten Menge von Unit Tests in einer definierbaren Reihenfolge

Höhere Varianten von Visual Studio bieten folgende Testformen zusätzlich:

- **Webtests** Testen von Webbenutzeroberflächen. Die Erstellung ist durch eine Aufzeichnung von HTTP-Anfragen mit dem Internet Explorer möglich.
- **Datenbanktests** Testen von Datenbanken durch Ausführung von SQL-Befehlen und gespeicherten Prozeduren (Stored Procedures)
- **Lasttests** Mehrfachaufrufe von Tests
- **Generische Tests** Aufruf externer Anwendungen
- **Manuelle Tests** Nicht-automatisierte Tests, bei denen ein Mensch anhand eines Test-Dokuments bestimmte Schritte vollzieht und am Ende eingibt, ob das gewünschte Ergebnis erzielt werden konnte.

HINWEIS

Microsoft nennt die Unit Tests in der deutschen Version von Visual Studio *Komponententests*. Da dieser Begriff aber eher unüblich ist, wird in diesem Buch primär der englische Begriff *Unit Tests* verwendet.

Testprojekte

Tests sind in Testprojekten gespeichert. Man erhält eine bessere Struktur mit klarer Kompetenztrennung, wenn man ein eigenes Testprojekt für jedes zu testende Visual Studio-Projekt erstellt.

Um ein Testprojekt anzulegen, gibt es in Visual Studio 2010 zwei Möglichkeiten:

- Funktion *Create Unit Test* im Code-Editor
- Anlegen eines Projekts vom Typ *Test-Projekt*

ACHTUNG Im letzteren Fall könnte sich mancher Entwickler wundern, dass Visual Studio ohne Nachfrage ein Visual Basic-basiertes Testprojekt anlegt. Die Grundeinstellung für neue Test-Projekte findet man unter *Tools/Options/Test Tools/Test Project*.

Unit Tests erstellen

Zum Erstellen von Unit Tests gibt es zwei Wege:

- Manuelle Erstellung mit einer der Unit Test-Elementvorlagen. Man hat die Wahl zwischen einem *Basic Unit Test* und einem *Unit Test*, der etwas mehr vorgefertigte Funktionen bietet.
- Erstellung mit dem Unit Test-Assistenten, der für vorhandenen Programmcode Tests erstellt. In diesem Assistenten kann man die Projekte/Klassen/Klassenmitglieder wählen, für die man Tests generieren möchte. Zu diesem Assistenten kommt man über die Elementvorlage *Unit Test Wizard* oder durch *Create Unit Test* im Code-Editor.

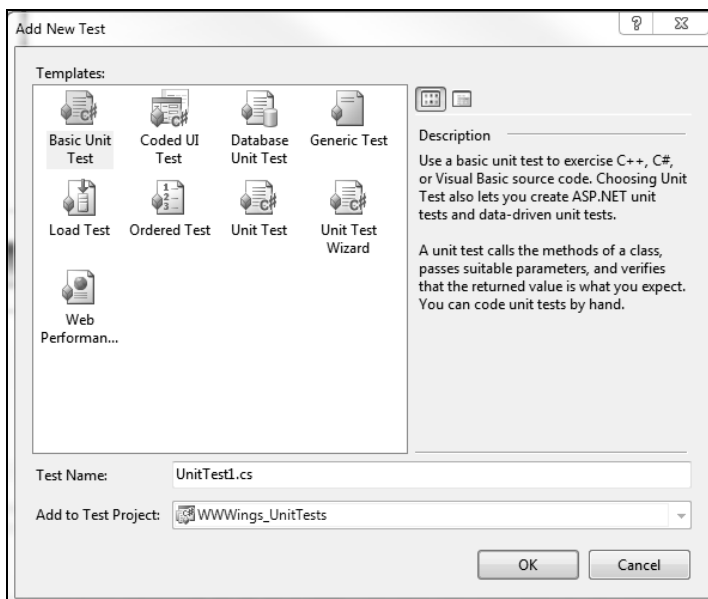


Abbildung 5.88 Anlegen eines Unit Tests

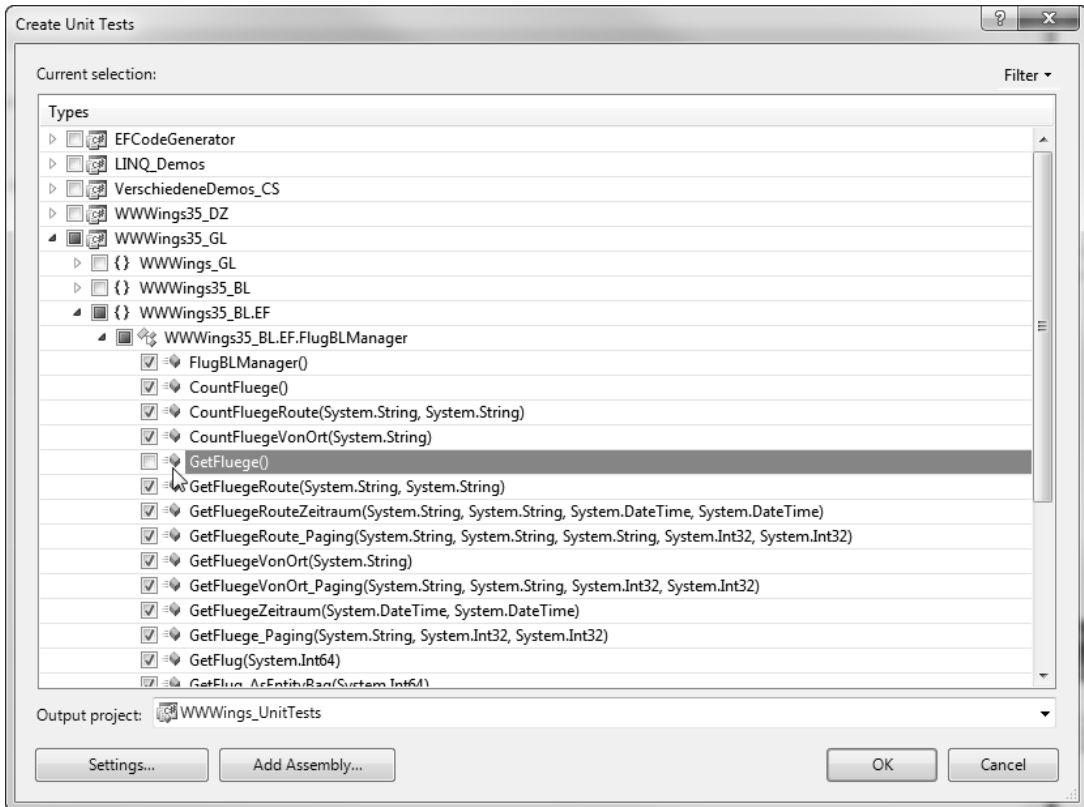


Abbildung 5.89 Assistent zur Erstellung von Unit Tests

Aufbau eines Unit Tests

Ein Unit Test ist aus der Sicht von Visual Studio eine Methode, die mit `Microsoft.VisualStudio.TestTools.UnitTesting.TestMethod` annotiert und Teil einer .NET-Klasse ist, die wiederum mit `Microsoft.VisualStudio.TestTools.UnitTesting.TestClass` annotiert ist. Durch die Klassen `Assert` und `CollectionAssert` kann der Entwickler dann Prüfbedingungen setzen.

TIPP Visual Studio unterstützt die Generierung von Rümpfen für Unit Tests durch *Add/New Test/Unit Test Wizard* oder durch die Funktion *Komponententests erstellen (Generate Unit Tests)*, die im Code-Editor im Kontextmenü zur Verfügung steht. Die generierten Tests werden aber als *nicht beweiskräftig* (`Assert.Inconclusive`) markiert und müssen manuell nachbearbeitet werden.

Visual Studio erlaubt auch das Testen von privaten Klassenmitgliedern via .NET-Reflection. Den notwendigen Programmcode lässt man sich am besten über die Testgenerierung erzeugen. Visual Studio erzeugt dann für jede Klasse mit privaten Mitgliedern eine Wrapper-Klasse mit Namen *Klasse_Accessor* über die alle Klassenmitglieder, auch die privaten, aufgerufen werden können.

```

using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using WWings35_BL;
using WWings35_BO;

namespace WWings_UnitTests
{
    /// <summary>
    /// Hier handelt es sich um eine Testklasse für FlugBLManagerTest,
    /// die alle FlugBLManagerTest-Unit Tests beinhaltet
    /// </summary>
    [TestClass()]
    public class FlugBLManagerTest
    {

        private TestContext testContextInstance;

        /// <summary>
        /// Holt (gets) oder setzt (sets) den Test-Kontext, der Informationen
        /// über und Funktionalität für den Test zur Verfügung stellt..
        /// </summary>
        public TestContext TestContext
        {
            get
            {
                return testContextInstance;
            }
            set
            {
                testContextInstance = value;
            }
        }

        /// <summary>
        /// Testet das Holen eines einzelnen Flugs
        /// </summary>
        [TestMethod()]
        public void HoleFlugTest_GueltigeFlugnummer()
        {
            FlugBLManager target = new FlugBLManager();
            Flug actual;

            long Nr1 = 101;
            actual = target.HoleFlug(Nr1);
            Assert.AreEqual(actual.FlugNr, Nr1);
            Assert.AreEqual(actual.Abflugort, "Berlin");
            Assert.AreEqual(actual.Zielort, "Frankfurt");
            Assert.AreEqual(actual.Plätze, (short)250);
        }

        /// <summary>
        /// Testet das Holen eines einzelnen Flugs
        /// </summary>
        [TestMethod()]
        [ExpectedException(typeof(System.InvalidOperationException))]
        public void HoleFlugTest_UngueltigeFlugnummerErzeugtFehler()
        {
            FlugBLManager target = new FlugBLManager();
            Flug actual;

```

```
long Nr2 = 99; // ungültige Nummer!
actual = target.HoleFlug(Nr2);
Assert.IsNull(actual);
}

/// <summary>
/// A test for HoleAlle
/// </summary>
[TestMethod()]
public void HoleAlleTest1()
{
    List<Flug> actual;
    actual = new FlugBLManager().HoleAlle();
    Assert.IsTrue(actual.Count > 200);
    CollectionAssert.AllItemsAreUnique(actual);
    CollectionAssert.AllItemsAreNotNull(actual);
    CollectionAssert.AllItemsAreInstancesOfType(actual, typeof(Flug));
}

/// <summary>
/// A test for HoleAlle
/// </summary>
[TestMethod()]
public void HoleAlleTest2()
{
    FlugBLManager target = new FlugBLManager();
    string Ort = "Rom";
    List<Flug> actual;
    actual = target.HoleAlle(Ort);
    Assert.IsTrue(actual.Count > 10);
}
}
```

Listing 5.2 Beispiele für einen Unit Test

WICHTIG Vergessen Sie nicht, dass auch das Unit Test-Projekt alle Verbindungszeichenfolgen und andere Konfigurationseinstellungen braucht, die der zu testende Programmcode verwendet. Kopieren Sie die Konfigurationsdatei des zu testenden Projekts in das Testprojekt!

Weitere Möglichkeiten für Unit Tests

Visual Studio Unit Testing unterstützt weitere Annotationen:

- **[TestInitialize]** Dieser Code wird vor jeder Testmethode ausgeführt
- **[TestCleanup]** Dieser Code wird nach jeder Testmethode ausgeführt
- **[ClassInitialize]** Dieser Code wird vor der ersten Testmethode in dieser Klasse ausgeführt
- **[ClassCleanup]** Dieser Code wird nach der letzten Testmethode in dieser Klasse ausgeführt

Das folgende Listing zeigt den Einsatz von `[TestInitialize()]`. Hierbei wird vor jedem der Tests ein Flugobjekt erzeugt. In den einzelnen Tests wird dann jeweils nur einmal *Assert* aufgerufen. Das Problem bei *Assert* ist, dass ein Test bei der ersten Bedingung mit der Testmethode aufhört. Dadurch kann man auf den ersten Blick das Ausmaß des Problems oft gar nicht erkennen, wenn ein Test fehlschlägt. Das nachstehende Listing verfolgt daher den Ansatz *Single Assert Per Unit Test* (siehe [OSH01]).

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using WWWings35_BL;
using WWWings35_BO;

namespace WWWings_UnitTests
{
    /// <summary>
    /// Summary description for SelbstErstellterTest
    /// </summary>
    [TestClass]
    public class SelbstErstellterTest
    {

        private TestContext testContextInstance;

        /// <summary>
        /// Gets or sets the test context which provides
        /// information about and functionality for the current test run.
        /// </summary>
        public TestContext TestContext
        {
            get
            {
                return testContextInstance;
            }
            set
            {
                testContextInstance = value;
            }
        }

        FlugBLManager BL;
        Flug f ;

        [TestInitialize()]
        public void MyTestInitialize()
        {
            BL = new FlugBLManager();
            f = BL.HoleFlug(101);
        }

        [TestMethod]
        public void Teste_FlugNr()
        {
            Assert.AreEqual(f.FlugNr, 101);
        }
    }
}
```

```
[TestMethod]
public void Teste_AbflugOrt()
{
    Assert.AreEqual(f.Abflugort, "Berlin");
}

[TestMethod]
public void Teste_Zielort()
{
    Assert.AreEqual(f.Zielort, "Frankfurt");
}

[TestMethod]
public void MeinZweiterTest()
{
    Assert.AreEqual(f.FlugNr, 101);
    Assert.AreEqual(f.Abflugort, "Berlin");
    Assert.AreEqual(f.Zielort, "Frankfurt");
}
}
```

Listing 5.3 Beispiele für Unit Tests mit [TestInitialize()]

Über das Mitglied `TestContext`, das bei allen Unit Tests vorhanden ist, die nicht mit der Vorlage *Basic Unit Test* erstellt wurden, kann man auf die Eigenschaften des Tests (z. B. Name, Verzeichnis) zugreifen.

```
[TestMethod]
public void SpielereienMitTestContext()
{
    System.Diagnostics.Debug.WriteLine(this.TestContext.TestName);
    System.Diagnostics.Debug.WriteLine(this.TestContext.TestDir);
    System.Diagnostics.Debug.WriteLine(this.TestContext.TestDeploymentDir);
    System.Diagnostics.Debug.WriteLine(this.TestContext.CurrentTestOutcome);
    System.Diagnostics.Debug.WriteLine(this.TestContext.TestName);
}
```

Listing 5.4 Verwendung des `TestContext`-Objekts

Testausführung

Die Testlaufkonfiguration (Test Run Configuration, *.testrunconfig*) entscheidet über den genauen Ablauf der Unit Tests (z. B. kann dieser sich auf Programmcode in ASP.NET-Webprojekten beziehen und dann müssen diese ebenfalls im Webserver laufen). Tests kann man auch an der Kommandozeile ausführen mit *MSTest.exe*.

Die zu startenden Tests legt man im Fenster *Test View* oder *Test List Editor* fest. *Test View* zeigt alle Tests in einer flachen Liste an. *Test List Editor* erlaubt die Speicherung beliebiger benutzerdefinierter Testlisten in einem hierarchischen Baum.

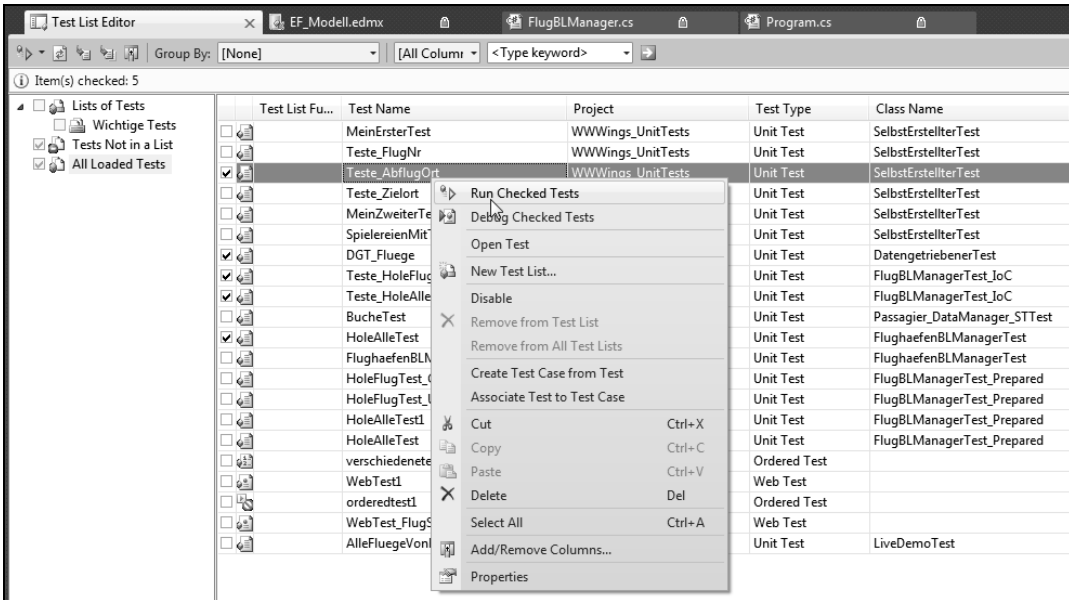


Abbildung 5.90 Zusammenstellen von Unit Tests zu Testlisten im Test List Editor

Bei jeder Testausführung erstellt Visual Studio einen Ordner und kopiert alle an dem Test beteiligten Assemblys dort hinein. Der Ordner liegt meist unter `/TestResults/BenutzerName_Rechner_Datum_Zeit`. Er enthält dann auch die Testergebnisse in `.vsmdi`-Dateien. Im Standard speichert Visual Studio die letzten 25 Testergebnisse und warnt dann vor dem Überschreiben der Ergebnisse. Die Anzahl kann aber verändert werden.

Datengetriebene Tests

Ein datengetriebener Unit Test ermöglicht die automatische Wiederholung eines Unit Tests für jeden Datensatz einer Datenmenge, wobei der jeweils aktuelle Datensatz als Eingabeparameter für den Test behandelt wird. Ein datengetriebener Test wird durch die Annotation `[DataSource]` (zusätzlich zu `[TestMethod]`) ausgezeichnet. Innerhalb der Testmethode kann man auf die Daten über `TestContext.DataRow` zugreifen.

