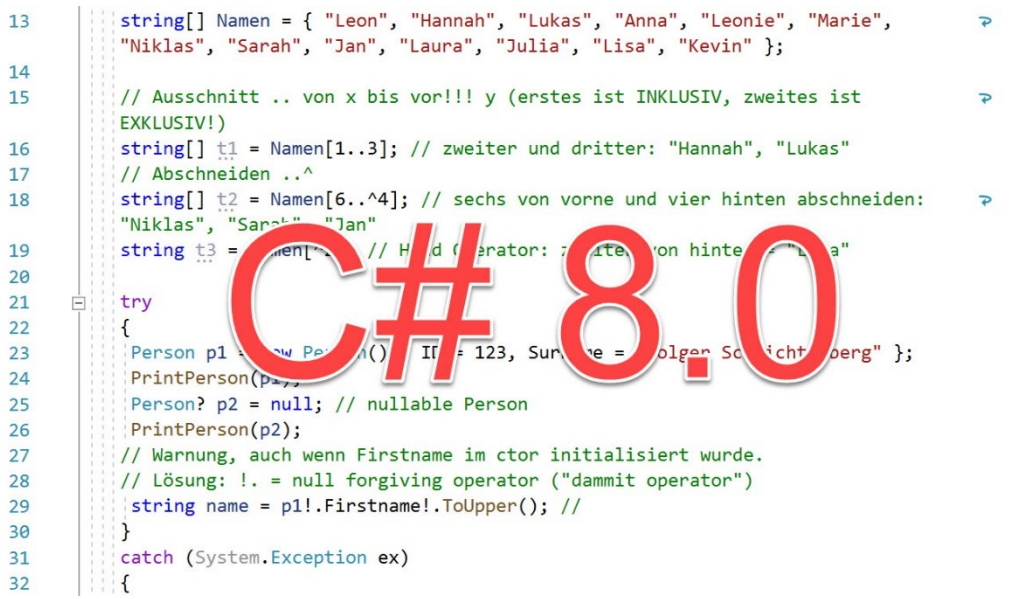
Dr. Holger Schwichtenberg

C# 8.0 Crashkurs

Die Syntax der Programmiersprache C#
für die Softwareentwicklung
in .NET Framework, .NET Core, Xamarin und Mono



www.IT-Visions.de®
Dr. Holger Schwichtenberg

Verlag:	www.IT-Visions.de, Fahrenberg 40b, D-45257 Essen
ISBN:	3934279-32-5
Version:	3.0 Preview / 11.09.2019
Sprachliche Korrektur:	Heike Rickert, Katrin Lettau und Matthias Bloch
Formatierung:	Katrin Lettau
Bezugsquelle:	https://www.amazon.de/exec/obidos/ASIN/3934279317/itvisions-21
Bezugsquelle:	https://leanpub.com/CSharp8

1 Inhaltsverzeichnis

2 Vorwort

Liebe Leserinnen und Leser,

der "C# Crashkurs" ist ein prägnanter Überblick über die Syntax der Programmiersprache C# in der aktuellen Version 8.0.

Dieses Buch ist geeignet für Softwareentwickler, die von einer anderen objektorientierten Programmiersprache (z.B. C++, Java, Visual Basic .NET oder PHP) auf C# umsteigen wollen oder bereits C# einsetzen und ihr Wissen erweitern insbesondere die neusten Sprachfeatures kennenlernen wollen. Wir schulen bei www.IT-Visions.de jedes Jahr hunderte Entwickler auf C# bzw. die neuste Version der Sprache um. Da es viele Umsteiger von Visual Basic .NET zu C# gibt, werden hier die Unterschiede von C# gegenüber Visual Basic .NET an einigen Stellen hervorgehoben.

Für Neueinsteiger, die mit C# erstmals überhaupt eine objektorientiere Programmiersprache erlernen wollen, ist dieses Werk hingegen nicht geeignet, denn es werden die OO-Grundkonzepte nicht erklärt, da die meisten Softwareentwickler heutzutage diese aus anderen Sprachen kennen und das Buch nicht mit diesen Grundlagen unnötig in die Länge gezogen werden soll.

Es erhebt nicht den Anspruch, alle syntaktischen Details zu C# aufzuzeigen, sondern nur die in der Praxis am wichtigsten Konstrukte.

In diesem Buch werden bewusst alle Syntaxbeispiele anhand von Konsolenanwendungen gezeigt. So brauchen Sie als Leser kein Wissen über irgendeine GUI-Bibliothek und die Beispiele sind prägnant fokussiert auf die Syntax.

Dieses Buch wird vertrieben über Amazon.de

- Kindle-E-Book von Amazon.de für 9,99 Euro (der Autor erhält 5,56 Euro):
  www.amazon.de/exec/obidos/ASIN/B07G2STYMH/itvisions-21
- Gedruckt (Print-on-Demand) bei Amazon.de für 14,99 Euro (der Autor erhält 5,53 Euro):
  www.amazon.de/exec/obidos/ASIN/3934279325/itvisions-21
- PDF bei leanpub.com für 10,99 Dollar (der Autor erhält ca. 8,93 Euro):
  www.leanpub.com/CSharp8

Da solch niedrige Preise leider nicht nennenswert dazu beitragen können, den Lebensunterhalt meiner Familie zu bestreiten, ist dieses Projekt ein Hobby. Dementsprechend kann ich nicht garantieren, wann es Updates zu diesem Buch geben wird. Ich werde dann an diesem Buch arbeiten, wenn ich neben meinem Beruf als Softwarearchitekt, Berater und Dozent und meinen sportlichen Betätigungen noch etwas Zeit für das Fachbuchautorenhobby übrig habe.