```
[TestMethod]
[DataSource("System.Data.SqlClient",
@"Data Source=.\SQLEXPRESS;AttachDbFilename=h:\www\Datenbanken\WorldWideWings.mdf;
Integrated Security=True;User Instance=true", "FL_Fluege", DataAccessMethod.Sequential)]
public void DGT_Fluege()
{
    Console.WriteLine("FlugNr: {0}, Abflugort: {1}, Zielort: {2}",
```



```

TestContext.DataRow["FL_FlugNr"], TestContext.DataRow["FL_AbflugOrt"],
TestContext.DataRow["FL_ZielOrt"]);

System.Data.DataRow dr =
de.WWings.DAL.Flug_DataManager.AllFluegeRoute(TestContext.DataRow["FL_AbflugOrt"].ToString(),
TestContext.DataRow["FL_ZielOrt"].ToString()).Tables[0].Rows[0];

Assert.Equals(TestContext.DataRow["FL_FlugNr"], dr["FL_FlugNr"]);
}

```

Listing 5.5 Datengetriebener Test für die Datenzugriffsschicht von World Wide Wings

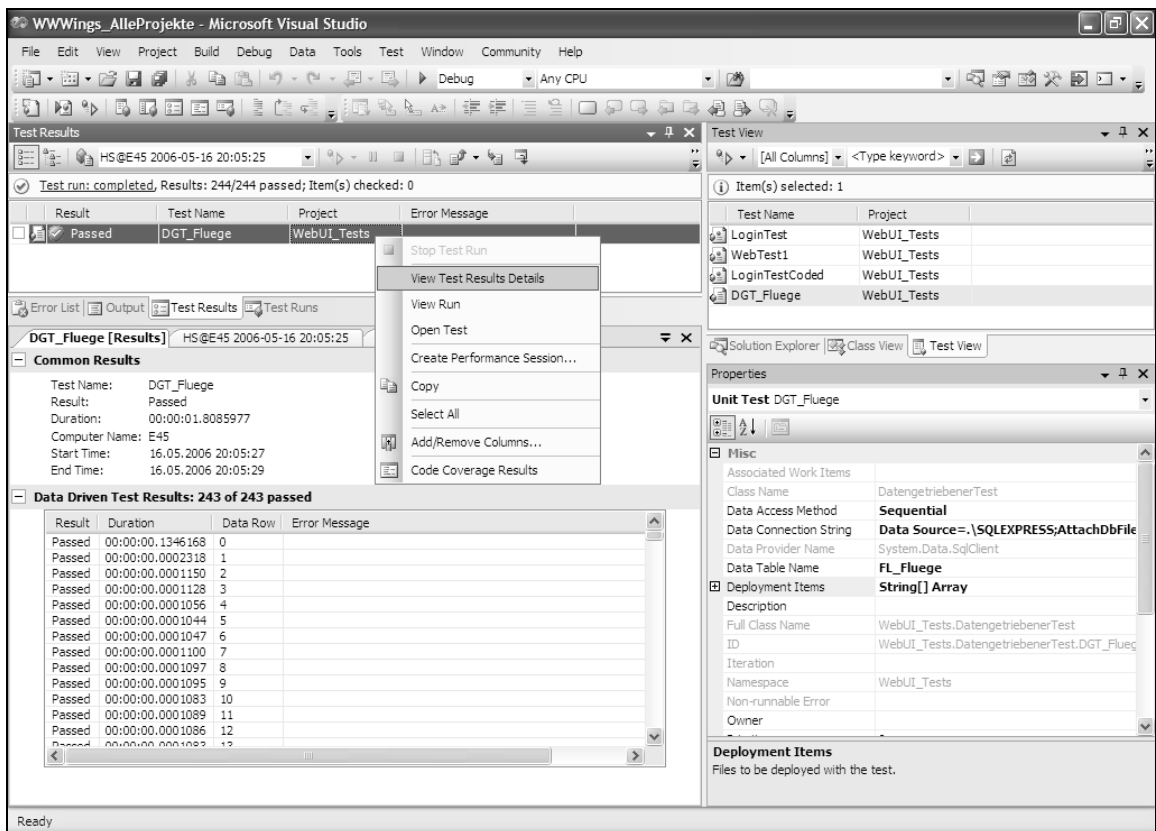


Abbildung 5.91 Anzeige der Ergebnisse eines datengetriebenen Tests mit automatischer Wiederholung des Tests für jeden Datensatz einer Quelldatenmenge

Testabdeckung (Code Coverage)

Anders als das beliebte Freeware-Werkzeug *NUnit* zeigt die in Visual Studio integrierte Unit-Testing-Funktion auch, welche Teile des Programmcodes durch den Test abgedeckt und welche Teile nicht ausgeführt wurden (Code Coverage).

Code Coverage ist der Grad der Abdeckung des zu testenden Programmcodes im Rahmen von Unit Tests. Code Coverage muss vor dem Start eines Tests in der Testkonfiguration aktiviert werden. Nach Ablauf des Tests stehen Daten über die Testabdeckung zur Verfügung (siehe Abbildung 5.93).

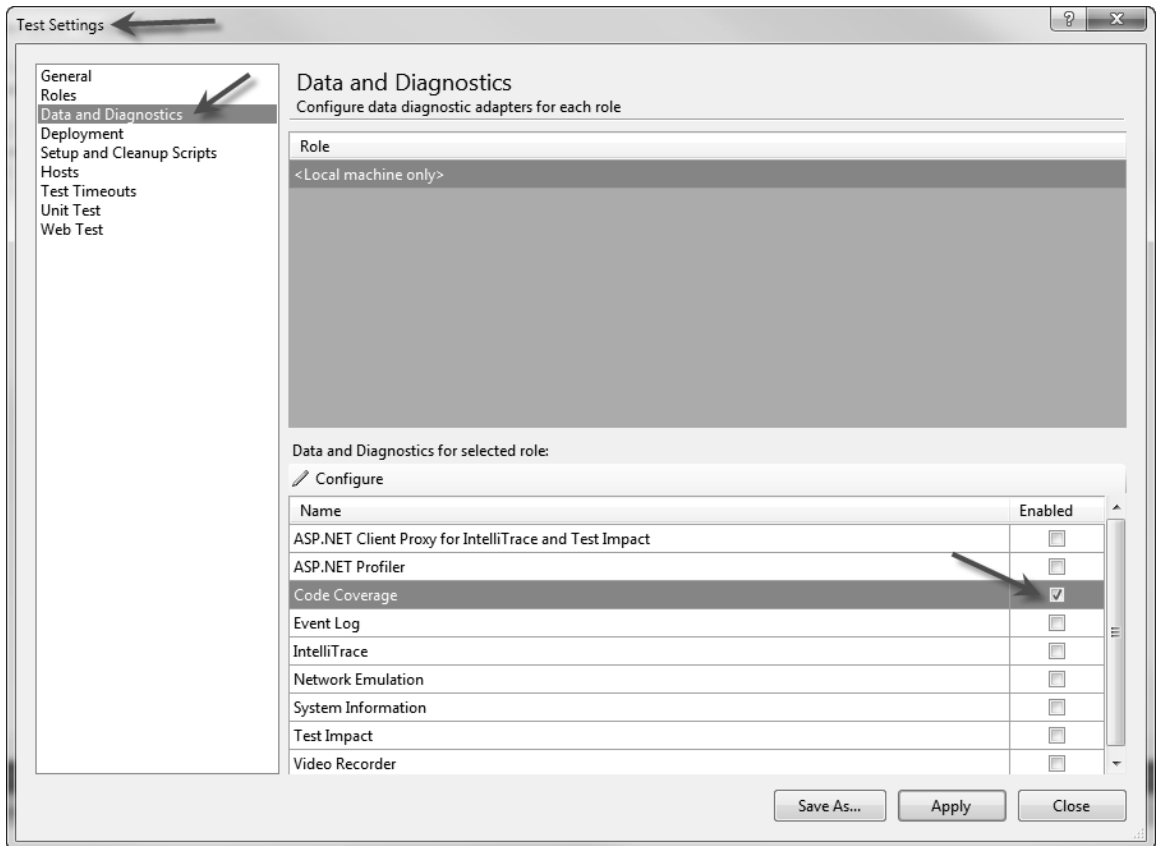


Abbildung 5.92 Aktivierung der Code Coverage-Funktion in der Konfiguration eines Testdurchlaufs

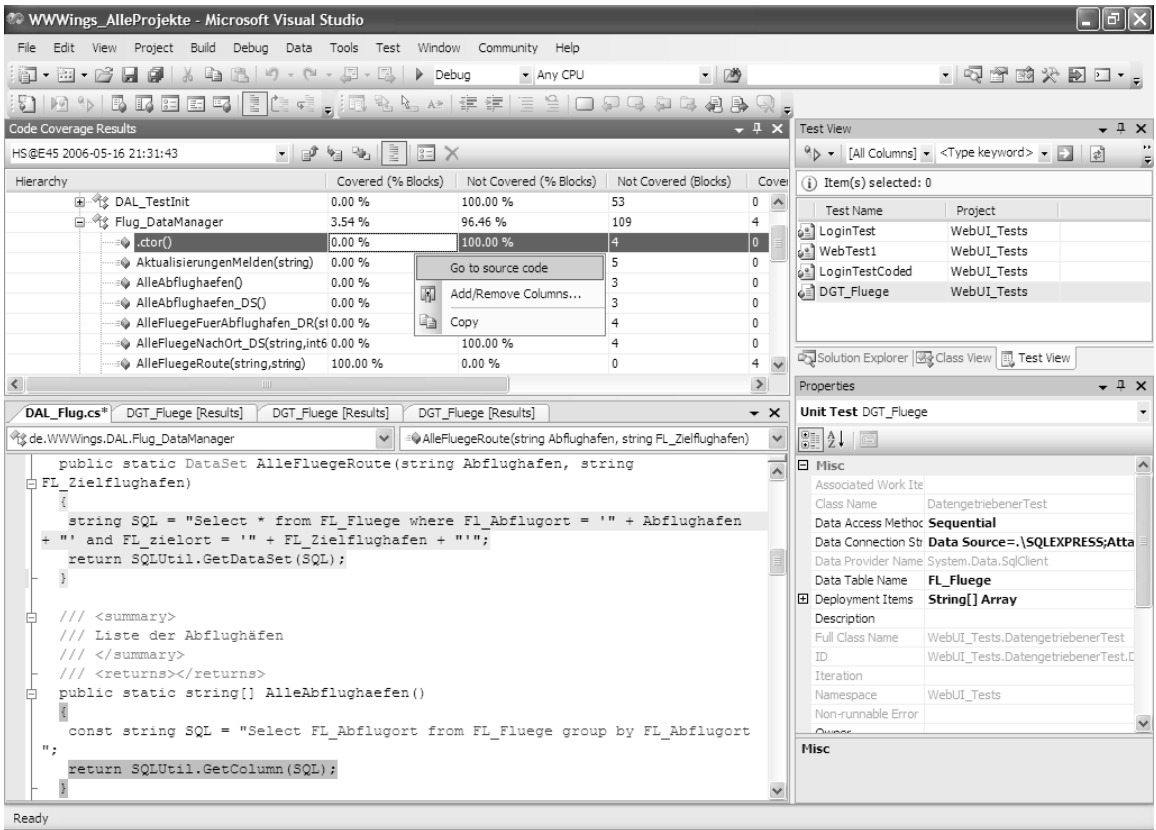


Abbildung 5.93 Anzeige der Testabdeckung durch Prozentzahlen und farbige Hinterlegung des Quellcodes (Code Coverage)

Webtests

Webtests sind Unit Tests für Webbenutzeroberflächen. Bei dem in Webtests verwendeten Verfahren ist die auf dem Webserver verwendete Programmier Technik irrelevant, d. h., es können auch Websites getestet werden, die nicht mit ASP.NET erstellt wurden. Visual Studio steuert bei den Tests nicht den Browser, sondern spielt vorher aufgezeichnete HTTP-Abfragen wieder ein. Dabei werden gezielte Ersetzungen (z. B. für Sitzungs-IDs) vorgenommen.

Beim Anlegen eines Webtests bietet Visual Studio die Möglichkeit, den Test mit dem Internet Explorer aufzuzeichnen. Hierzu speichert Visual Studio den HTTP-Datenstrom zwischen Client und Server ab. Die Aufzeichnung kann später wieder abgespielt werden. Dabei berücksichtigt der Webtest, dass sich bei der Abspielung einige Werte (z. B. Sitzungsnummer) geändert haben können. Die Bedenkzeiten des Nutzers bei der Aufzeichnung können verwendet oder ignoriert werden.

ACHTUNG Durch diese Form des Tests kann man JavaScript-Code und ActiveX-Steuerelemente im Browser nur dann »simulieren«, wenn diese eine Interaktion (d. h. Datenaustausch) mit dem Webserver haben.

Der Entwickler muss dem Ablauf von Anfragen und Antworten anschließend manuell Regeln (*Validation Rules*) hinzufügen, mit denen der Webtest automatisch prüfen kann, ob das gewünschte Ergebnis vom Webserver geliefert wurde.

Die Aufzeichnung wird in XML-Form abgelegt. Der Entwickler kann durch die Funktion *Code generieren* (*Generate Code*) daraus Programmcode generieren, den er anpassen kann. Dabei ist zu beachten, dass zwei voneinander unabhängige Tests entstehen.

Webtests können in zwei Formen gespeichert werden:

- als XML-Dateien (.webtest) mit dem Wurzelement <TestCase> oder
- als Programmcode in einer von `Microsoft.VisualStudio.TestTools.WebTesting.WebTest` abgeleiteten Klasse (*Coded Webtest*)

Eine XML-Webtest-Datei kann man in eine Codedatei umwandeln (Symbol *Generate Code*), aber nicht umgekehrt.

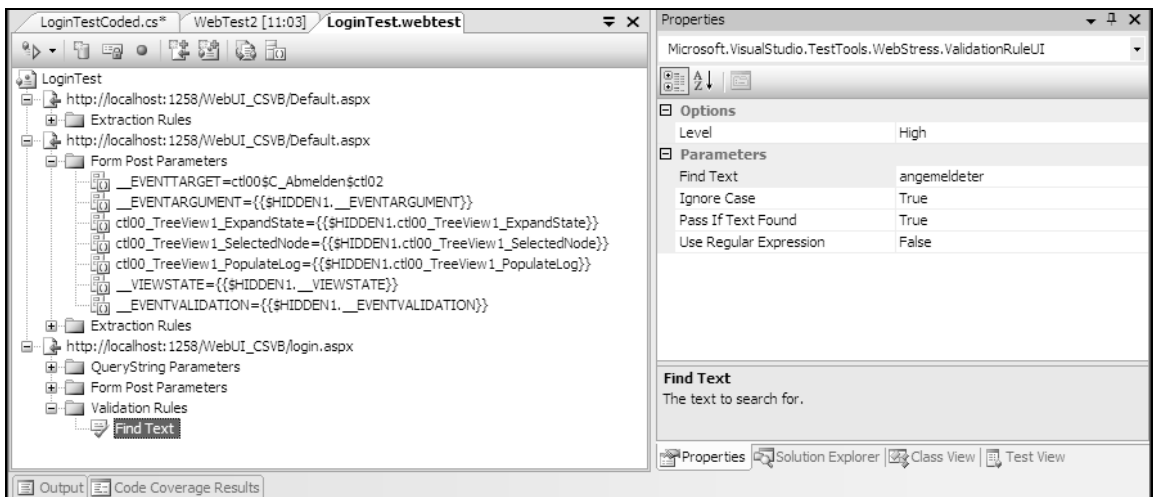


Abbildung 5.94 Ein aufgezeichneter Webtest mit einer Validation Rule, die prüft, ob ein Text in der Ausgabe enthalten ist

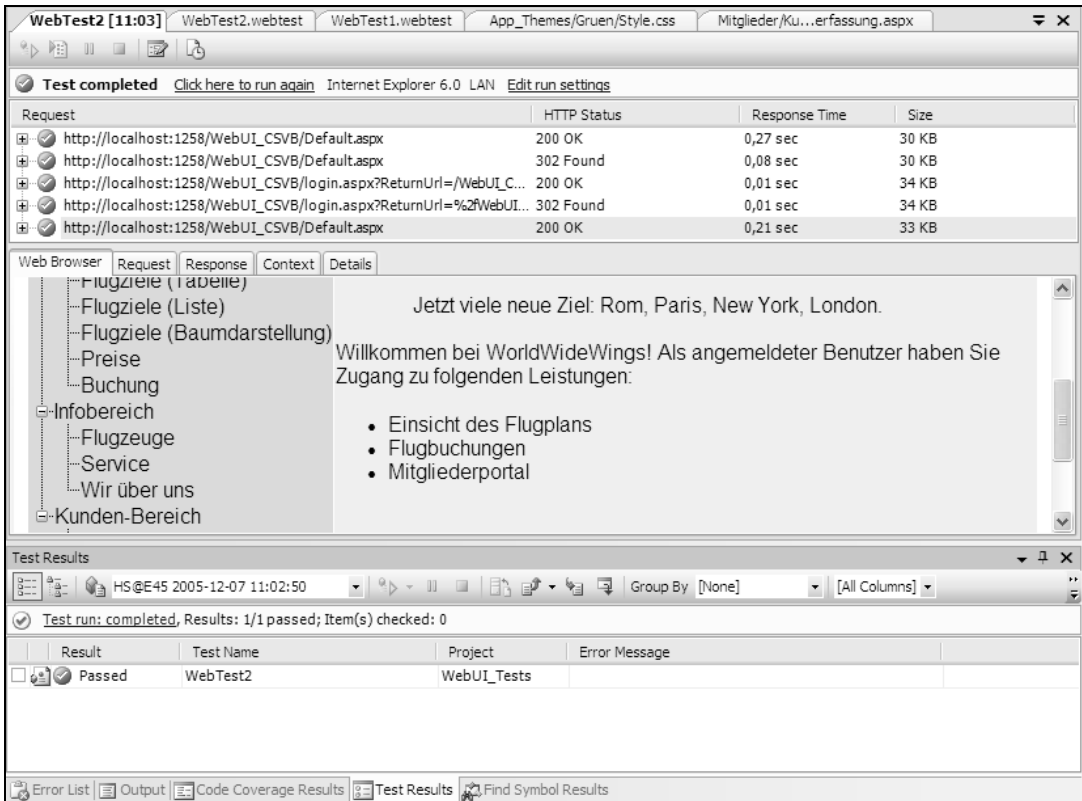


Abbildung 5.95 Ein Webtest wurde erfolgreich ausgeführt

Testwirkungsanalyse (Test Impact Analysis)

Test Impact Analysis ist eine Funktion in Visual Studio 2010 Premium und Ultimate, die den Entwickler darüber informiert, welche Tests er nach einer Codeänderung unbedingt ausführen sollte.

Aktivierung der Test Impact Analysis

Das Fenster *Test Impact View* aktiviert man im Menü *Test/Windows/Test Impact View*. Beim ersten Aufruf fordert das Fenster zunächst auf, durch Klick auf einen Link die Test Impact Analysis überhaupt zu aktivieren. Der Klick auf den Link ist gleichbedeutend mit der in der folgenden Bildschirmabbildung gezeigten Einstellung.

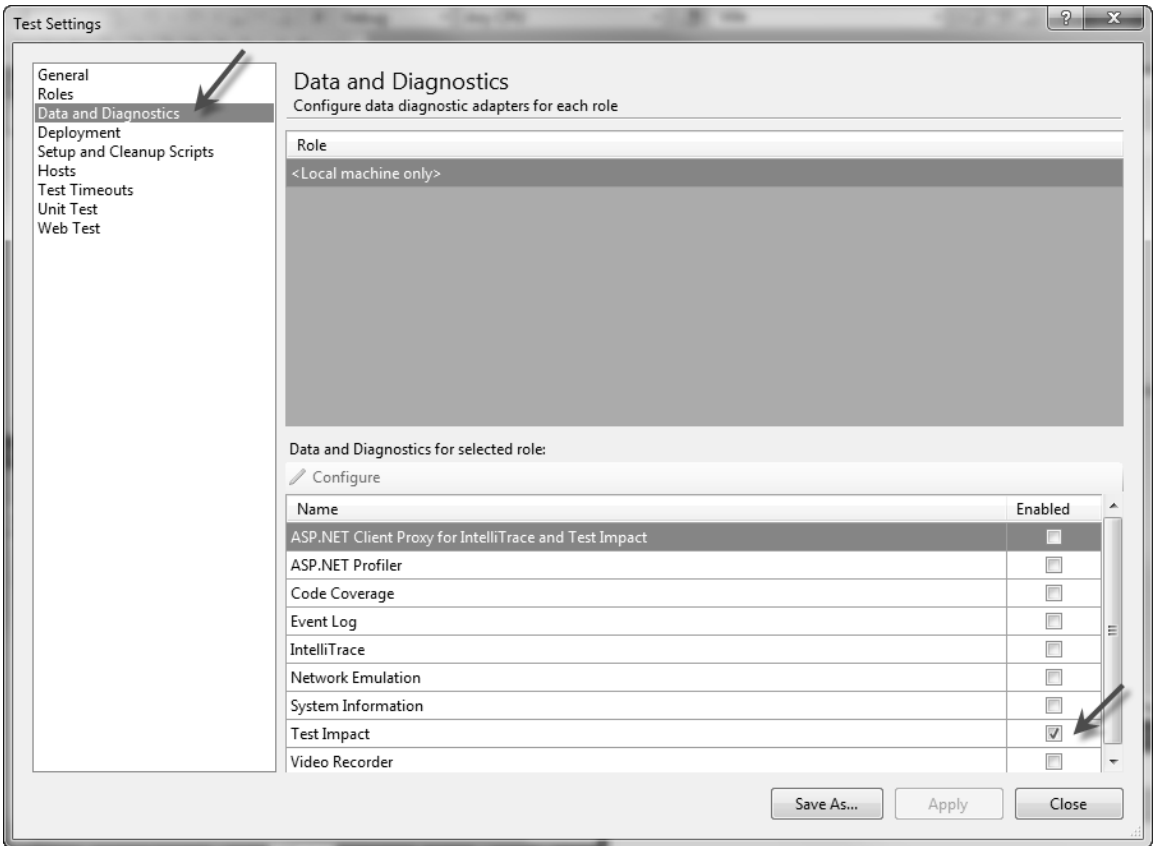


Abbildung 5.96 Testeinstellungen (Test/Edit Test Settings/Local)

Aufzeichnung von Daten

Die Erkenntnis, welcher Code von welchen Tests aufgerufen wird, gewinnt Visual Studio nicht durch statische Codeanalyse, sondern durch Überwachung der Tests beim Ablauf. Daher muss man alle Tests einmal gestartet haben, bevor die Test Impact Analysis wirken kann.

Anzeige der Analyse

Nach erfolgreichen Kompilierungsvorgängen zeigt das Fenster *Test Impact View* dann an, welche Tests durch die Änderungen betroffen sind und bietet die Neuausführung dieser Tests an. Dabei werden nicht nur Änderungen an dem zu testenden Programmcode, sondern auch an den Tests selbst berücksichtigt. Man kann die betroffenen Tests dann direkt aus dem Fenster *Test Impact View* starten (siehe Symbolleiste).

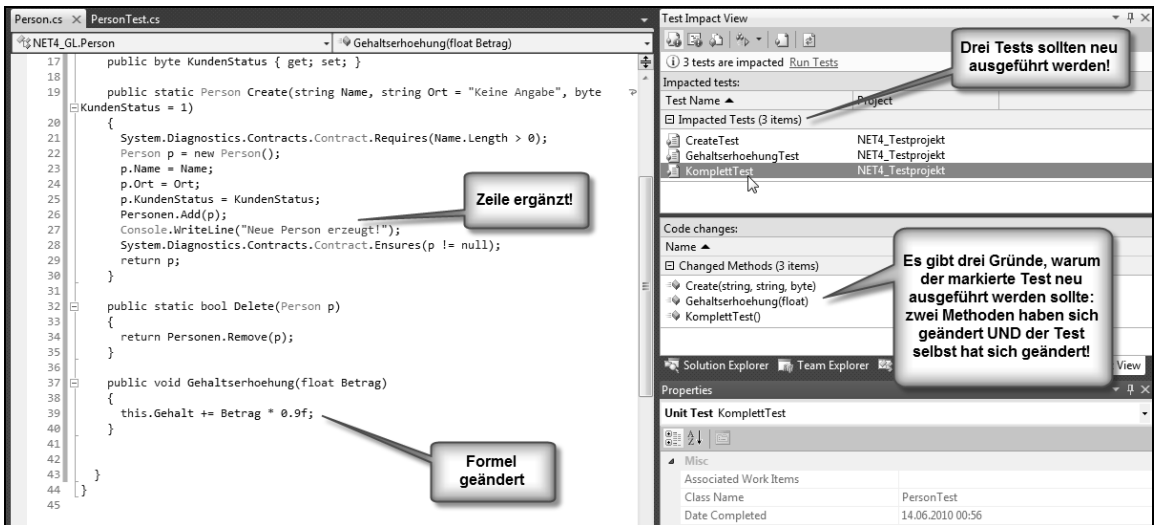


Abbildung 5.97 Ergebnis der Testwirkungsanalyse an einem Beispiel

TIPP Bei eingeschalteter Testwirkungsanalyse kann man auf einem Methodennamen *Show Calling Tests* oder *Run Calling Tests* ausführen. Leider funktioniert dies nicht für Klassen und Attribute (Fields oder Properties). Es passiert dann nichts; es gibt auch keine Fehlermeldung.

Oberflächentests (UI Tests)

Mit Visual Studio 2010 (ab Premium-Variante) unterstützt Microsoft nun auch das Aufzeichnen und Wiederabspielen von Oberflächentests (*UI-Tests*). Der Entwickler oder ein dedizierter Tester wird dabei von einem Rekorder mit Namen *Coded UI Test Builder* unterstützt, der automatisch startet, wenn man einen neuen Coded UI Test anlegt.

Zu Beginn der Aufzeichnung muss man die zu testende Anwendung starten. Dann nimmt man beliebige Aktionen in der Anwendung vor. Der Coded UI Test Builder zeigt diese auf Wunsch in einem getrennten Fenster an. Zur Überprüfung, ob die zu testende Benutzeroberfläche so reagiert, wie es erwartet wird, legt man in so genannten *Assertions* fest, welche Werte man in bestimmten Steuerelementeigenschaften erwartet. An beliebiger Stelle kann man dann die Codegenerierung anstoßen. Visual Studio fordert zur Vergabe eines Methodennamens auf. Ein Methodenne ist kein unabhängiger Test, sondern nur Teil eines Tests.

HINWEIS Leider kann man innerhalb einer Methode keine Aktionen und Assertions mischen. Man muss die Aktion(en) aufzeichnen, dann Code erzeugen, dann die Assertions aufzeichnen, dann wieder Code erzeugen, usw. Ein mit dem Coded UI Test Builder generierter Programmcode sieht typischerweise so aus, dass es mehrere Methoden gibt, bei denen sich Aktionen und Prüfungen in einzelnen Methoden abwechseln.

```
public class UITest_ProzentRechner
{
...
    [TestMethod]
    public void UITest_ProzentRechner1 ()
```

```

{
    // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu
    and select one of the menu items.
    // For more information on generated code, see http://go.microsoft.com/fwlink/?LinkId=179463
    this.UIMap.ZahlEingebenUndSliderSetzen();
    this.UIMap.SliderAusgabePruefen();
    this.UIMap.ButtonDruecken();
    this.UIMap.ErgebnisPruefen();
}
}

```

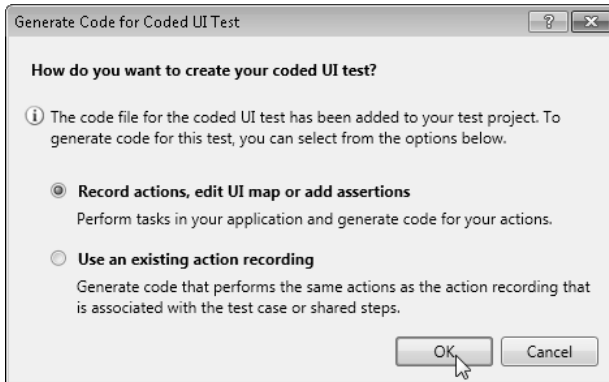


Abbildung 5.98 Entscheidung nach dem Anlegen eines Code UI Tests

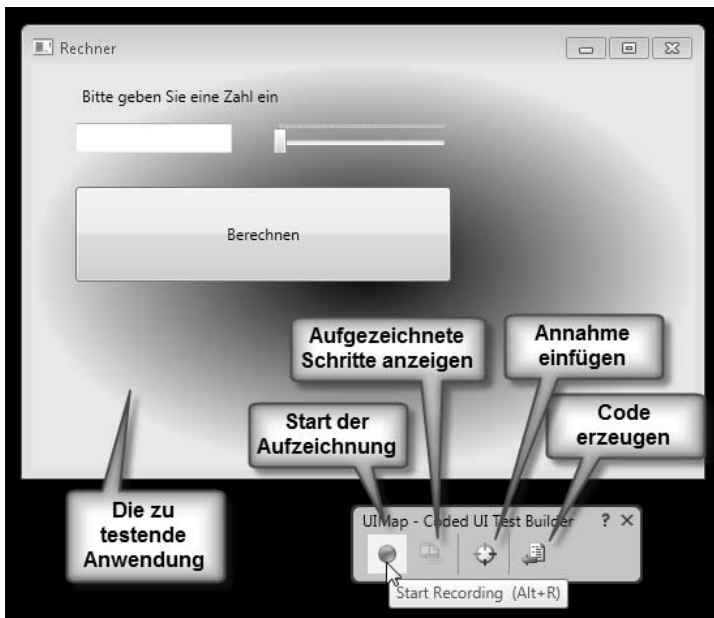


Abbildung 5.99 Aufzeichnung mit dem Testrekorder

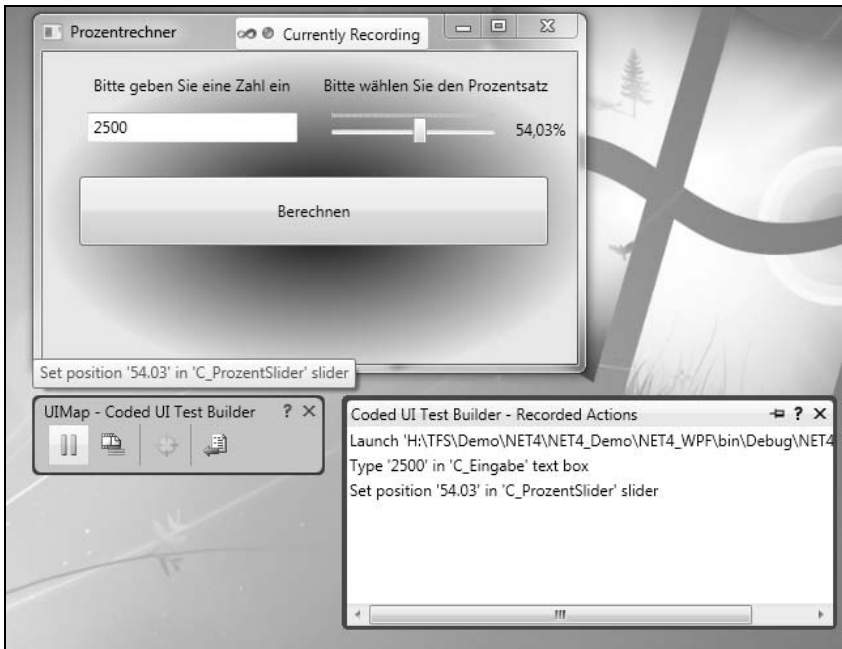


Abbildung 5.100 Schritte während der Aufzeichnung

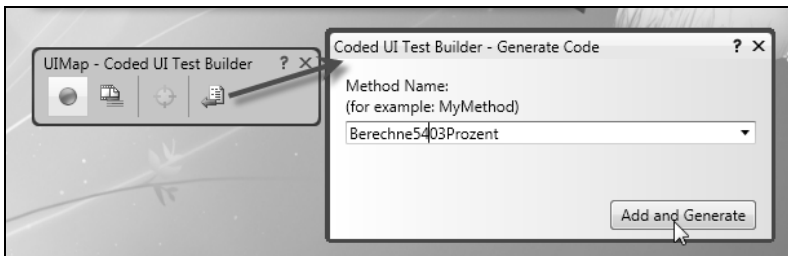


Abbildung 5.101 Erzeugung von Programmcode aus den aufgezeichneten Schritten

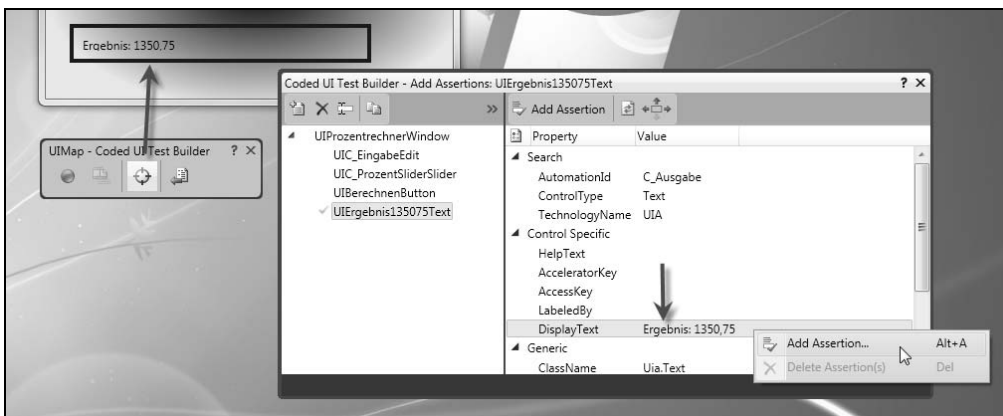


Abbildung 5.102 Festlegung von Bedingungen

Datenbanktests

Bei Datenbanktests legt man fest:

- Die Datenbank, die getestet werden soll
- Den auszuführenden Befehl (SQL oder gespeicherte Prozedur)
- Das erwartete Ergebnis

HINWEIS Datenbanktests sind nur in den Varianten Premium und Ultimate verfügbar.

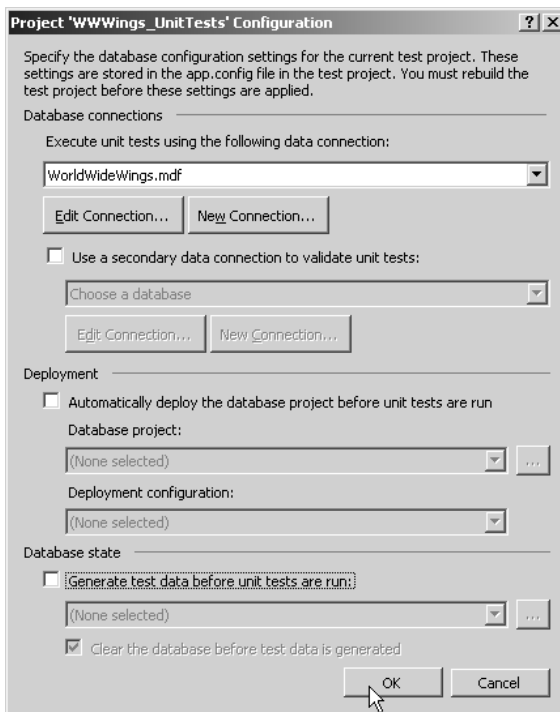


Abbildung 5.103 Festlegung der Datenbank

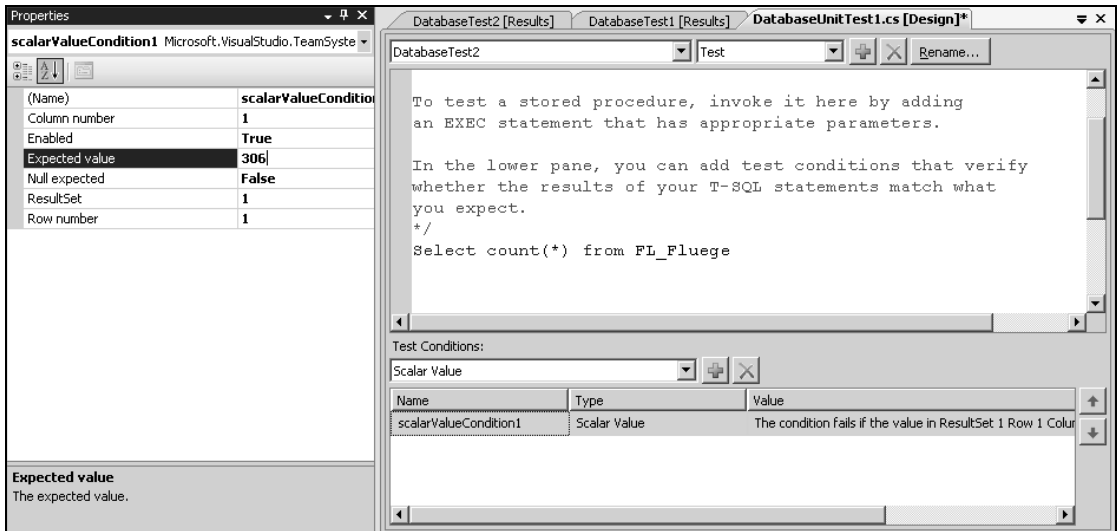


Abbildung 5.104 Festlegung von Befehl und erwartetem Ergebnis

Leistungsmessung

Visual Studio Premium und Ultimate erlauben die Messung von Geschwindigkeit und Speicherverbrauch von .NET-Anwendungen. Das Fenster *Leistungs-Explorer* (*Performance Explorer*) erreicht man über *Ansicht/Weitere Fenster/Leistungs-Explorer* (*View/Other Windows/Performance Explorer*) oder über *Extras/Leistungstools* (*Tools/Performance Tools*).

Eine so genannte *Performance Session*, die man im Performance Explorer manuell oder per Assistent anlegen kann, misst die Geschwindigkeit und wahlweise zusätzlich auch den Speicherbedarf. Die Messung des Speicherbedarfs muss man in den Eigenschaften der Performance Session aktivieren (Registerkarte *Allgemein*, siehe Abbildung).

TIPP

Die Einstellung des Leistungs-Explorers kann man in Form einer .psess-Datei abspeichern.

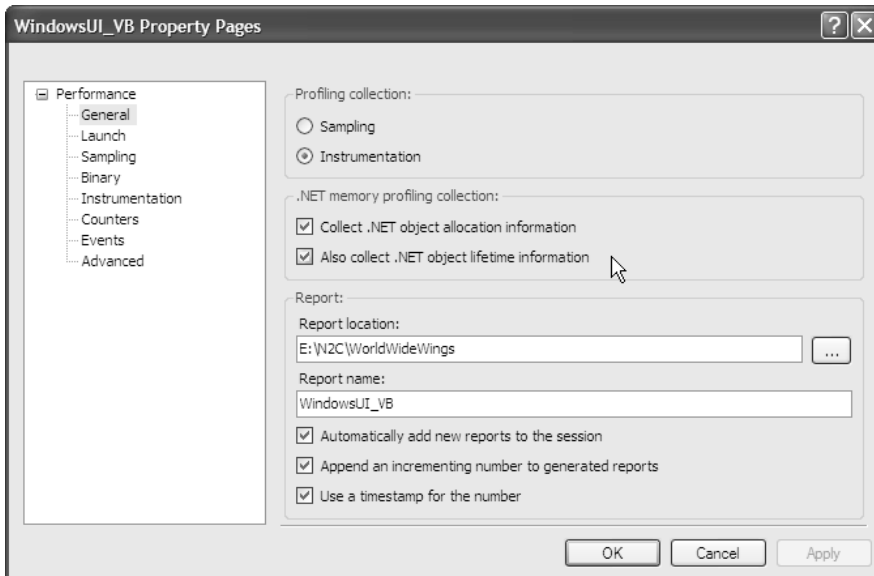


Abbildung 5.105 Aktivierung der Speichermessung

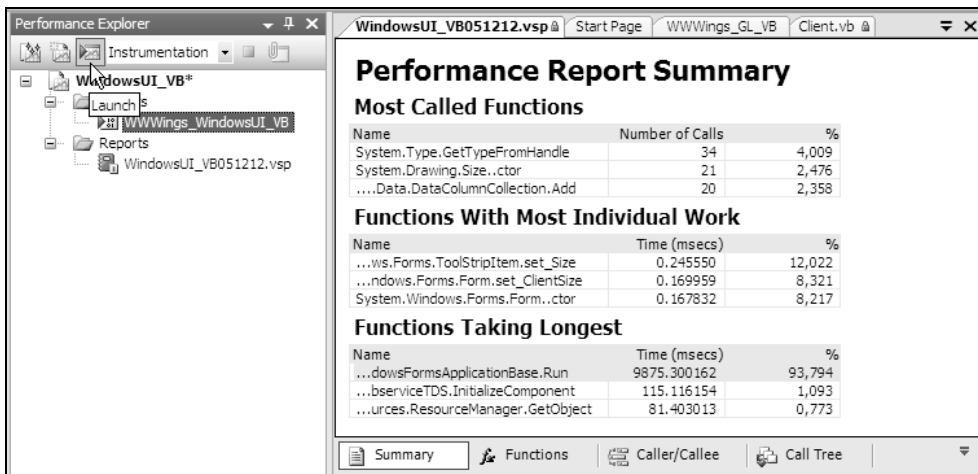


Abbildung 5.106 Bericht über die Geschwindigkeit einer Anwendung

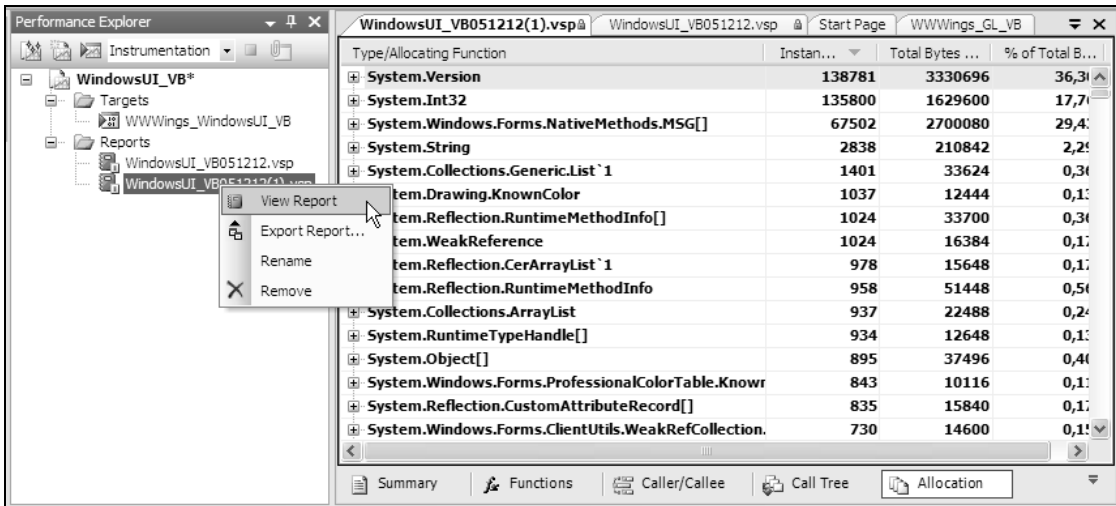


Abbildung 5.107 Bericht über den Speicherverbrauch einer Anwendung

Test Suite Manager

Mit dem *Microsoft Test and Lab Manager* liefern die Redmonder als Beiwerk zu Visual Studio 2010 nun ein eigenes Werkzeug zum Ausführen und Verwalten von Tests mit, damit Tester nicht mehr das für ihre Bedürfnisse zu unübersichtliche Visual Studio benötigen.

Im Test Suite Manager kann man alle Testarten und die jeweiligen Fälle sowie Ergebnisse verwalten und auswerten. Die Basis bildet der Team Foundation Server (TFS). Insbesondere für manuelle Tests gibt es eine umfangreiche Unterstützung. Mit dem Manual Test Runner können manuelle Tests anhand eines Testplans durchgeführt werden. Der Test wird dabei aufgezeichnet. Wird ein Fehler gefunden, kann der Tester zum Bug auch ein Video liefern. Aus aufgezeichneten manuellen Tests können automatisierte Tests gemacht werden (CodedUI Test).

Der *Test Suite Manager* kann hier leider aus Platzgründen nicht näher beschrieben werden.

Verbreiten von Anwendungen (Deployment)

Im Kapitel 4 »Grundkonzepte des .NET Framework 4.0« im Abschnitt »Verbreitung und Installation von .NET-Anwendungen« wurden die Grundprinzipien der Anwendungsverbreitung in .NET beschrieben. Hier geht es um die Werkzeuge, mit denen Visual Studio die Anwendungsverbreitung unterstützt.

Erstellen von MSI-Paketen

Visual Studio 2010 bietet zwei grundsätzliche Wege zur Erstellung von Installationspaketen:

- Wie in der Vorgängerversion kann man MSI-Pakete mit dem Visual Studio-Installer erstellen
- Alternativ kann man MSI-Pakete mit einer reduzierten Version von dem Drittanbieterprodukt InstallShield (diese wird mit Visual Studio 2010 mitgeliefert ohne Aufpreis) erstellen

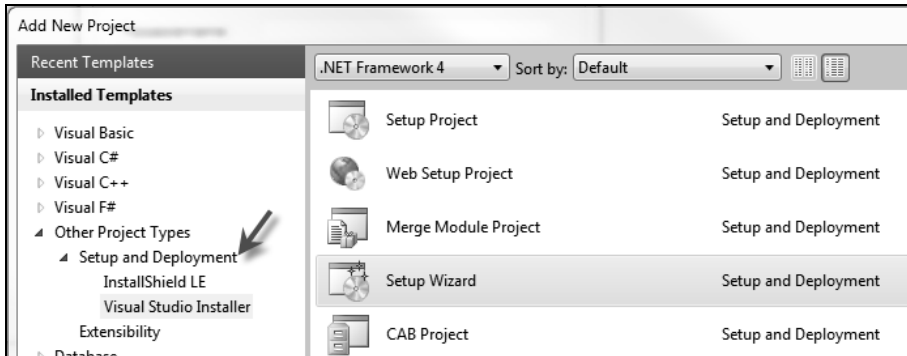


Abbildung 5.108 Projektvorlagen für Installationspakete in Visual Studio 2010

Es folgen Hinweise zum Erstellen eines Setup-Projekts mit der Vorlage *Setup Project*.

Man fügt in einer Projektmappe, in der sich auch die zu installierende Anwendung befindet, ein neues *Setup Project* hinzu. Nach dem Anlegen des Projekts sieht man ein Ordnerfenster mit der Überschrift *File System On Target Machine*. Im Projektmappenexplorer zeigt sich das Setup-Projekt nur mit einem Unterordner *Detected Dependencies*. Hier kann man über *View* auch andere Fenster einblenden, z.B. eine Sicht auf die Registry oder die Benutzerschnittstelle der Installationsroutine.