Zudem möchte ich darauf hinweisen, dass ich natürlich keinen kostenfreien technischen Support zu den Inhalten dieses Buchs geben kann. Ich freue mich aber immer über konstruktives Feedback und Verbesserungsvorschläge. Bitte verwenden Sie dazu das Kontaktformular auf *www.dotnet-doktor.de*.

Wenn Sie **technische Hilfe** zu C# und seinen Einsatzgebieten (.NET, Mono, Xamarin) oder anderen Themen rund um Visual Studio, Windows oder andere Microsoft-Produkte benötigen, stehe ich Ihnen im Rahmen meiner beruflichen Tätigkeit für die Firmen *www.IT-Visions.de* (Beratung, Schulung, Support) und 5Minds IT-Solutions GmbH & Co KG (Softwareentwicklung, siehe *www.5minds.de*) gerne zur Verfügung. Bitte wenden Sie sich für ein Angebot an das jeweilige Kundenteam.

Auf der von mir ehrenamtlich betriebenen **Leser-Website** unter *www.IT-Visions.de/Leser*, können Sie die Beispiele zu diesem Buch herunterladen. Dort müssen Sie sich registrieren. Bei der Registrierung wird ein Losungswort abgefragt. Bitte geben Sie dort **The Expanse** ein.

Herzliche Grüße aus Essen, dem Herzen der Metropole Ruhrgebiet

Holger Schwichtenberg

3 Über den Autor

- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Gebiet komponentenbasierter Softwareentwicklung
- Seit 1996 selbstständig als unabhängiger Berater, Dozent, Softwarearchitekt und Fachjournalist
- Fachlicher Leiter des Berater- und Dozententeams bei *www.IT-Visions.de*
- Leitung der Softwareentwicklung im Bereich Microsoft/.NET bei der 5Minds IT-Solutions GmbH & Co. KG (*www.5Minds.de*)
- Über 65 Fachbücher beim Carl Hanser Verlag, bei O'Reilly, Microsoft Press, APress und Addison-Wesley sowie mehr als 1000 Beiträge in Fachzeitschriften
- Gutachter in den Wettbewerbsverfahren der EU gegen Microsoft (2006-2009)
- Ständiger Mitarbeiter der Zeitschriften iX (seit 1999), dotnetpro (seit 2000) und Windows Developer (seit 2010) sowie beim Online-Portal heise.de (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z.B. Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, BASTA, BASTA-on-Tour, .NET Architecture Camp, Advanced Developers Conference, Developer Week, OOP, DOTNET Cologne, MD DevDays, Community in Motion, DOTNET-Konferenz, VS One, NRW.Conf, Net.Object Days, Windows Forum, Container Conf)
- Zertifikate und Auszeichnungen von Microsoft:
  - Microsoft Most Valuable Professional (MVP)
  - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
  - Softwarearchitektur, mehrschichtige Softwareentwicklung, Softwarekomponenten, SOA
  - Visual Studio, Continous Integration, Continous Delivery, Azure DevOps
  - Microsoft .NET Framework, .NET Core, C#, Visual Basic
  - .NET-Architektur/Auswahl von .NET-Technologien
  - Einführung von .NET Framework/.NET Core und Visual Studio/Migration auf .NET
  - Webanwendungsentwicklung und Cross-Plattform-Anwendungen mit HTML, ASP.NET, JavaScript/TypeScript und Webframeworks wie Angular
  - Enterprise .NET, verteilte Systeme/Webservices mit .NET, insbesondere Windows Communication Foundation und WebAPI
  - Relationale Datenbanken, XML, Datenzugriffsstrategien
  - Objektrelationales Mapping (ORM), insbesondere ADO.NET Entity Framework und EF Core
  - Windows PowerShell, PowerShell Core und Windows Management Instrumentation (WMI)
- Ehrenamtliche Community-Tätigkeiten:
  - Vortragender für die International .NET Association (INETA)
  - Betrieb diverser Community-Websites: www.dotnet-lexikon.de, www.dotnetframework.de, www.windows-scripting.de, www.aspnetdev.de u. a.
- Firmenwebsites: *http://www.IT-Visions.de* und *http://www.5Minds.de*
- Weblog: *http://www.dotnet-doktor.de*
- Kontakt für geschäftliche Anfragen via Kundenteam:
  E-Mail **kundenteam@IT-Visions.de** sowie Telefon **0201 / 64 95 90 - 0**
- Kontakt für Feedback zu diesem Buch: Kontaktformular auf *http://www.dotnet-doktor.de*

www.IT-Visions.de
Dr. Holger Schwichtenberg

5Minds
IT-SOLUTIONS

4 Über dieses Buch

4.1 Versionsgeschichte dieses Buchs

Die folgende Tabelle zeigt die Versionen, die von diesem Buch erschienen sind, sowie die darin besprochenen C#-Versionen.

Ergänzungen der Versionsnummer an der dritten Stelle (z.B. 1.2.3) sind kleine Korrekturen im Buch, die nicht explizit in dieser Versionstabelle erscheinen.