Als ist festzulegen, welches Projekt aus der Projektmappe die zu installierende Anwendung darstellen soll. Dazu wählt man auf dem Setup-Projekt im Kontextmenü *Add/Project Output* und dann *Primary Output*. Im Fall, dass man eine .NET-Anwendung mit mehreren referenzierten eigenen Assemblys wählt, könnte die Anzeige dann wie rechts in der folgenden Bildschirmabbildung aussehen.

Um den linken Teil in der *File System*-Ansicht zu definieren, sind folgende Schritte notwendig:

1. Auf *Application Folder* Funktion *Add/Project Output* und dann *Primary Output* der zu installierenden Anwendung.
2. *Add/Folder* auf *User's Programs Menu*.
3. Auf diesem Ordner dann *Create New Shortcut* und *Primary Output* aus dem Ordner, der in Schritt 2 angelegt wurde.
4. Man sollte diese Verknüpfung anschließend gut benennen, denn so erscheint die Anwendung im Startmenü.

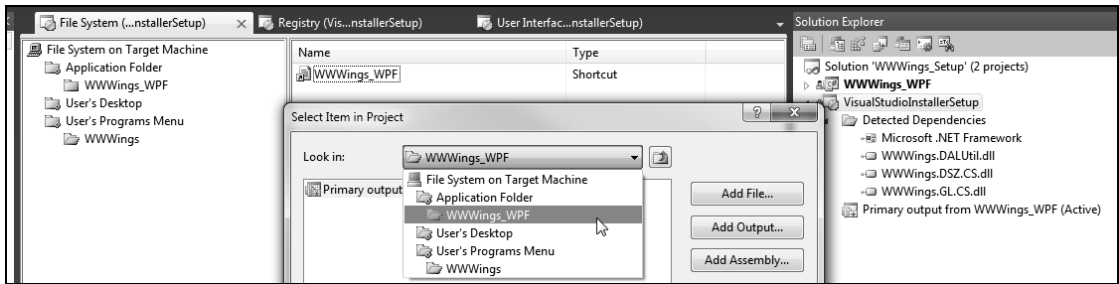


Abbildung 5.109 Erstellen einer Verknüpfung im Startmenü

Optional kann man noch Einstellungen für die Registry vornehmen (siehe Bildschirmabbildung). Hier kann man in den Eigenschaften zum Beispiel festlegen, ob die Schlüssel beim Deinstallieren entfernt werden sollen (*DeleteAtUninstall*).

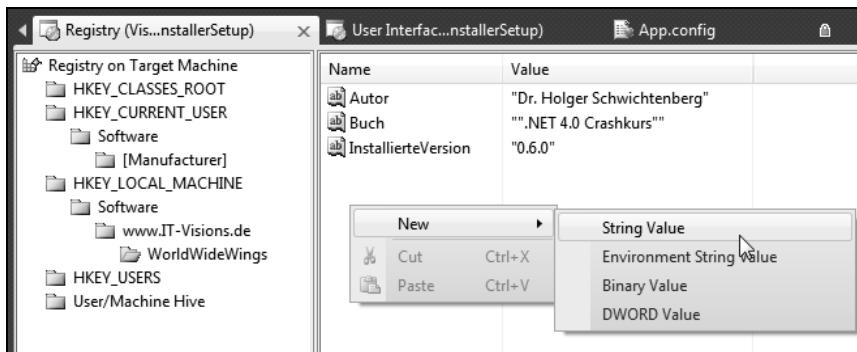


Abbildung 5.110 Festlegung von zu installierenden Registry-Einstellungen

Den Namen der Installationsroutine legt man in den Eigenschaften des Setup-Projekts fest. Hier kann man auch andere wichtige Einstellung vornehmen, z.B. ob beim Installieren vorherige Versionen entfernt werden sollen (*RemovePreviousVersions*).

Weitere Anpassungen der Benutzeroberfläche kann man über die Ansicht *User Interface* vornehmen (Texte der Standard-Dialogfelder verändern, Dialogfelder entfernen oder hinzufügen).

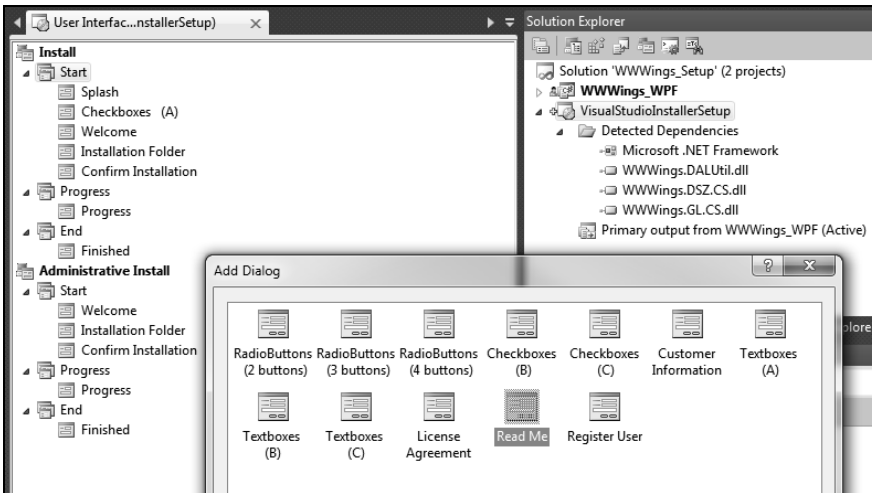


Abbildung 5.111 Anpassung der Benutzeroberfläche der Installationsroutine

Die Bildschirmabbildung zeigt, wie man ein Dialogfeld mit zwei Kontrollkästchen anlegt, in denen der Benutzer wählen kann, wo eine Verknüpfung zu der Anwendung entstehen soll.

ACHTUNG Die Dialogfeldvorlagen *Checkboxes* bestehen immer aus vier Kontrollkästchen, die man sichtbar/unsichtbar schalten muss.

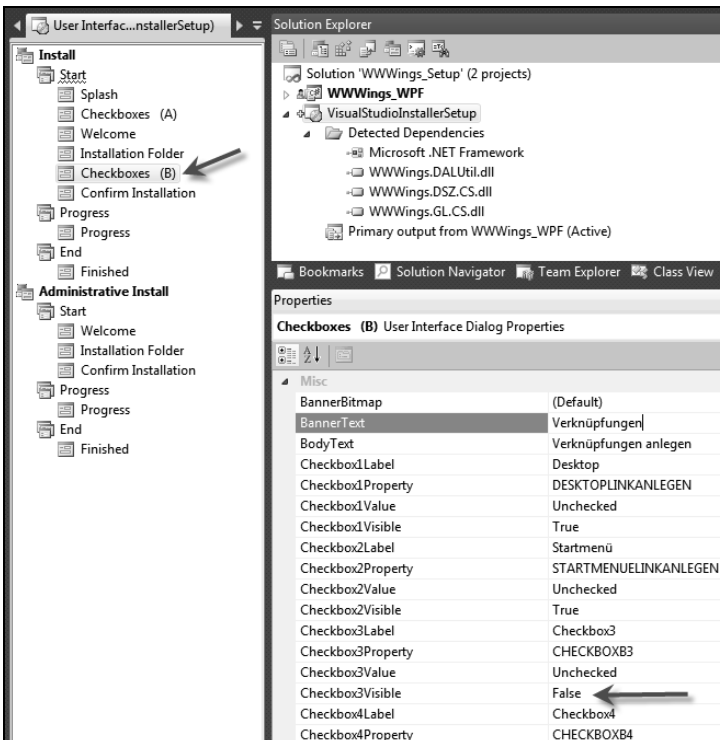


Abbildung 5.112 Definition von Kontrollkästchen

Damit die Installationsroutine auch reagieren kann wie gewünscht, muss man in den entsprechenden Verknüpfungen in der Datei- oder Registry-Ansicht eine »Condition« hinterlegen, z.B. STARTMENUE-LINKANLEGEN=1, wenn diese Aktion nur ausgeführt werden soll, wenn das so genannte Kontrollkästchen aktiviert wurde.

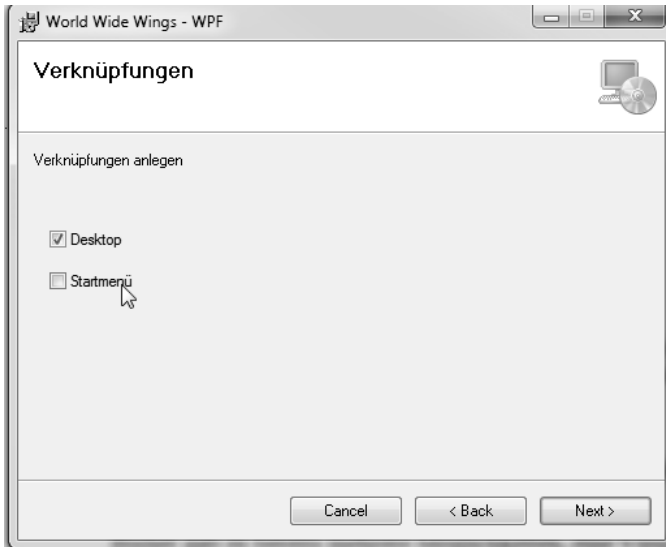


Abbildung 5.113 Ergebnis der Dialogfelddefinition zur Ausführungszeit der Installation

Außerdem gibt es die Möglichkeit, in der Ansicht *Launch Conditions* Bedingungen für die Installation festzulegen:

- **File Launch Condition** Eine Datei muss existieren
- **Registry Launch Condition** Ein Registry-Eintrag muss existieren
- **Windows Installer Launch Condition** Eine andere Anwendung muss existieren
- **.NET Framework Installer Launch Condition** Das .NET Framework oder .NET Framework Client Profile muss in einer bestimmten Version existieren
- **Internet Information Server Launch Condition** Der IIS muss existieren

ClickOnce-Deployment

Für das Click-Once-Deployment findet man in den Projekteigenschaften passender Projekte (WPF, Windows Forms, Konsole, Office) entsprechende Registerkarten mit dem Namen *Security* und *Publish*.

Wichtige Einstellungen sind:

- Ziel (Installationsordner): Entweder einen Netzwerkpfad oder ein IIS-Webserver

- Titel (Schaltfläche *Options/Description*)
- Soll eine HTML-Informationssseite generiert werden? (Schaltfläche *Options/Deployment*)
- Soll die Anwendung beim Start nach Updates suchen? (Schaltfläche *Updates*)
- Digitale Signatur für das Paket (Registerkarte *Signing*)
- Soll die Anwendung mit vollen Rechten (Full Trust) oder eingeschränkten Rechten (Partial Trust) laufen?

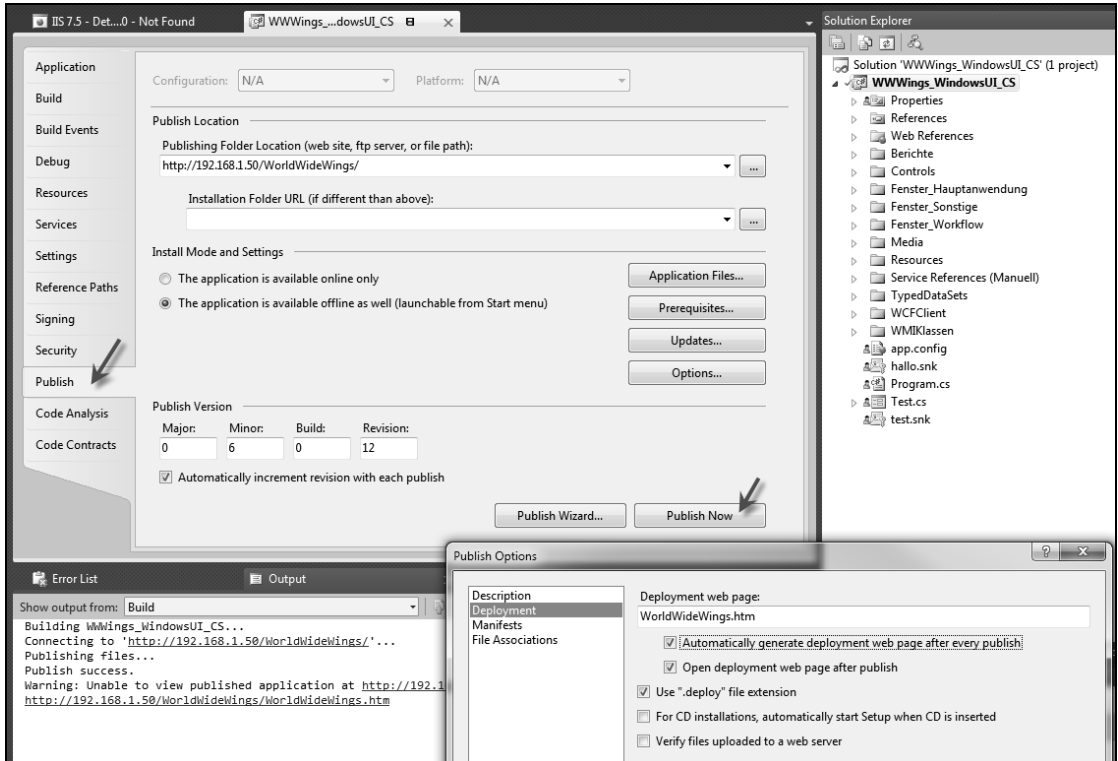


Abbildung 5.114 Veröffentlichungen eines Click-Once-Installationspakets

Die folgende Bildschirmabbildung zeigt die HTML-Informationssseite und den Start der Installation von dieser.

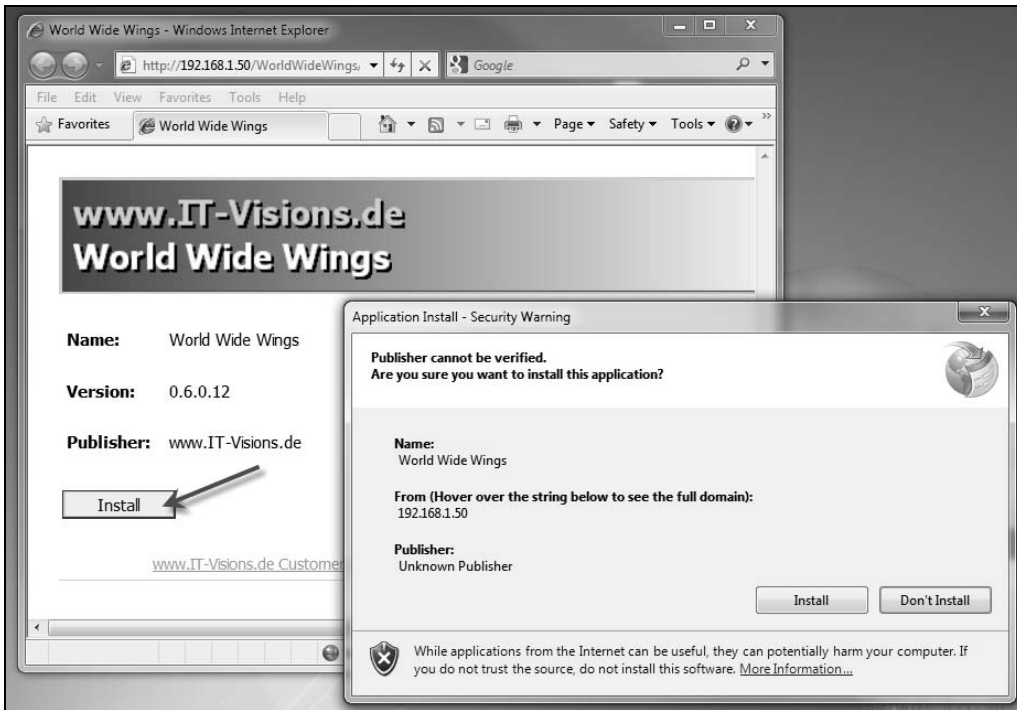


Abbildung 5.115 Ausführung des Click-Once-Deployments

Verbreiten von Webanwendung / Webservices mit dem IIS Web Deployment Tool

Bisher gab es bereits mehrere Möglichkeiten, ein Visual Studio-Projekt auf einem Internet Information Server (IIS) zu installieren:

- XCopy-Deployment der Dateien und anschließend manuelle oder skriptbasierte Konfiguration des IIS
- Erstellung von Microsoft Installer-Paketen (MSI) – in Visual Studio: Web Setup Project
- *Publish Website*-Funktion in Visual Studio 2005/2008
- *Web Deployment Projects* (WDP) als kostenloses Zusatzwerkzeug für Visual Studio 2005/2008
- IIS Web Deployment Tool (*MSDeploy.exe*) [<http://www.iis.net/download/webdeploy>]

Die ersten drei Möglichkeiten gibt es weiterhin in Visual Studio 2010. Anstelle der *Web Deployment Projects* tritt in Visual Studio 2010 aber nun die Integration des Microsoft Web Deployment Tool in die Visual Studio-Oberfläche.

Überblick über das IIS Web Deployment Tool

Das IIS Web Deployment Tool (kurz: *Web Deploy*) wird auch als *MSDeploy* bezeichnet, wobei dieser Name suggeriert, dass man damit auch mehr als nur Webanwendungen verbreiten könnte. Dies ist jedoch derzeit nicht möglich.

Grundidee des Web Deployment Tool ist es, alle für eine Website notwendigen Codedateien (*.aspx*, *.config*, *.asax*, *.css*, *.html*, *.dll*, *.svc*, ggf. auch *.cs* oder *.vb*), alle notwendigen Datendateien (z.B. *.mdf*, *.mdb*, *.xml*, *.resx*, etc.) und alle zugehörigen Konfigurationseinstellungen für IIS, das Dateisystem, den Global Assembly Cache (GAC), die Windows Registrierungsdatenbank und Microsoft SQL Server in ein ZIP-Paket zusammenzufassen. Dieses ZIP-Paket kann auf einem Zielsystem entpackt und angewendet werden, sodass eine komplette Webanwendung leicht von einem IIS-Webserver zum nächsten kopiert oder bewegt werden kann.

Als alternativen Verbreitungsweg bietet das IIS Web Deployment Tool die Verbreitung über einen Webservice (*msdeploy.axd*) an, der auf dem Zielsystem installiert sein muss.

HINWEIS

Es gibt einige Webhoster, die *msdeploy.axd* unterstützen und damit eine komfortable Möglichkeit zur Beförderung einer lokal entwickelten Anwendung auf den Webspace beim Provider bieten.

Erstellen von Deployment-Paketen

Visual Studio 2010 bietet drei Funktionen im Zusammenhang mit dem IIS Web Deployment Tool

- *Publish* im Kontextmenü eines Webprojekts oder im Menü *Build*
- *Package/Publish Settings* im Kontextmenü eines Webprojekts oder im Menü *Project*
- *Build Deployment Package* im Kontextmenü eines Webprojekts oder im Menü *Project*

Publish

Publish bietet die direkte Veröffentlichung der Webanwendung auf einem Zielsystem. *Publish* zeigt ein Dialogfeld, in dem man aus vier Verbreitungswegen wählen kann:

- **Web Deploy** Verbreitung über den Webservice (*msdeploy.axd*) des IIS Web Deployment Tool,
- **FTP** Kopieren auf einen FTP-Server
- **Filesystem** Kopieren im lokalen Dateisystem
- **FPSE** Kopieren auf einen IIS mit Front Page Server Extensions

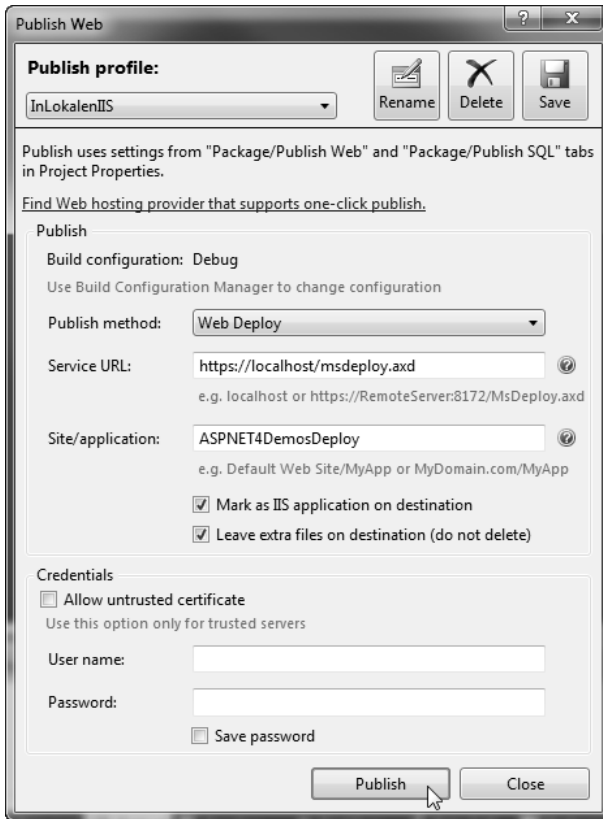


Abbildung 5.116 Funktion Publish

TIPP Einstellungen kann man als *Profil* anspeichern und später wieder aufrufen.

Package/Publish Settings

Mit *Package/Publish Settings* öffnet man die Registerkarte *Package/Publish Web* in den Projekteigenschaften. Hier kann man das Deployment-Paket für das IIS Web Deployment Tool konfigurieren. Man kann u.a. auswählen:

- Welche Dateien auf das Zielsystem kopiert werden
- Ob das Deployment-Paket als ZIP-Paket oder als unkomprimierter Ordner erstellt werden soll
- Wie die IIS-Website auf dem Zielsystem heißen soll

ACHTUNG Die Funktion zur Übernahme aller bestehenden IIS-Einstellungen ist ausgegraut, wenn man in Visual Studio 2010 den integrierten Web Development Web Server benutzt. Man muss die Website dann erst auf dem lokalen IIS konfigurieren und dies unter der Registerkarte *Web* auch Visual Studio bekannt machen.

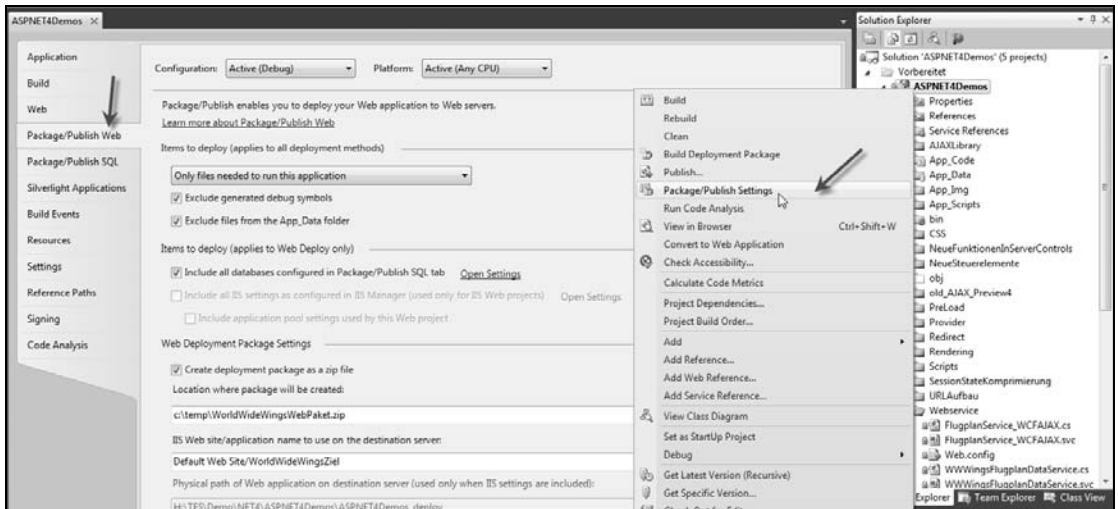
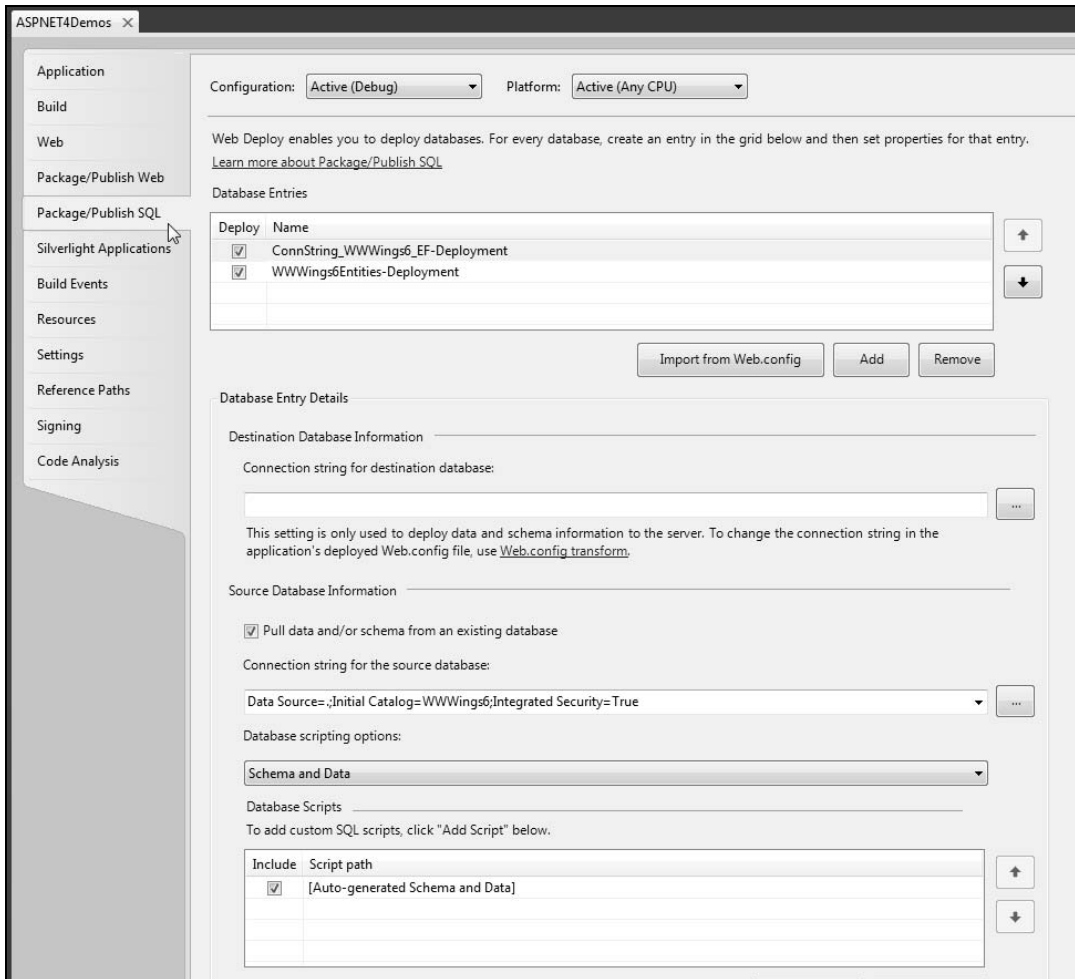


Abbildung 5.117 Projektregisterkarte *Package/Publish Web*

Auf einer weiteren Registerkarte legt man fest, ob und wie Datenbanken in das Deployment-Paket aufgenommen werden sollen. Man kann die bestehenden Verbindungszeichenfolgen aus der *web.config*-Datei übernehmen. Anschließend kann man die Verbindungen anwählen und dann separat festlegen:

- Verbindungszeichenfolge für das Zielsystem
- Ob Schema und Daten aus dem Entwicklungssystem übernommen werden sollen oder einfach nur die Verbindungszeichenfolge angepasst werden soll
- Ob in das Deployment-Paket nur das Schema, nur die Daten oder Schema und Daten des Entwicklungssystems übernommen werden sollen
- Optional weitere SQL-Skripte, die bei der Installation auf der Zieldatenbank ausgeführt werden sollen

Abbildung 5.118 Projektregisterkarte *Package/Publish SQL*

Build Deployment Package

Nach dem Konfigurieren des Deployment-Pakets stößt man die eigentliche Erstellung des Pakets dann an mit *Build Deployment Package*. Den Ablauf sieht man im *Output*-Fenster. Bei erfolgreicher Ausführung steht dort etwa:

```
Copying all files to temporary location below for package/publish:
obj\Debug\Package\PackageTmp.
Adding MSDeploy.dbFullSql (MSDeploy.dbFullSql).
Adding dbFullSql
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\ConnString_SchemaAndData.sql).
Adding child sqlScript
(MSDeploy.dbFullSql/dbFullSql[@path='H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\
ConnString_SchemaAndData.sql']/sqlScript).
Updating dbFullSql (H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\WWWings6Entities-
Deployment_SchemaAndData.sql).
```

```

Adding child sqlScript
(MSDeploy.dbFullSql/dbFullSql[@path='H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\WW
Wings6Entities-Deployment_SchemaAndData.sql']/sqlScript).
Scanning sql command variable(s) from sql script
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\ConnString_SchemaAndData.sql).
Scanning sql command variable(s) from sql script
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\WWWings6Entities-
Deployment_SchemaAndData.sql).
Packaging into c:\temp\WorldWideWingsWebPaket.zip.
Adding package (package).
Adding child iisApp (H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\Package\PackageTmp).
Adding child createApp (H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\Package\PackageTmp).
...
Adding child filePath
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\Package\PackageTmp\AJAXLibrary\Flugplan_Filter_Dat
aContext_ADS.aspx).
Adding child filePath
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\Package\PackageTmp\AJAXLibrary\Flugplan_Filter_Dat
aContext_WCF.aspx).
...
Adding child dbFullSql
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\ConnString_SchemaAndData.sql).
Adding child sqlScript
(sitemanifest/dbFullSql[@path='H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\ConnString_
SchemaAndData.sql']/sqlScript).
Adding child dbFullSql
(H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\WWWings6Entities-
Deployment_SchemaAndData.sql).
Adding child sqlScript
(sitemanifest/dbFullSql[@path='H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos\obj\Debug\AutoScripts\WWWings6
Entities-Deployment_SchemaAndData.sql']/sqlScript).
Adding declared parameter 'IIS Web Application Name'.
Adding declared parameter 'ConnString_WWWings6_EF-Deployment Connection String'.
Adding declared parameter 'WWWings6Entities-Deployment Connection String'.
Adding declared parameter 'ConnString_WWWings6_EF-Web.config Connection String'.
Adding declared parameter 'WWWings6Entities-Web.config Connection String'.
Package "WorldWideWingsWebPaket.zip" is successfully created as single file at the following location:
file:///c:/temp
To get the instructions on how to deploy the web package please visit the following link:
http://go.microsoft.com/fwlink/?LinkId=124618
Sample script for deploying this package is generated at the following location:
c:\temp\WorldWideWingsWebPaket.deploy.cmd
For this sample script, you can change the deploy parameters by changing the following file:
c:\temp\WorldWideWingsWebPaket.SetParameters.xml
===== Build: 5 succeeded or up-to-date, 0 failed, 0 skipped =====
===== Publish: 1 succeeded, 0 failed, 0 skipped =====

```

Listing 5.6 Ausschnitt aus dem Protokoll einer erfolgreichen Paketbildung

Verbreiten von Deployment-Paketen

Zum Installieren eines Deployment-Pakets benötigt man einen Internet Information Server 7.x mit dem Web Deployment Tool, das man als kostenlose Zugabe unter [<http://www.iis.net/download/webdeploy>] erhält. In der IIS-Konsole gibt es dann im Kontextmenü der Servers oder einer einzelnen Website ein neues Kontextmenü *Deploy*.

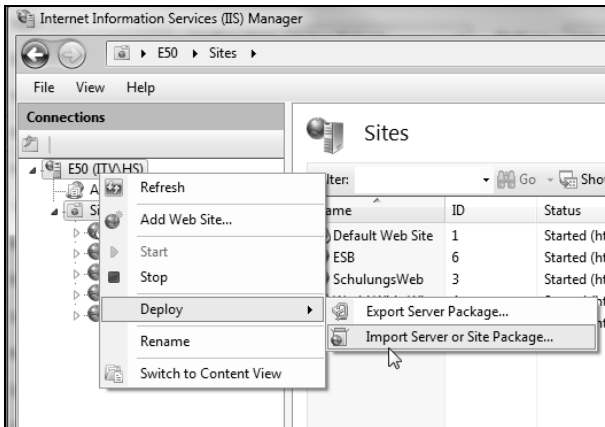


Abbildung 5.119 Menü *Deploy* in der IIS-Konsole

Im ersten Schritt des Assistenten wählt man das zu installierende ZIP-Paket. Die zweite Seite des Assistenten gibt einen Überblick über den Inhalt des Pakets.

ACHTUNG Man muss die Ebene für das Deployment (Server-Ebene oder Website-Ebene) in Abhängigkeit davon wählen, auf welcher Grundlage das Paket erstellt wurde. Wenn ein Paket auf Basis eines virtuellen Verzeichnisses erstellt wurde und man dann versuchen würde, es auf Serverebene zu installieren, erscheint die Fehlermeldung »The selected package contains applications which cannot be installed at server level«, denn es fehlen Informationen über die IIS-Website. Man muss dann die IIS-Website erst manuell anlegen. Umgekehrt sieht man bei dem Versuch, ein auf Basis einer IIS-Website erstelltes Paket unterhalb einer bestehenden IIS-Website zu installieren, den Fehler: »The selected package contains sites which cannot be installed under another site«. Dies kann man aber heilen, indem man die Serverebene wählt oder auf Siteebene im Assistentenschritt *Contents of the Package* den IIS-Anwendungspool vom Deployment ausschließt.

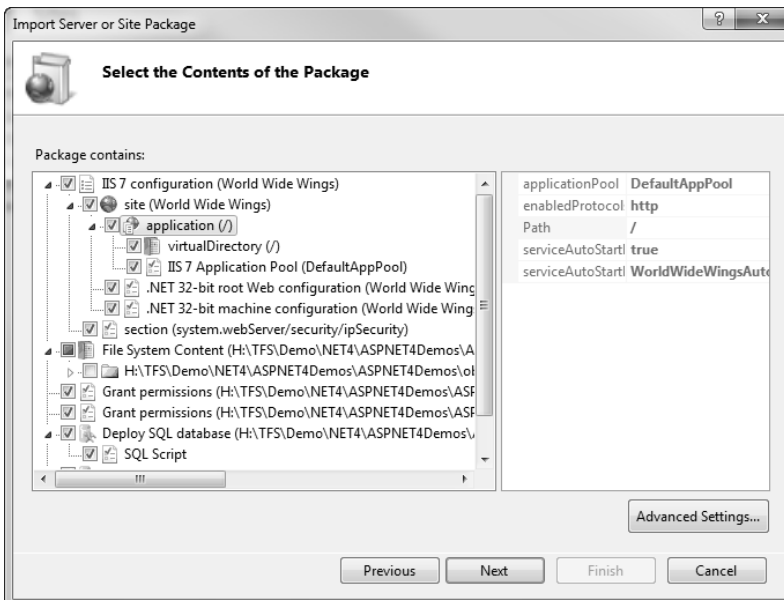


Abbildung 5.120 Inhalt des Pakets

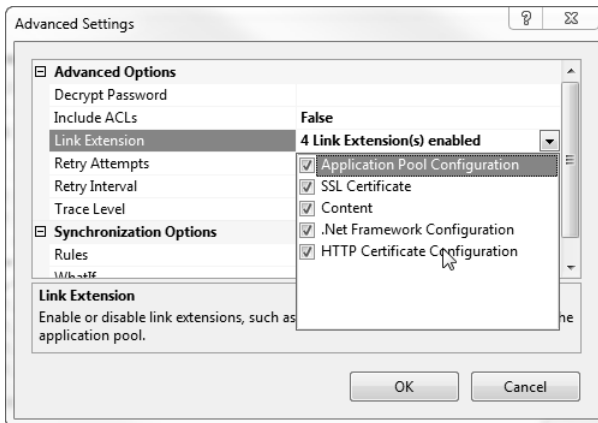


Abbildung 5.121 Erweiterte Einstellung für das Deployment

Im dritten Schritt ist zum einen der Pfad im IIS-Verzeichnisbaum zu setzen und die Verbindungszeichenfolgen für die Datenbanken.

ACHTUNG Hier sind einige Dinge zu beachten, sonst läuft man in Probleme:

- Für die Verbreitung eines virtuellen Verzeichnisses muss man als *Application Path* eine bestehende IIS-Website mit angeben, z.B. *Default Web Site/WorldWideWings40*.
- Für jede im Paket konfigurierte Datenbankverbindung wird man nach einem Ziel-Datenbankserver gefragt. Leider sind die Dialogfelder nicht gut beschriftet, sodass man die verschiedenen Datenbankverbindungen nur gemäß ihrer im Paket hinterlegten Reihenfolge identifizieren kann.
- Datenbankverbindungen sind als normale Verbindungszeichenfolgen anzugeben, auch wenn die Ursprungsverbindungen für das ADO.NET Entity Framework angelegt waren, das umfangreichere Verbindungszeichenfolgen besitzt.

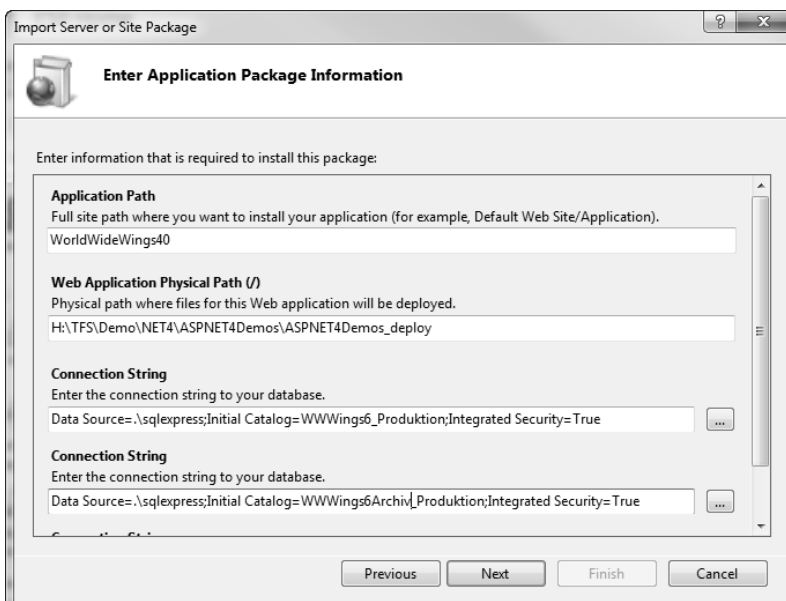


Abbildung 5.122 Festlegung der Ziele

Nach Beenden dieses Schrittes wird die Installation der Website und der Datenbanken ausgeführt. Der Benutzer sieht am Ende ein Protokoll des Installationshergangs.

TIPP

Alternativ kann man die Installation eines Deployment-Pakets an der Kommandozeile ausführen mit `msdeploy.exe`. Codebasiert kann man mit der Klasse `Microsoft.Web.Deployment.DeploymentManager` installieren.

Konfigurationstransformationen

Beim Installieren von Anwendungen ist die Regel, dass auf dem Zielsystem andere Konfigurationseinstellungen als auf dem Entwicklersystem gelten (z.B. Datenbankverbindungen). Microsoft bietet (derzeit nur für Webanwendungen) eine Möglichkeit, die in den XML-Konfigurationsdateien (*web.config*) gespeicherten Konfigurationsinformationen automatisiert abzuändern im Zuge der Erstellung eines Deploymentpakets ab Visual Studio 2010.

Die beim Installieren des Deployment-Pakets angegebenen Verbindungszeichenfolgen dienen nur dazu, die Datenbankskripte während der Installation auszuführen. Die in der *web.config*-Datei hinterlegten Verbindungszeichenfolgen bleiben unberührt. Um diese anzupassen, muss man in Visual Studio zusätzlich so genannte »Konfigurationsdateitransformationen« anlegen. In einer Konfigurationsdateitransformation legt man fest, welche Teile der Konfiguration beim Verbreiten ersetzt, gelöscht und ergänzt werden sollen.

Microsoft verwendet hierfür nicht die XML Stylesheet Transformations (XSLT), sondern eine eigene XML-basierte Syntax, die Microsoft XML-Document-Transform (XDT) nennt.

Erstellen der Transformationsdateien

Im ersten Schritt wählt man im Kontextmenü einer *web.config*-Datei die Aktion *Add Config Transformations*. Visual Web Developer erzeugt dann für jede im Configuration Manager hinterlegte Build Configuration eine Transformationsdatei.

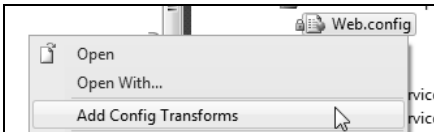


Abbildung 5.123 Automatisches Erstellen der Transformationsdateien

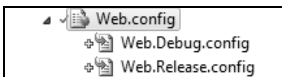


Abbildung 5.124 Erstellte Transformationsdateien

Syntax der Transformationsdateien

Die Syntax der Transformationsdateien besteht aus zwei Arten von Befehlen: Locators und Transformers.

Befehl	Beschreibung
<code>xdt:Locator="Match(xy)"</code>	Finde ein Element, das ein Attribut xy besitzt mit dem gleichen Inhalt
<code>xdt:Locator="Condition(@x='a' or @y='b')"</code>	Finde ein Element innerhalb des aktuellen Elements, das auf den XPath-Ausdruck zutrifft
<code>xdt:Locator="XPath(/m/n)"</code>	Finde ein Element global im Dokument, das auf den XPath-Ausdruck zutrifft

Tabelle 5.14 Locators

xdt:Transform="Replace"	Ersetzt das erste passende Element
xdt:Transform="Remove"	Entfernt das erste passende Element
xdt:Transform="RemoveAll"	Entfernt alle passenden Elemente
xdt:Transform="Insert"	Füge ein Element ein
xdt:Transform="SetAttributes(x)"	Setzt einen Attributwert für Attribut x
xdt:Transform="RemoveAttributes(x,y,z)"	Entfernt die drei Attribute x, y und z
xdt:Transform="InsertAfter(/m/n/[@x='a'])"	Ergänzt das Element nach dem durch den X-Path-Ausdruck festgelegten Element
xdt:Transform="InsertBefore(/m/n/[@x='a'])"	Ergänzt das Element vor dem durch den X-Path-Ausdruck festgelegten Element

Tabelle 5.15 Transformers

HINWEIS Voraussetzung ist folgende XML-Namensraumdeklaration: `<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">`

Das folgende Listing zeigt an drei Beispielen die typischen Aktionen:

- Entfernen eines Attributs (hier: `Debug="true"`)
- Hinzufügen eines XML-Elements (hier: `<customErrors>`)
- Ersetzen eines XML-Elements (hier: `<connectionStrings><add>`)

```
<?xml version="1.0"?>
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <system.web>
    <!-- Entferne Debug-Attribut -->
    <compilation xdt:Transform="RemoveAttributes(debug)" />
    <!-- Ergänze Umlenkung für Statuscode 500 -->
    <customErrors defaultRedirect="GenericError.htm"
      mode="RemoteOnly" xdt:Transform="Replace">
      <error statusCode="500" redirect="InternalServerError.htm"/>
    </customErrors>
  </system.web>
  <connectionStrings>
    <!-- Setze Verbindungszeichenfolge -->
    <add name="WWWings6EF"
      connectionString="metadata=res://*/EF.EFModel1.csd1|res://*/EF.EFModel1.ssd1|res://*/EF.EFModel1.ms1;
      provider=System.Data.SqlClient;provider connection string="Data Source=.\sqlexpress;
      Initial Catalog=WWWings6_Produktion;Integrated Security=True;MultipleActiveResultSets=True";
      providerName="System.Data.EntityClient" xdt:Transform="Replace" xdt:Locator="Match(name)"/>
    </connectionStrings>
  </configuration>
```

Listing 5.7 Web.Release.config

Ergebnis der Transformation

Das folgende Listing zeigt Ausschnitte aus der *web.config*-Datei nach der Installation.

```
<?xml version="1.0"?>
<configuration>
<connectionStrings>
  <!-- Hauptgeschäftsdaten-Datenbank Version 6 für Modell in WWings.G0.csproj-->
  <add name="WWings6EF"
connectionString="metadata=res://*/EF.EFModel1.csd1|res://*/EF.EFModel1.ssd1|res://*/EF.EFModel1.msl;
provider=System.Data.SqlClient;provider connection string=&quot;
Data Source=.\sqlexpress;Initial Catalog=WWings6_Produktion;
Integrated Security=True;MultipleActiveResultSets=True&quot;;" providerName="System.Data.EntityClient" />
</connectionStrings>