Buchversion Datum	**Umfang**	**Preis Kindle-Ausgabe**	**Preis gedruckte Ausgabe**	**C#-Version**	**Bemerkung**
1.0 27.03.2018	166 Seiten	9,99 €	14,99 €	7.2	▪ Grundversion
1.1 20.07.2018	167 Seiten	9,99 €	14,99 €	7.2 (7.3)	▪ Ref Local Reassignment (C# 7.3) ▪ Ausblick auf C# 8.0
2.0 21.07.2018	172 Seiten	9,99 €	14,99 €	7.3 (8.0)	▪ Vergleich mit Tupeln (C# 7.3) ▪ Annotationen für Backing Field von Auto-Properties (C# 7.3) ▪ Verbesserungen für unsafe-Blöcke (C# 7.3)
2.1 27.11.2018	189 Seiten	9,99 €	14,99 €	7.3 (8.0)	▪ Kapitel "Grundkonzepte von C#" erweitert ▪ Kapitel "Attribute (Fields und Properties)" erweitert ▪ Kapitel "Ereignisse" überarbeitet ▪ Kapitel "Funktionale Programmierung in C#" erweitert ▪ Kapitel "Behandlung von null" ergänzt
3.0 11.09.2019	214 Seiten	9,99 €	19,99 €	8.0	▪ Neues Kapitel "Operatoren/Ranges und Indexe" ▪ Kapitel "Verzweigungen"

					überarbeitet, "Switch Expressions" ergänzt ▪ Kapitel "Schnittstellen" überarbeitet, ergänzt "Standardimplementierungen in Schnittstellen" ▪ Kleinere neue Funktionen von C# 8.0 an verschiedenen Stellen eingebaut ▪ Neues Kapitel "IDisposable/ Using-Blöcke" ▪ Neues Kapitel "Strukturen/Strukturen mit Readonly" ▪ Neues Kapitel "Strukturen/Readonly für einzelne Mitglieder einer Struktur" ▪ Kapitel "Behandlung von null/Nullable Reference Types" aktualisiert.

4.2 Geplante Themen

Folgende Themen sind für kommenden Ausgaben dieses Buchs geplant:

- Span<T> / Memory<T> (C# 7.2)
- Strukturen auf dem Stack (ref struct) in C# 7.2
- Unmanaged Constructed Types (C# 8.0)
- Aliase für referenzierte Assemblies
- Indexer
- Implicit Cast Operator [https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/keywords/implicit]
- Async Streams (C# 8.0)
- Design Pattern in C#
- Clean Code-Programmierung mit C#

4.3 Programmcodebeispiele zu diesem Buch

Die Programmcodebeispiele zu diesem Buch können Sie auf der auf der von mir ehrenamtlich betriebenen Leserwebsite *www.IT-Visions.de/Leser* herunterladen. Dort müssen Sie sich registrieren. Bei der Registrierung wird ein Losungswort abgefragt. Bitte geben Sie dort das Losungswort **The Expanse** ein.

Alle Programmbeispiele aus diesem Buch sind in einer Visual Studio 2019-Projektmappe mit zwei Projekten enthalten. Es muss seit C# 8.0 zwei Projekte geben, weil einige Sprachfeatures nicht mehr im .NET Framework laufen. Die beiden Projekte enthalten:

- CSharpSprachsyntax (.NET Framework): Alle Sprachfeatures von C# 1.0 bis 7.3 und solche von C# 8.0, die auf .NET Framework laufen
- CSharpSprachsyntaxNETCore (.NET Core): Alle Sprachfeatures von C# 8.0, die NICHT auf .NET Framework laufen

Die Beispiele sind in Unterordnern nach Sprachversionen aufgeteilt. Dies heißt, dass Sie zum Beispiel Sprachfeatures von C# 8.0 im Ordner CS80 finden.

Wie im Vorwort bereits erwähnt handelt es sich um den Anwendungstyp "Konsolenanwendung". So brauchen Sie als Leser kein Wissen über irgendeine GUI-Bibliohek und die Beispiele sind prägnant fokussiert auf die Syntax.

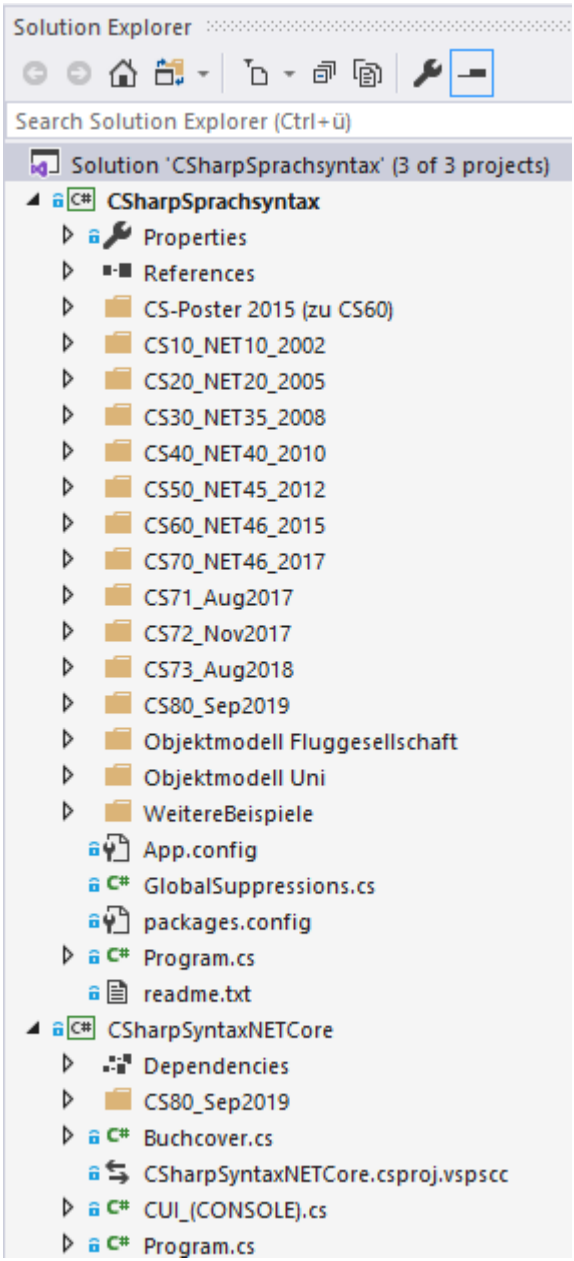


Abbildung: Programmcodebeispiele zu diesem Buch in zwei Visual Studio-Projekten

5 Fakten zu C#

5.1 Der Name C#

C# wird gesprochen „C Sharp“. Das # könnte man auch in ein vierfaches Pluszeichen aufspalten (also C++++, eine Weiterentwicklung von C++). Ursprünglich sollte die Sprache "Cool" heißen. Eine Zeit lang wurde auch "C# .NET" verwendet; das ist heute aber nicht mehr üblich. Microsoft spricht aber gelegentlich noch von "Visual C#", z.B. meldet sich der Kommandozeilencompiler von C# auch in der aktuellen Version mit "Microsoft (R) Visual C# Compiler".

5.2 Ursprünge von C#

C# ist das Ergebnis eines Projektes bei Microsoft, welches im Dezember 1998 gestartet wurde, nachdem die Firma Sun Microsoft die Veränderung der von Sun entwickelten Programmiersprache Java verboten hatte. Vater von C# ist Anders Heljsberg [https://de.wikipedia.org/wiki/Anders_Hejlsberg], der zuvor auch Turbo Pascal und Borland Delphi erschaffen hat.Er war früher bei Borland und arbeitet seit 1996 bei Microsoft. Heutzutage ist er auch verantwortlich für die Sprache TypeScript.

5.3 Status der Programmiersprache C#

Früher gab es einen wahren Glaubenskrieg in der .NET-Entwicklergemeinde um die Wahl der »richtigen« Programmiersprache. C# oder Visual Basic .NET hieß die Frage, die viele Projektteams bewegt hat. Auch wenn Visual Basic .NET in allen wesentlichen Punkten syntaktisch ebenbürtig war, hat C# klar gewonnen.

C# ist heute nicht nur eine von vielen Programmiersprachen für .NET, es hat sich durchgesetzt als DIE Programmiersprache für .NET. Gegenwärtig gibt es nur noch wenige .NET-Projekte, die Visual Basic .NET, F# oder C++/CLI oder exotischere Sprachen verwenden.

Schaut man in die aktuelle Dokumentation der .NET-Klassen auf *https://docs.microsoft.com*, sieht man dort nur noch Beispiele für C#, während die alte MSDN-Dokumentation noch Beispiele in C#, Visual Basic .NET, und C++ enthielt.