<runtime>
  <appDomainResourceMonitoring enabled="true" />
</runtime>

<system.web>
...
<compilation targetFramework="4.0">
<customErrors defaultRedirect="GenericError.htm" mode="RemoteOnly">
  <error statusCode="500" redirect="InternalError.htm" />
</customErrors>
...
</system.web>
...
</configuration>
```

Listing 5.8 Ausschnitt aus der *web.config*-Datei nach der Installation des Pakets

Nutzung des Team Foundation Server (TFS)

Dieser Abschnitt gibt einen kurzen Überblick über das Application Lifecycle Management (ALM) mit dem Team Foundation Server (TFS). TFS ist ein komplexes Thema. Keineswegs kann dieser Abschnitt das Lesen eines Buchs zu TFS ersetzen.

TIPP Auch für TFS 2010 gibt es wieder eine Sammlung von Zusatzwerkzeugen, die es nicht mehr in das Produkt geschafft haben, die TFS 2010 Power Tools (<http://visualstudiogallery.msdn.microsoft.com/en-us/3e8c9b68-6e39-4577-b9b7-78489b5cb1da>).

Funktionsüberblick

Bei den Funktionen die Visual Studio 2010 Ultimate bietet, kann man differenzieren zwischen solchen, die nur in Verbindung mit dem TFS arbeiten und solchen, die man auch lokal ohne TFS verwenden kann. Lokal verwendbare ALM-Funktionen sind insbesondere:

- Modellierung (ab Visual Studio 2010 auch begrenzte UML-Funktionen)
- Ablaufverfolgung während des Debuggens (IntelliTrace)

- Statische Codeanalyse und -Metriken (Richtlinien und Codequalität)
- Unit Tests, Testabdeckung (Code Coverage), Webtests und Desktop-Oberflächentests
- Profiling (Geschwindigkeits- und Speicherverbrauchsmessung)
- Datenbanken (Versionierung und Deployment von DB-Schemata, Testdatengenerator, Unit Tests in T-SQL)

Nur in Verbindung mit einem TFS funktionieren:

- Quellcodeverwaltung (Team Foundation Version Control - TFVC)
- Projektmanagement
- Serverseitige Kompilierung/Continuous Integration (Build-Server)
- Tätigkeitsverwaltung (Anforderungen, Aufgaben, Fehler), die man hierarchisch anordnen kann
- Berichtswesen
- Team Web Access (Webportal u.a. zum Melden von Fehlern) und Projektportal mit Dokumentenverwaltung (auf SharePoint-Basis)
- Lab Management (virtualisierte Testsysteme für automatische Tests)

Der Wert des Einsatzes eines TFS liegt vor allem in der Integration der einzelnen Funktionsbereiche. So kann ein Entwickler beim Bereitstellen von Code im Quellcodeverwaltungssystem (Check-In) nicht nur wie sonst üblich einen Kommentar hinterlegen, sondern den Check-In direkt einer oder mehreren Aufgaben zuordnen, z.B. gemeldeten Fehlern. Ein Projektleiter kann im TFS eine solche Zuordnung verpflichtend machen durch so genannte Check-In-Policies. Visual Studio beschwert sich dann beim Entwickler, wenn er den Check-In keiner Tätigkeit zuordnet. Ab TFS 2010 kann eine Check-In-Policy auch besagen, dass der eingelagerte Code erst eine Liste von definierten Unit Tests und Codeanalysen bestehen muss. Solange ist der Code nur vorläufig eingecheckt (Gated Check-In). Erst wenn der Code die Tests bestanden hat, können ihn die anderen Entwickler abrufen.

Gated Check-Ins sind ein Beispiel, wie clientseitige Funktionen von Visual Studio einen Mehrwert durch die Ausführung auf der Serverseite generieren. Ein zweites Beispiel ist Continuous Integration. Durch Konfiguration eines so genannten Build-Servers kann man erreichen, dass Projekte serverseitig kompiliert werden. Diese serverseitige Kompilierung kann entweder periodisch erfolgen oder aber nach jedem Check-In. Das Ergebnis der Kompilierung kann in Visual Studio oder dem Webportal abgerufen werden oder die Entwickler können per E-Mail informiert werden.

Für Projektleiter bietet TFS eine Vielzahl von Berichten, die den Projektfortschritt anzeigen, z.B. die Anzahl der implementierten Features und die Anzahl der gemeldeten Fehler im Zeitablauf. Berichte kann man über Visual Studio, das Webportal oder Excel abrufen. Natürlich integriert sich der TFS in das Sicherheitssystem von Windows Server, d.h. durch die Zuordnung von Benutzern zu Gruppen in Active Directory erhalten Anwender Rechte auf dem TFS.

HINWEIS

Der TFS speichert alles (Aufgaben, Quellcode, Build-Prozesse, etc.) in einer oder mehreren Microsoft SQL Server-Datenbanken. Anders als der Vorgänger kann TFS 2010 mehrere verschiedene Datenbanken (so genannte *Collections*) besitzen. So kann man TFS-Projekte besser isolieren, z. B. in Hosting-Szenarien. TFS-Projekte lassen sich in Team Project Collections zusammenfassen.

Team Explorer und Source Control Explorer

Die Schaltzentrale für die Arbeit mit dem Team Foundation Server ist der Team Explorer. In Visual Studio 2008 war er noch eine Erweiterung, in Visual Studio 2010 gehört er zum Standardlieferungsumfang.

Im Team Explorer muss man zunächst eine Verbindung zu einem Team Server aufbauen (Symbol *Connect to Team Project*). Ein Team Projekt kann eine Vielzahl von Visual Studio-Projekten und -Projektmappen umfassen. Der Team Explorer zeigt die gewählten Team-Projekte in einem Baum.

Unter jedem Team-Projekt sieht man Äste für die Tätigkeitsverwaltung (Work Items), Dokumentenverwaltung (Documents), Berichte (Reports), serverseitige Übersetzungsvorgänge (Build), die Liste der Teammitglieder und eine Verknüpfung zu den zugehörigen Dateien im Source Control Explorer.

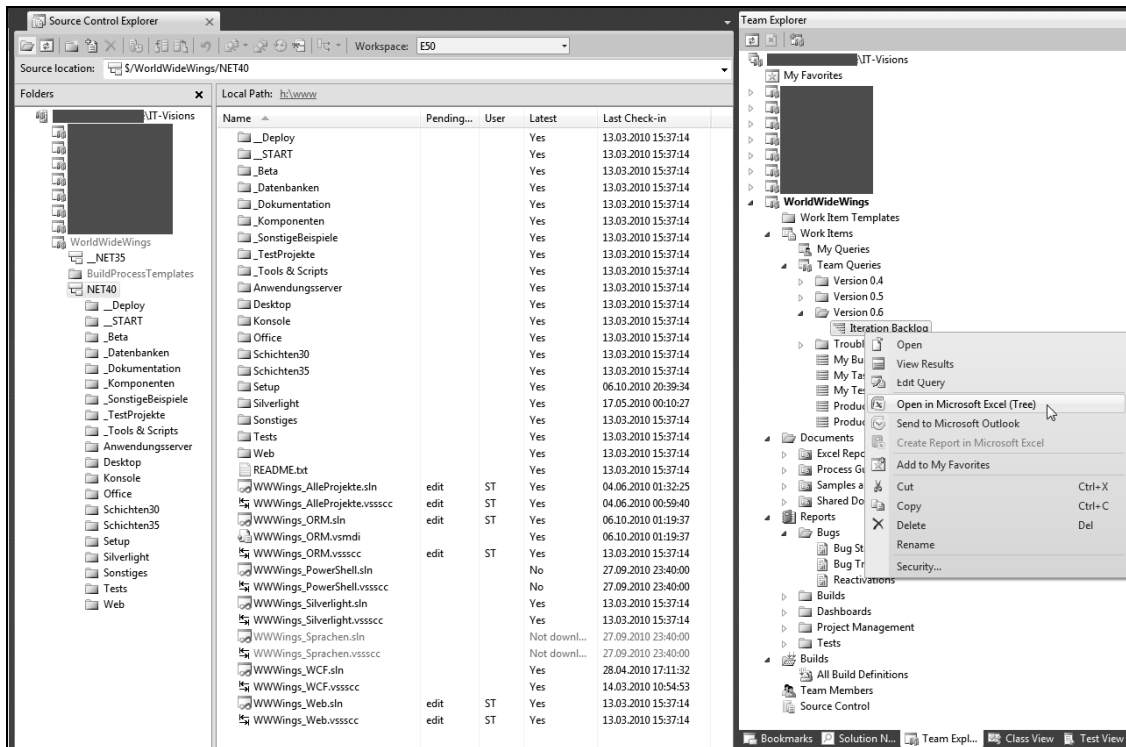


Abbildung 5.125 Team Explorer und Quellcodeverwaltungs Fenster

Tätigkeitsverwaltung (Work Items)

TFS erlaubt verschiedene Typen von Tätigkeiten. Im Standard kennt TFS als Tätigkeitstypen z.B. Task, Feature, Bug und Test Case, jeweils mit zahlreichen Attributen. Sowohl die Tätigkeitstypen als auch die Attribute lassen sich aber umdefinieren.

Das Erstellen und Abfragen von Tätigkeiten ist nicht nur über die Visual Studio-Oberfläche (siehe folgende Bildschirmabbildung) und die TFS-Weboberfläche möglich, sondern auch über Microsoft Excel und Microsoft Project sowie – durch das Zusatzprodukt Team Companion [<http://www.teamcompanion.com>] – auch in Outlook. Neu auf TFS 2010 ist die Möglichkeit zur hierarchischen Anordnung von Tätigkeiten.

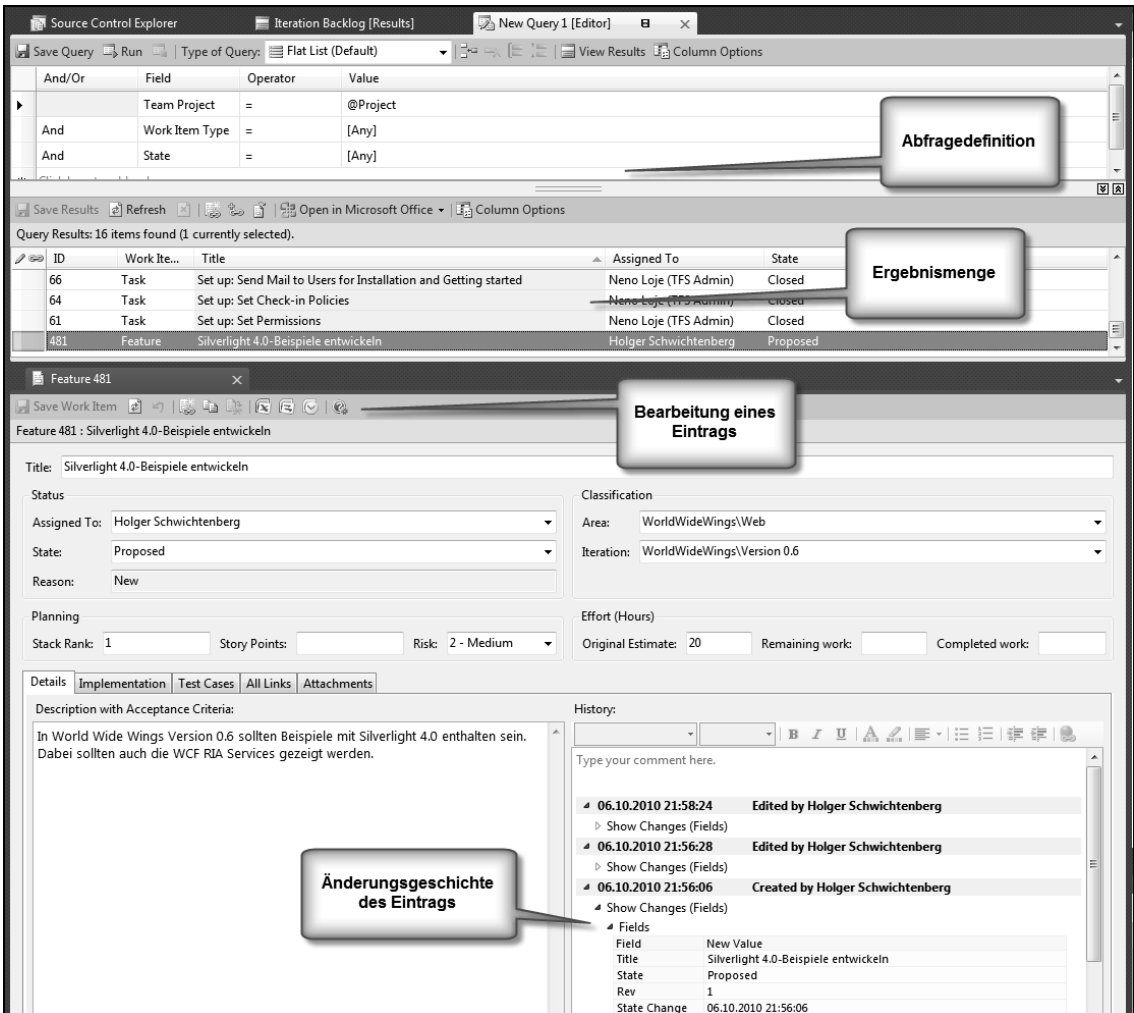


Abbildung 5.126 Aufgabenverwaltung in Visual Studio 2010 mit TFS 2010

Quellcodeverwaltung

Für die Quellcodeverwaltung mit TFS (alias Team Foundation Version Control (TFVC)) muss man unter *Tools/Options/Source Control/Plug-in Selection* zunächst den TFS als Quellcodeverwaltungssystem auswählen.

HINWEIS Team Foundation Version Control (TFVC) ist das im Team Foundation Server integrierte Versionsverwaltungssystem. TFVC ist nicht zu verwechseln mit Microsoft Visual SourceSafe (VSS). TFVC ist ein komplett neues System, das viele Schwächen von VSS überwindet.

Danach kann man beim Anlegen eines Visual Studio-Projekts durch ein Häkchen wählen, dass das neue Projekt dem Versionsverwaltungssystem hinzugefügt werden soll. In einem Dialogfeld fragt Visual Studio dann nach dem Standort im Versionsverwaltungssystem.

Alternativ kann man über den Source Control Explorer bestehende Dateien vom TFS herunterladen. Beim Aufruf von *Get latest Version* fragt Visual Studio dann nach dem gewünschten Standort im Dateisystem, wenn die Dateien noch nicht abgerufen waren. Ansonsten holt *Get latest Version* immer die aktuellste Version.

Nach dem Ändern von Dateien (markiert durch einen roten Pfeil neben dem Symbol, siehe Bildschirmabbildung) wählt man im Projektmappenexplorer oder Source Control Explorer die Funktion *Check-In*. Im Check-In-Dialogfeld zeigt Visual Studio eine Liste aller geänderten, neuen oder gelöschten Dateien. Abhängig von der Hierarchieebene, auf der *Check-In* angewählt wurde, sind die Elemente ausgewählt. Man kann einen Kommentar eingeben und eine Verbindung zu Tätigkeiten herstellen. Der TFS-Projektadministrator kann solche Angaben zur Pflicht machen. Der Entwickler kann dann zwar immer noch davon abweichen, wird aber aufgefordert, eine Begründung anzugeben.

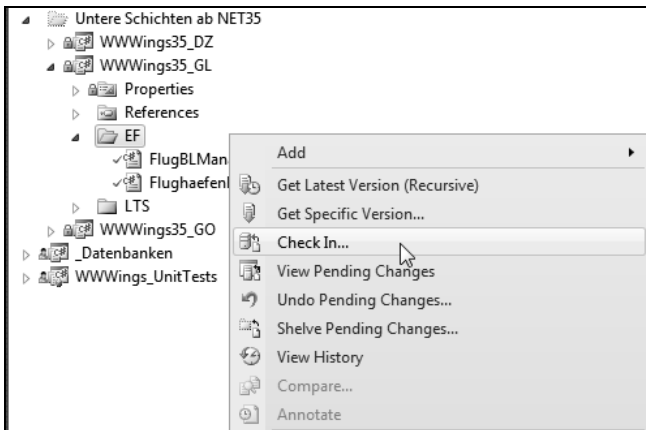


Abbildung 5.127 Versionsverwaltungsfunktionen im Projektmappenexplorer

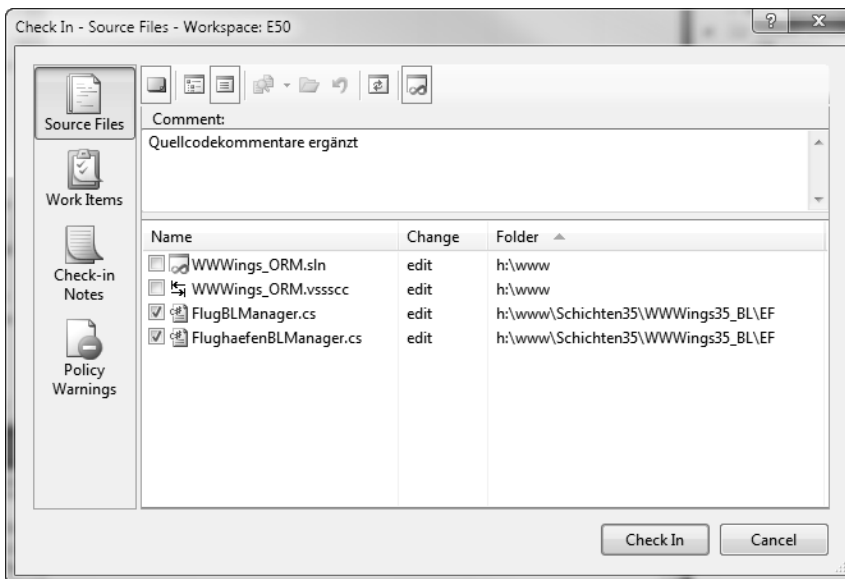


Abbildung 5.128 Dialogfeld beim Check-In

TIPP

Im Standard sperrt der TFS heruntergeladene Dateien nicht zur Änderung durch andere Entwickler. Man kann aber eine solche Sperre manuell auf einzelnen Dateien, Ordner und ganzen Projekten setzen durch die Funktion *Check-Out for Edit*. Vorsicht ist aber geboten: Wenn ein Entwickler diese Sperre nicht per *Check-In* oder *Undo* aufhebt, kann kein anderer Entwickler mehr die Datei bearbeiten.

Das ist unangenehm, wenn der sperrende Entwickler nicht nur im Feierabend ist, sondern die nächsten drei Wochen im Urlaub oder die Festplatte in seinem Rechner das Lebensende erreicht hat. In diesen Fällen muss der TFS-Administrator manuell eingreifen.

HINWEIS

Ein Gated Check-In ist eine Form der pessimistischen Continuous Integration. Ein Check-In von Quellcode wird auf der Serverseite zunächst vorläufig gespeichert und dann einem Build-Prozess (ggf. inklusive Unit Tests) unterzogen. Erst wenn der Build-Prozess erfolgreich war, steht der Quellcode anderen Entwicklern zur Verfügung. War der Build-Prozess nicht erfolgreich, wird alles zurückgerollt und der Entwickler benachrichtigt, der den Quellcode geliefert hat. Mit dieser Funktion soll verhindert werden, dass ein Projekt nach einem Check-In nicht mehr automatisch gebaut werden kann.

Mit dem Source Control Explorer kann man Verzweigungen (Branches) und Zusammenführungen von Zweigen (Merging) realisieren. Ab Version 2010 kann man diese Verzweigungen und Zusammenführungen auch grafisch darstellen. Es ist möglich, genau nachzuverfolgen, in welchen Branches ein bestimmtes Work Item (mit seinen Changesets) eingeflossen ist. Direkt aus der Visualisierung kann per Drag & Drop die Zusammenführung (Merging) erfolgen.

Automatische Kompilierungsvorgänge (Continuous Integration)

Im Team Foundation Explorer im Ast *Builds* kann der Entwickler serverseitige Kompilierungsvorgänge definieren und überwachen. Kompilierungsvorgänge können manuell, zeitgesteuert oder durch einen Check-In von geändertem Quellcode angestoßen werden. In einer *Build Definition* legt man neben einem Trigger (siehe Bildschirmabbildung) auch fest, welche Projektmappen übersetzt werden sollen. Optional kann man Unit Tests angeben, die auf dem Server nach der Übersetzung ausgeführt werden müssen. Ebenso muss man festlegen, wie lange die Kompilierungsvorgänge und deren Ergebnisse aufbewahrt werden sollen.