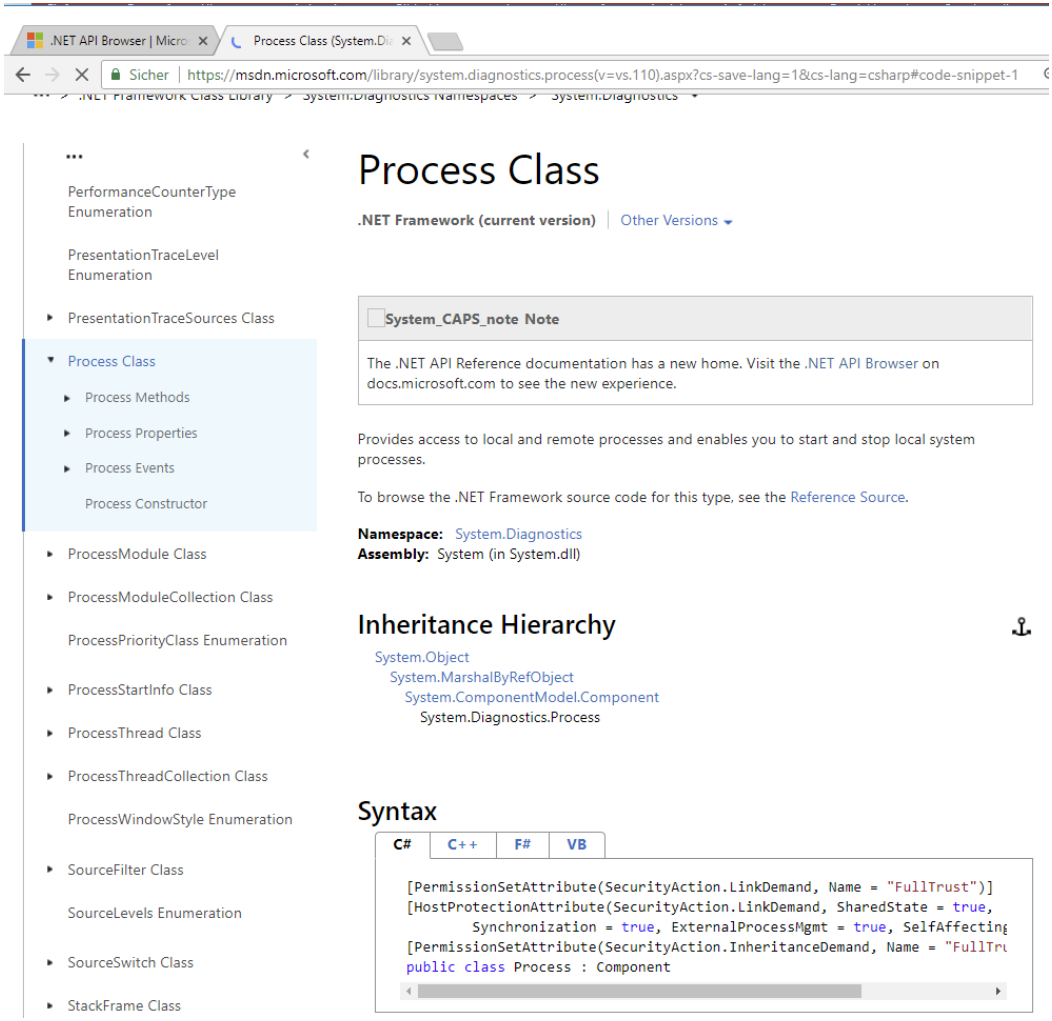


Abbildung: Beispiele in vier Sprachen in der alten MSDN-Dokumentation der .NET-Klassen

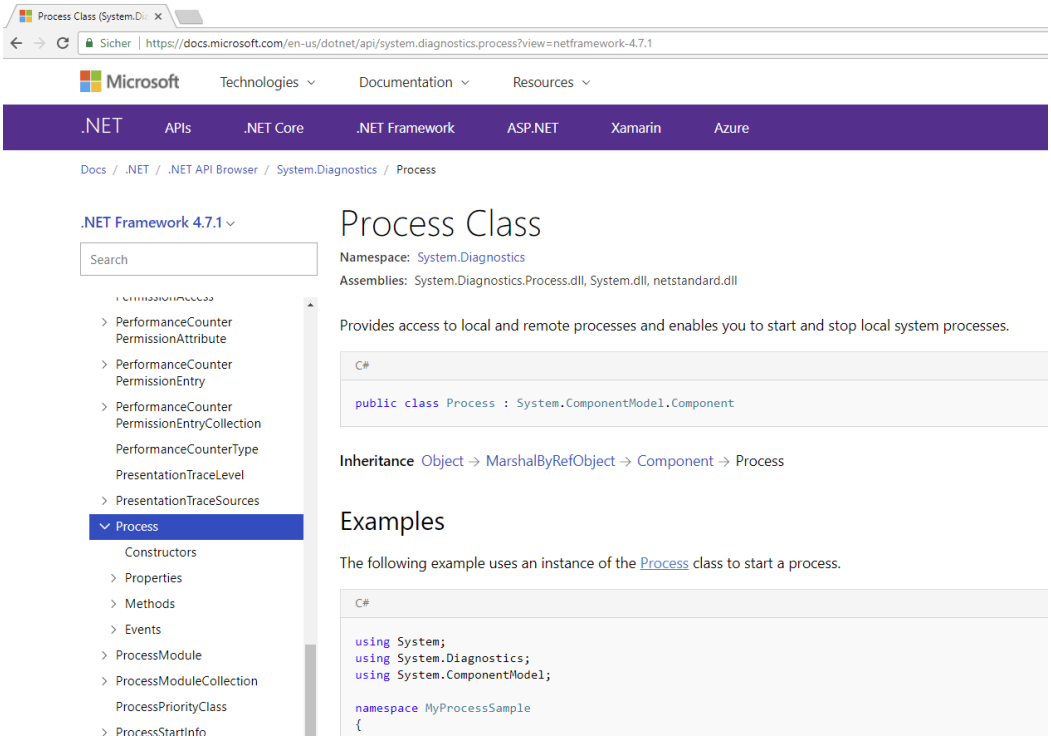


Abbildung: In der neuen .NET-Klassendokumentation gibt es nur noch Beispiele in C#

5.4 Versionsgeschichte

Hinsichtlich der Versionsnummern der Sprache C# herrschte früher etwas Verwirrung. Es gab einerseits eine offizielle Zählung mit Versionsnummer (parallel zum .NET Framework), andererseits mit Jahreszahlen (parallel zu Visual Studio). Intern wird eine dritte Zählung für den Compiler verwendet. Die erste Version von C# im Rahmen des .NET Framework 1.0 trug intern die Versionsnummer 7.0. Zu .NET 1.1 gab es dann C# 7.1, im .NET Framework 2.0 und 3.0 meldet sich der C#-Compiler mit Version 8.0. Ab .NET Framework 3.5 hat Microsoft dies aber bereinigt. Dort meldet sich der Compiler nun auch mit Version 3.5.

Die folgende Liste dokumentiert die Versionsgeschichte von C# einschließlich der verschiedenen Namen, die es jeweils gibt.

- C# 1.0 ist erschienen am 05.01.2002 (in Visual Studio.NET 2002+2003 / .NET Framework 1.0 und 1.1. Erste Version des ISO-Standards für C#.)
- C# 2.0 ist erschienen am 07.11.2005 (C# 2005 / in Visual Studio.NET 2005 / .NET Framework 2.0 und 3.0. Zweite Version des ISO-Standards für C#.)
- C# 3.0 ist erschienen am 15.08.2008 (C# 2008 / in Visual Studio.NET 2008 / .NET Framework 3.5)
- C# 4.0 ist erschienen am 12.04.2010 (C# 2010 / in Visual Studio.NET 2010 / .NET Framework 4.0)
- C# 5.0 ist erschienen am 12.08.2012 (C# 2012 / in Visual Studio.NET 2012 / .NET Framework 4.5)

- C# 6.0 ist erschienen am 20.07.2015 (C# 2015 / in Visual Studio.NET 2015 / .NET Framework 4.6)
- C# 7.0 ist erschienen am 05.03.2017 (C# 2017 / in Visual Studio 2017)
- C# 7.1 ist erschienen am 14.08.2017 (in Visual Studio 15.3)
- C# 7.2 ist erschienen am 15.11.2017 (in Visual Studio 15.5)
- C# 7.2 ist erschienen am 02.08.2018 (in Visual Studio 15.7)
- C# 8.0 ist erschienen am 23.09.2019 (in Visual Studio 16.3)

Version der Sprachsyntax mit Versionsnummer	**Ausgeliefert mit**	**Version der Sprachsyntax mit Jahreszahl**	**Interne Versionsnummer des C#-Compilers**
C# 1.0	.NET Framework 1.0	Visual C# 2002	7.0 (alter Compiler)
C# 1.1	.NET Framework 1.1	Visual C# 2003	7.1 (alter Compiler)
C# 2.0	.NET Framework 2.0	Visual C# 2005	8.0 (alter Compiler)
C# 2.0	.NET Framework 3.0	Visual C# 2005	8.0 (alter Compiler)
C# 3.0	.NET Framework 3.5	Visual C# 2008	3.5 (alter Compiler)
C# 4.0	.NET Framework 4.0	Visual C# 2010	4.0 (alter Compiler)
C# 5.0	.NET Framework 4.5	Visual C# 2012	4.5 (alter Compiler)
C# 6.0	.NET Framework 4.6 / .NET Core 1.0	Visual C# 2015	1.x (Neuer Compiler)
C# 7.0	Visual Studio 2017 15.0 / .NET Core 2.0	Visual C# 2017	2.0 (Neuer Compiler)
C# 7.1	Visual Studio 2017 15.4 / .NET Core 2.0	Visual C# 2017	2.3 (Neuer Compiler)
C# 7.2	Visual Studio 2017 15.5 / .NET Core 2.0	Visual C# 2017	2.7 (Neuer Compiler)
C# 7.3	Visual Studio 2017 15.7 / .NET Core 2.1	Visual C# 2017	2.8 + 2.9 + 2.10 (Neuer Compiler)
C# 8.0 Preview	Visual Studio 2019 16.0 bis 16.2 / .NET Core 3.0 Preview	Visual C# 2018	3.0 + 3.1 + 3.2 (Neuer Compiler)
C# 8.0 RTM	Visual Studio 2019 16.3 / .NET Core 3.0	Visual C# 2018	3.3 (Neuer Compiler)

Tabelle: Verschiedene Versionsnummernzählungen für die Sprache C#

5.5 Standardisierung

Microsoft hat einige Teile des .NET Framework unter dem Namen Common Language Infrastructure (CLI) standardisieren lassen. Die CLI wurde erstmals im Dezember 2001 von der

European Computer Manufacturers Association (ECMA) standardisiert (ECMA-Standard 335, Arbeitsgruppe TC49 / TG3, früher: TC39 / TG3, siehe [ECMA01]); mit kleinen Änderungen wurde der Standard im Dezember 2002 von der weltweit wichtigsten Standardisierungsorganisation, der International Standardization Organization (ISO), übernommen als ISO / IEC 23271.

Die Begriffe lauten in den Standards zum Teil allerdings anders als bei Microsoft: Was im .NET Framework Microsoft Intermediate Language (MSIL) heißt, entspricht im Standard der Common Intermediate Language (CIL). Anstelle der Framework Class Library (FCL) spricht man von der CLI Class Library. Von der Standardisierung ausgenommen sind jedoch z.B. die Datenbankschnittstelle ADO.NET und die Benutzeroberflächen-Bibliotheken Windows Forms und ASP.NET Webforms. Auch die neueren .NET-Bibliotheken (WPF, WCF und WF) sind nicht standardisiert.

Auch die Programmiersprache C# ist von beiden Gremien akzeptiert (ECMA-334 bzw. ISO / IEC 23270). Die Standardisierung bezieht sich aber auf ältere Versionen. Die letzten C#-Versionen hat Microsoft nicht mehr standardisieren lassen. Die Standardisierung ist auf dem Stand C# 2.0

Ein weiterer, von Microsoft initiierter Standard ist von der ECMA im Dezember 2005 unter ECMA-372 (Arbeitsgruppe TC49 / TG5, früher: TC39 / TG5) verabschiedet worden: C++ / CLI ist eine Spracherweiterung für C++ (ISO / IEC 14882:2003), die eine elegantere Nutzung von C++ auf der CLI-Plattform ermöglicht, als dies bisher mit den Managed Extensions for C++ (alias Managed C++) möglich war.