ACHTUNG

Der TFS Build Server lädt jeweils die aktuelle Version aller Dateien aus dem Versionsverwaltungssystem und bewahrt diese auf, um Fehler nachvollziehen zu können. Die Datenmenge auf dem TFS Build Server kann also schnell anwachsen.

Die nachstehende Bildschirmabbildung zeigt jeweils nur die letzten fünf erfolgreichen und die letzten fünf fehlgeschlagenen Kompilierungsvorgänge, weil dies so eingestellt wurde. Daher der zeitliche Abstand zwischen den Vorgängen.

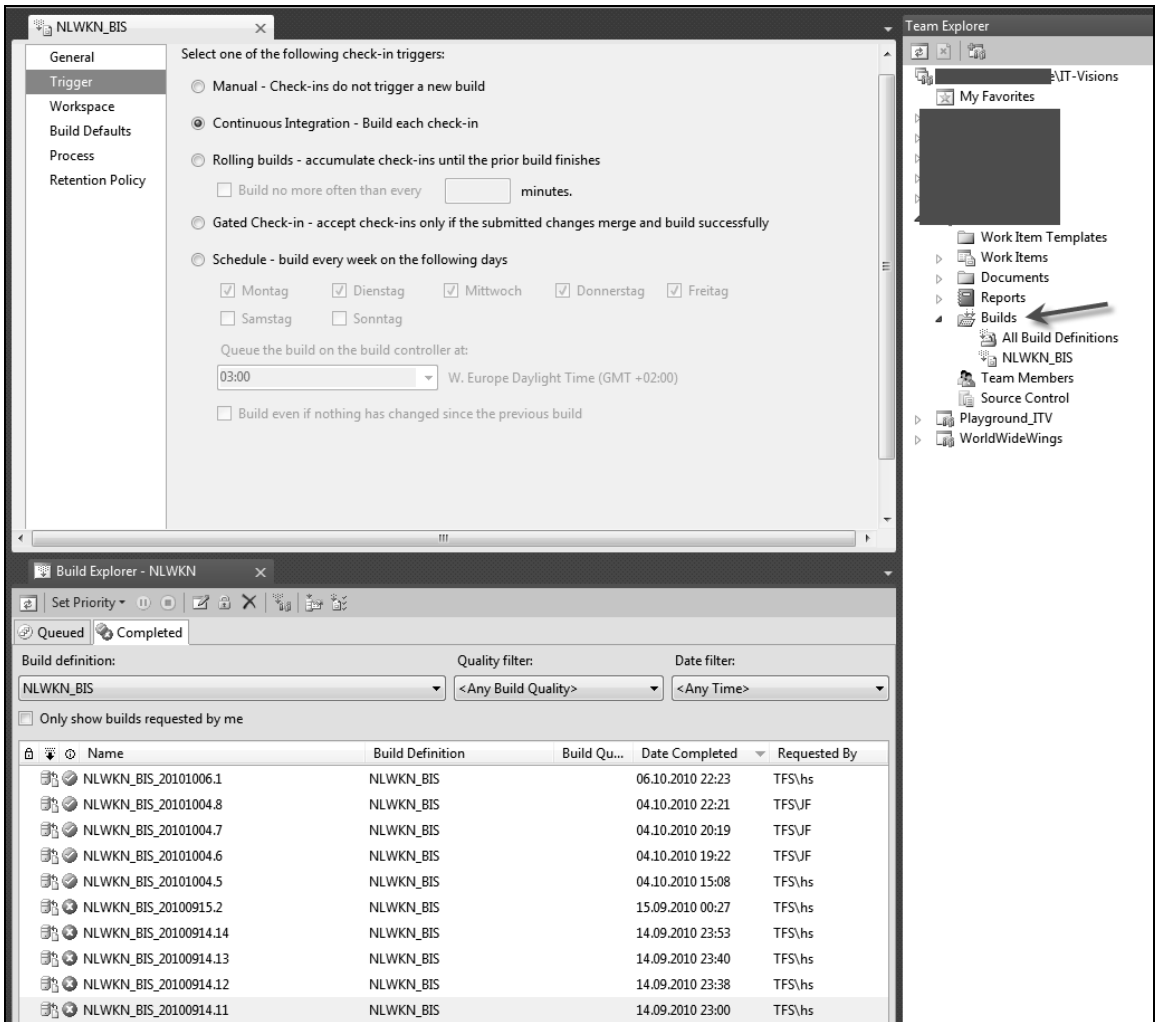


Abbildung 5.129 Übersetzungsdefinitionen und -ergebnisse

TIPP

Bei den serverseitigen Kompilierungsvorgängen muss der TFS natürlich über alle notwendigen Komponenten verfügen. Wenn zum Beispiel eine Komponentensammlung eines Drittanbieters zum Einsatz kommt, die pro Arbeitsplatz eine Lizenz erfordert, müssen Sie oft auch die Installation auf dem TFS Build Server extra lizenzieren.

Benachrichtigungen

Im Alerts Explorer (aus den TFS Power Tools), der über das Menü *Team* erreicht wird, kann man festlegen, bei welchen Ereignissen (z.B. neue Aufgabe, Zustandsänderung einer Aufgabe, Check-In, fehlgeschlagene Übersetzung) welche Personen eine E-Mail erhalten sollen.

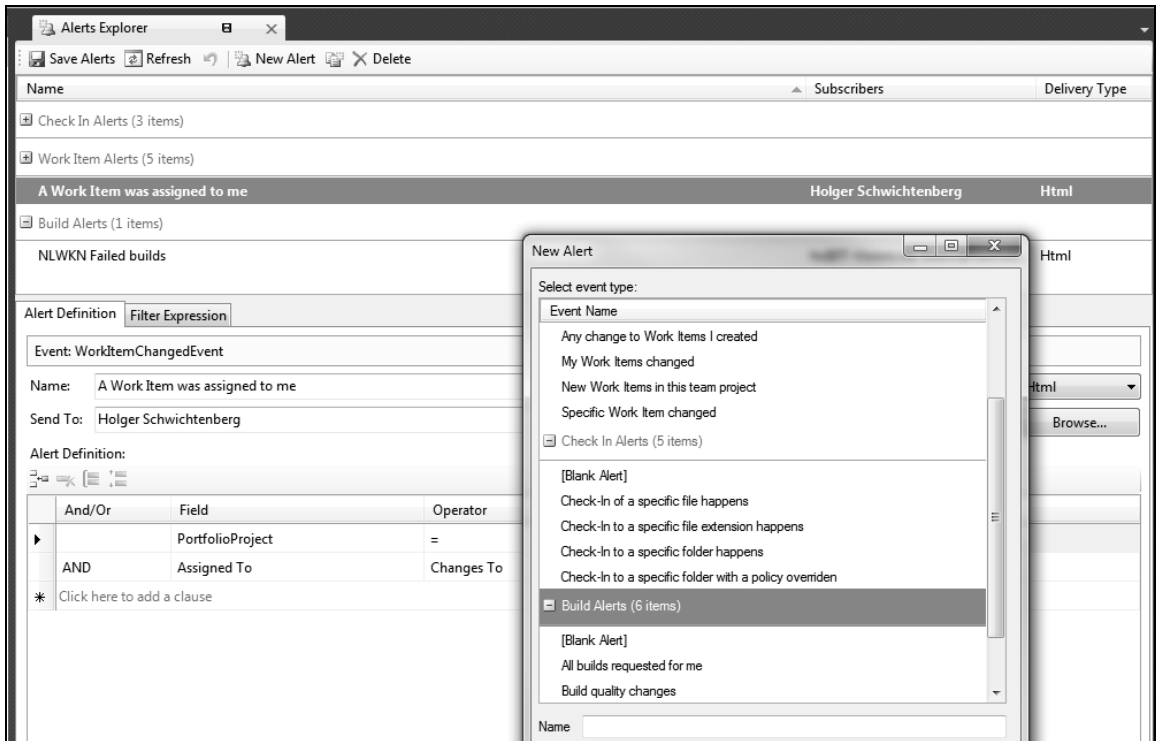


Abbildung 5.130 Alerts Explorer

Installation und Lizenzierung

Für die Lizenzierung des TFS benötigt man neben der Serverlizenz auch Client-Zugriffslizenzen (Client Access Licence, CAL)

Vor TFS 2010 gab es zwei wesentliche Hintergründe für den Einsatz des TFS: die hohen Lizenzkosten und der hohe Installationsaufwand. An beiden Stellschrauben hat Microsoft gedreht. Eine TFS 2010-Lizenz gibt es nun für rund 500 Euro und beinhaltet fünf Client-Zugriffslizenzen, also das Recht, dass fünf Clients zugreifen können (nicht aber der Client selbst). Das ist dann nur noch rund ein Fünftel des bisherigen Preises für TFS 2008. Erst ab dem sechsten Teammitglied wären weitere 400 Euro pro Person fällig. MSDN-Abonnenten erhalten seit April 2010 neben der Client-Zugriffslizenz (CAL) auch noch eine Serverlizenz. So könnte theoretisch jeder Entwickler seinen eigenen TFS auf einem Notebook betreiben. Das macht aber insofern noch keinen Sinn, weil die Replikation zwischen verschiedenen TFS-Instanzen auch in TFS 2010 noch nicht unterstützt wird.

Das Schlüsselwort für vereinfachte Installation in TFS ist der neue TFS-Installationsassistent mit den Optionen *Basic*, *Standard*, *Advanced* und *Upgrade*. Eine Basisinstallation benötigt als Betriebssystem keinen Windows Server, vielmehr reicht hier auch ein aktuelles Client-Betriebssystem (Vista oder Windows 7). Als Datenbank kann SQL Server Express zum Einsatz kommen. Nun kann TFS 2010, anders als zuvor, auch auf einem Domänencontroller installiert werden – eine Option für kleine Unternehmen, die nur einen Server besitzen.

Die Basisinstallation richtet sich dabei an Einzelentwickler oder sehr kleine Teams. Bei den Funktionen werden Quellcodeverwaltung, Aufgabenverwaltung und Continuous Integration sowie das Webportal angeboten, nicht verfügbar ist hingegen die Dokumentenverwaltung über SharePoint. *Standard* und *Advanced* unterscheiden sich dann nicht mehr hinsichtlich der Funktionen, sondern hinsichtlich der Individualität der Konfiguration. Die Codebasis bei dem TFS-Installationsassistenten ist aber die gleiche wie bei der Individualinstallation; eine Aufrüstung der fehlenden Funktionen ist also möglich.

Und noch eine Köder legt Microsoft mit TFS 2010 aus: Es gibt nun als kostenlose Beigabe auch die von der Firma SourceGear übernommene TeamPrise-Produktpalette. TeamPrise bietet Unix- /Linux- /Mac-Clients und Eclipse-Plug-Ins für den Team Foundation Server (TFS). Damit wird der TFS auch dann interessant, wenn man noch andere Entwicklungsplattformen außer .NET und Visual Studio im Einsatz hat.

Mit den Neuerungen sind aber nicht alle lizenz- und konfigurationstechnischen Herausforderungen gelöst. Weiterhin brauchen auch Anwender, die nur Tätigkeitsverwaltung über das Webportal machen wollen (z.B. Endkunden, die Fehler melden wollen), eine Client-Zugriffslizenz für den TFS. Und auch sind Installation und Betrieb eines TFS nicht so trivial, dass man hier nicht viel Know-How erwerben müsste. Das Hosting des TFS bei einem Provider (vgl. www.tfshosting.de) kann hier eine Alternative sein.

Erweiterbarkeit

Microsoft bietet schon seit Langem zahlreiche Schnittstellen in Visual Studio an, die Drittanbietern die Erweiterung von Visual Studio ermöglichen. Die Produktvielfalt in der *Visual Studio Gallery* [<http://visualstudiogallery.msdn.microsoft.com/en-us/>] und auch im Quellcodeportal Codeplex [<http://www.codeplex.com/>] sind Ausdruck davon.

Extension Manager

Neben den altbekannten Add-Ins unterstützt Visual Studio 2010 auch eine neue Form der Erweiterungen basierend auf dem Managed Extensibility Framework (MEF). Sie zeichnen sich dadurch aus, dass die Erweiterungen ohne Registrierung installierbar sind; man muss sie nur noch in ein bestimmtes Verzeichnis legen. Der Extension Manager im Menü *Tools* dient dazu, direkt auf die Visual Studio Gallery zuzugreifen.

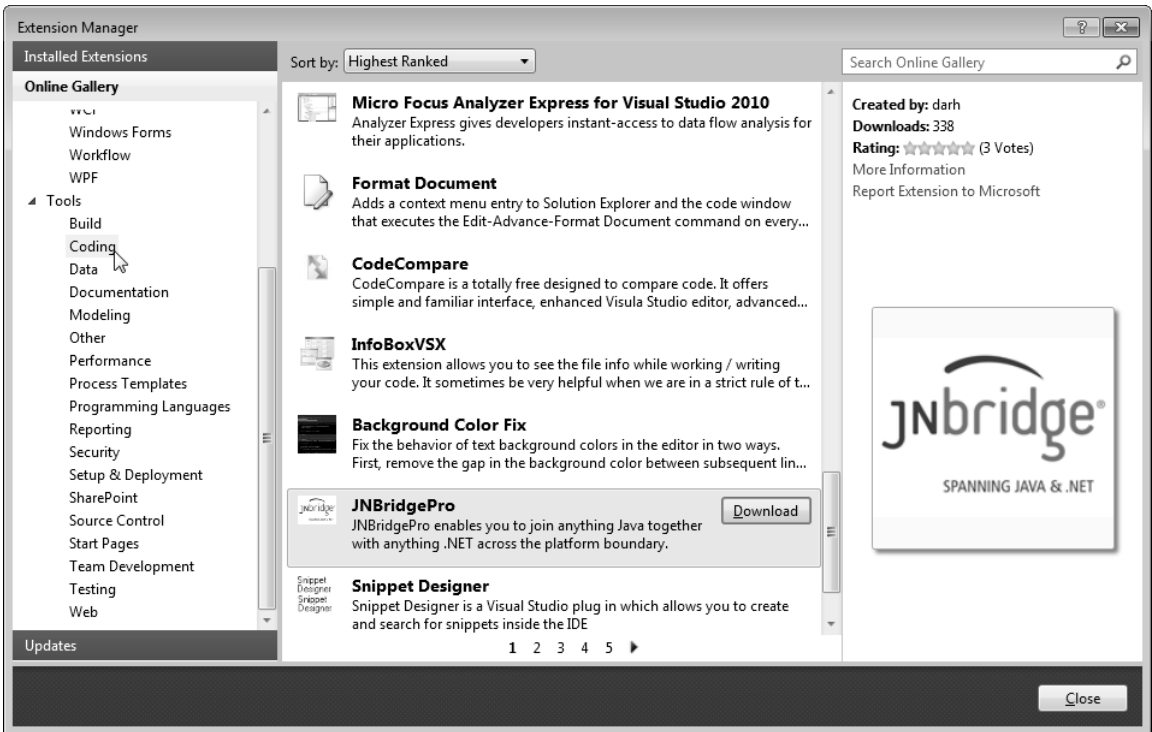


Abbildung 5.131 Extension Manager in Visual Studio 2010

TIPP

Eine Installierte Erweiterung kann man temporär deaktivieren oder dauerhaft deinstallieren, indem man sie unter *Installed Extensions* sucht und dann *Disable* oder *Uninstall* wählt.

HINWEIS

Das in Visual Studio 2010 neu eingeführte Format zur Erstellung von Visual Studio-Erweiterungen nennt Microsoft Visual Studio Integration Extension (VSIX). VSIX ist die Voraussetzung zur Veröffentlichung von Erweiterungen in der Visual Studio Code Gallery [MSDN38], damit die Benutzer dann die Erweiterungen direkt über den Extension Manager in Visual Studio installieren können. VSIX kann man aber auch direkt herunterladen und ausführen.

Eine VSIX-Datei ist eine ZIP-Datei gemäß der Open Packaging Conventions (OPC) mit Binärdateien und einem XML-basierten Manifest. Das VSIX-Format ist im Visual Studio SDK [MSDN31] beschrieben.

Name	Date modified	Type
_rels	13.06.2010 12:24	File folder
Logging	13.06.2010 12:24	File folder
package	13.06.2010 12:24	File folder
ProjectTemplates	13.06.2010 12:24	File folder
Resources	13.06.2010 12:24	File folder
Server%20Install%20Scripts	13.06.2010 12:24	File folder
Templates	13.06.2010 12:24	File folder
[Content_Types].xml	25.05.2010 15:12	XML Document
EULA.rtf	25.05.2010 15:12	Rich Text Format
extension.vsixmanifest	25.05.2010 15:12	VSDXMANIFEST File
Microsoft.VisualStudio.ArchitectureTools.Adapters.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.Extensibility.Layer.Authoring.Templates.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.Extensibility.Uml.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.ReverseEngineering.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.Shell.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.Shell.pkgdef	25.05.2010 15:12	PKGDEF File
Microsoft.VisualStudio.ArchitectureTools.TextTransformation.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.WebProvider.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.WebProvider.pkgdef	25.05.2010 15:12	PKGDEF File
Microsoft.VisualStudio.ArchitectureTools.WorkItemLinking.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.WorkItemLinking.pkgdef	25.05.2010 15:12	PKGDEF File
Microsoft.VisualStudio.ArchitectureTools.WorkItemLinking.ServerRegistration.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.Xml.Import.dll	25.05.2010 15:12	Application extension
Microsoft.VisualStudio.ArchitectureTools.Xml.Import.pkgdef	25.05.2010 15:12	PKGDEF File

Abbildung 5.132 Inhalt einer VSIX-Datei (am Beispiel des Visual Studio 2010 Visualization and Modeling Feature Pack)

Erweiterungsstrategie bei Microsoft

Immer häufiger in den letzten Jahren hat Microsoft zwischen Produktversionen weitere kostenlose Zusatzwerkzeuge veröffentlicht, von denen aber nicht immer klar war, welchen Status und welche Perspektive sie haben, in der kommenden Version enthalten zu sein. Es gab Veröffentlichungen von Zusatzwerkzeugen für .NET und Visual Studio bei Microsoft Download Center [<http://www.microsoft.com/downloads>], Codeplex [<http://www.codeplex.com>], der MSDN Code Gallery [<http://code.msdn.microsoft.com/>], der Visual Studio Gallery [<http://visualstudiogallery.msdn.microsoft.com/en-us/>], Microsoft LiveLabs [<http://livelabs.com/>] und nicht zuletzt auf Community-Portalen wie www.asp.net, windowsclient.net, iis.net und silverlight.net. Zudem verwendet Microsoft sehr unterschiedliche Begriffe für diese Zusätze, z.B. SDK, Toolkit, Starter Kit, Power Tools (früher auch Power Toy), Power Pack, Feature Pack, API Pack oder einfach Library.

Qualitätsstufen

Im Zuge von Visual Studio 2010 hat das Visual Studio-Produktteam angekündigt [HDC01], eine klarere Linie zu fahren und nun vier Stufen zu unterscheiden:

- *Produktveröffentlichungen* (Release to Manufacturing, RTM) ist eine ausführlich getestete Version, es gibt Lokalisierung in verschiedene Sprachen, Produktunterstützung und Service Packs.
- *Feature Packs* sind Ergänzungen, die ebenfalls getestet, lokalisiert und durch den Microsoft Support unterstützt sind. Die Wahrscheinlichkeit, dass sie in der nächsten RTM-Version direkt enthalten sein werden, ist hoch. Feature Packs sollen keine bestehenden Funktionen des Produkts ändern, sondern nur neue Funktionen hinzufügen. Wichtig ist auch, dass man ein MSDN-Abonnement braucht, um ein Feature-Pack zu bekommen (vgl. [BHARRY01]).

- *Power Tools* sind ebenfalls Ergänzungen, die aber nicht so ausführlich getestet sind. Es gibt möglicherweise keine Lokalisierungen. Und es gibt keinen Produktsupport von Microsoft, sondern allenfalls Webforen ohne Garantie auf eine Antwort. Die Funktionen aus *Power Tools* werden möglicherweise in der kommenden Version erscheinen, möglicherweise aber auch in einer späteren Version oder gar nicht. Auch *Power Tools* sollen eigentlich nur Ergänzungen und keine Änderungen enthalten.
- In der letzten Stufe spricht Microsoft dann noch von *Beispielen*, die (Zitat) »unterschiedliche Qualitäts- und Vollständigkeitsgrade« haben. Die o.g. Ankündigung nennt als einen für solche Beispiele verwendeten Begriff nur *Starter Kits*. Die in letzter Zeit bei Microsoft sehr inflationär verwendeten *Toolkits* gehören aber sicherlich auch in diese Gruppe.