5.6 Implementierung des C#-Compilers

Die ursprüngliche Version des C#-Compilers (csc.exe) wurde in C++ implementiert. Auch der C#-Compiler im Mono-Projekt ist in C++ geschrieben.

Mit dem Projekt "Roslyn" (alias: .NET Compiler Platform) hat Microsoft selbst den Compiler neu in C# implementiert. Die erste Version des neuen Compilers war C# 6.0.

5.7 Open Source

Bereits zu C# 1.0 gab es eine quelloffene Version im Projekt "Rotor" im Rahmen der Standardisierung von C#. Diese war jedoch nicht "Open Source", sondern nur "Shared Source", d.h. der Quellcode durfte betrachtet, aber nicht weiterverwendet werden. Seit C# 6.0 ist der neue Compiler im Rahmen der .NET Compiler Platform "Roslyn" ein Open Source-Projekt auf Github.

Projekt für das Design der Programmiersprache:

https://github.com/dotnet/csharplang

Projekt für die Implementierung der Programmiersprache:

https://github.com/dotnet/roslyn

5.8 Kommende Versionen

Aktuell entwickelt Microsoft an der Version C# 8.0.

5.9 Parität und Co-Evolution mit Visual Basic .NET

Im Jahr 2010 hatte Microsoft verkündet, die Programmiersprache C# und Visual Basic .NET hinsichtlich ihrer Funktionalität anzugleichen. »Die Sprachen sollen sich in Stil und Gefühl unterscheiden, nicht in ihrem Funktionsumfang«, schrieb Mads Torgersen, Produktmanager für C# damals. Scott Wiltamuth führt den Begriff "Co-Evolution" ein [https://blogs.msdn.microsoft.com/scottwil/2010/03/09/vb-and-c-coevolution/].

Einige Jahre hat Microsoft diese Strategie tatsächlich umgesetzt und bestehende Sprachfeatures, die nur eine Sprache hatte, in der anderen Sprache nachgerüstet und neue Sprachfeatures gleichzeitig oder zumindest zeitnah in beiden Sprachen veröffentlicht.

Im Jahr 2017 hat Microsoft sich von Parität und Co-Evolution wieder verabschiedet.

Visual Basic .NET ist nach C# die zweitwichtigste Programmiersprache in der .NET-Welt. Telemetriedaten [https://blogs.msdn.microsoft.com/dotnet/2017/02/01/the-net-language-strategy] von Microsoft zeigen einerseits, dass Visual Basic .NET hauptsächlich zur Programmierung mit älteren .NET-Techniken wie Windows Forms und ASP.NET Webforms zum Einsatz kommt. Andererseits beginnen viele neue .NET-Entwickler mit Visual Basic .NET, bevor sie sich an C# herantrauen. Microsoft nahm diese Erkenntnisse zum Anlass, von der im Jahr 2010 verkündigen Co-Evolutionsstrategie von C# und Visual Basic .NET abzurücken und zukünftig nicht mehr alle neuen C#-Features automatisch auf Visual Basic .NET zu übertragen. Die parallel zu C# 7.0 erschienene Version 15 von Visual Basic .NET bietet daher lediglich Tupel und binäre Literale als neue Sprachfeatures an. Zudem kann Visual Basic .NET 15 C#-Methoden nutzen, die Zeiger mit ref liefern, selbst aber solche Methoden nicht implementieren.

5.10 Neuerungen in C# 8.0

Die wichtigsten Neuerungen in C# 8.0 sind:

- Nullable Reference Types string? !.
- Standardimplementierungen in Schnittstellen (*)
- Index ^ und Range .. ..^ (*)
- Switch Expressions

Weitere Neuerungen in C# 8.0 sind:

- Null Coalescing Assignment ??=
- Alternative für verbatim interpolated Strings: @$ zusätzlich zu $@
- Async Streams (*)
- Static Local Functions
- using-Deklarations ohne Blöcke
- Unmanaged Constructed Types
- Readonly-Mitglieder in einer Struktur
- Dispose() für ref structs (Strukturen auf dem Stack)

(*) erfordert .NET Standard 2.1, d.h. nur für .NET Core, Xamarin, Mono und Unity. Diese Sprachefeatures sind im klassischen .NET Framework nicht verfügbar und Microsoft plant auch nicht, diese dort noch einzubauen.

6 Grundkonzepte von C#

Konzeptionell wurde C# vor allem von C++ und Java beeinflusst; man kann aber auch Parallelen zu Visual Basic und Delphi finden.

6.1 Sprachtypus

Im Gegensatz zu C++, das eine hybride Sprache aus objektorientierten und nicht-objektorientierten Konzepten ist, ist C# ebenso wie Java eine rein objektorientierte Sprache, d.h., alle Datentypen basieren auf Klassen und alle Anweisungen erfolgen in Klassen.

C# unterstützt alle zentralen Konzepte der Objektorientierung einschließlich Schnittstellen, Vererbung und Polymorphismus. Schon in C# 2005 wurde auch die Unterstützung für generische Klassen und partielle Klassen hinzugefügt. Außerdem besitzt C# Konzepte der funktionalen Programmierung (Delegates und Lambda-Ausdrücke).

6.2 Groß- und Kleinschreibung

Ein wesentlicher Unterschied zwischen C# und Visual Basic .NET ist die Tatsache, dass C# im Gegensatz zu Visual Basic .NET zwischen Groß- und Kleinschreibung unterscheidet. Dies gilt sowohl für die Schlüsselwörter der Sprache als auch für alle Bezeichner (a und A sind verschiedene Variablen!). Die Schlüsselwörter der Sprache C# werden komplett in Kleinbuchstaben geschrieben.

6.3 Schlüsselwörter der Sprache

Die folgende Liste zeigt die vordefinierten Schlüsselwörter der Programmiersprache C#. Diese Namen dürfen in der gleichen Groß-/Kleinschreibung nicht als Bezeichner verwendet werden (Quelle: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/index).

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
volatile	while			

6.4 Namensregeln und Namenskonventionen

Bei der Vergabe von eigenen Bezeichner (z.B. Variablenname, Parameternamen, Atributnamen und Methodennamen) gibt es verpflichtende Regeln und optionale Namenskonventionen.

Verpflichtende Regeln sind:

- Der Name darf nur Buchstaben (*), Zahlen und den Unterstrich enthalten.
- Der Name muss mit einem Buchstaben beginnen
- Die Groß- und Kleinschreibung ist relevant
- Es dürfen keine Namen von C#-Schlüsselwörtern verwendet werden.

Hinweis: (*) Umlaute sind erlaubt, aber sollten dennoch besser vermieden werden: Nicht alle Werkzeuge und alle Menschen kommen damit gut klar!

Optionale Regeln hat Microsoft in den ".NET Framework Design Guidelines" [https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines] definiert. Die wichtigsten Regeln dort sind:

- Für die Groß-/Kleinschreibung gilt grundsätzlich **PascalCasing**, d.h. ein Bezeichner beginnt grundsätzlich mit einem Großbuchstaben und jedes weitere Wort innerhalb des Bezeichners beginnt ebenfalls wieder mit einem Großbuchstaben.

  Beispiel: KundenPortalBenutzer

- Ausnahmen gibt es für Abkürzungen, die nur aus zwei Buchstaben bestehen. Diese dürfen komplett in Großbuchstaben geschrieben sein (z.B. UI und IO). Alle anderen Abkürzungen werden entgegen ihrer normalen Schreibweise in Groß-/Kleinschreibung geschrieben (z.B. Xml, Xsd und W3c).

  Beispiele: System.IO.File, System.Xml.XmlDocument

- Lokale Variablen, versteckte Attribute (private/protected) und Parameternamen sollen in **camelCasing** (Bezeichner beginnt mit einem Kleinbuchstaben, aber jedes weitere Wort innerhalb des Bezeichners beginnt mit einem Großbuchstaben) geschrieben werden.

  Beispiel: Login(KundenPortalBenutzer kundenPortalBenutzer)

6.5 Blockbildung und Umbrüche

Blockbildung findet im C / C++-Stil statt, also mit geschweiften Klammern { }. Befehlstrenner ist das Semikolon (;).

Ein Zeilenumbruch kann zwischen den Elementen des Ausdrucks auftreten, ohne das besondere Vorkehrungen getroffen werden müssen. Zahlen können seit C# 7.0 mit einem Unterstrich gegliedert werden; aber man darf innerhalb von Zahlen keinen Zeilenumbruch haben.

```
// Formel ohne Umbrüche
double Ergebnis1 = (2 + 3) * ( 5 + 6) * (7 * 8) + 3.141_592_653_59;

// Formel mit Umbrüchen
double Ergebnis2 = (2 + 3) *
                   (5 + 6) *
                   (7 * 8)
                   + 3.141_592_653_59;
```

6.6 Hello World

Das folgende Listing zeigt das Hello World-Beispiel in C#, das man in jeder Programmiersprache zuerst schreibt.

```
using System;

namespace HalloWelt
{
 class Program
 {
  static void Main(string[] args)
  {
   Console.WriteLine("Hallo Welt!");
  }
 }
}
```

Mariginal komplexer ist diese Variante, die – sofern vorhanden – den ersten übergebenen Kommandozeilenparameter als Name auffasst und die Person mit Namen grüßt.

```
namespace HalloWelt
{
 class Program
 {
  static void Main(string[] args)
  {
   if (args.Length > 0)
   {
    var name = args[0];
    // Ausgabe mit String Interpolation
    Console.WriteLine($"Hallo {name}!");
    Console.ReadLine();
   }
   else
   {
    Console.WriteLine("Hallo Welt!");
   }
  }
 }
}
```

6.7 Eingebaute Funktionen

Anders als in Visual Basic existieren in C# keine eingebauten Funktionen zur Typumwandlung (z.B. CBool(), CInt(), CLng(), CType()), Zeichenkettenverarbeitung (z.B. InStr(), Trim(), LCase()) und Ausgabe (z.B. MsgBox()). Auch die My-Klassenbibliothek ist nicht vorhanden. Grundsätzlich ist es möglich, die in Visual Basic eingebauten Funktionen und die My-Bibliothek durch Referenzierung der Microsoft.VisualBasic.dll auch in C# zu nutzen. Dies sollte jedoch vermieden werden, um sprachunabhängig zu bleiben. Alle Visual Basic-Funktionen und -Objekte sind auch in der .NET-Klassenbibliothek enthalten, z.B. String.IndexOf() statt InStr() und Convert.ToInt32() statt CInt().

7 Der C#-Compiler

Es gibt zwei Varianten des C#-Compilers: eine alte, in C++ geschriebene, und neue, in C# geschriebene Implementierung.

7.1 Der ursprüngliche (alte) C#-Compiler

Der Kommandozeilencompiler für C# im .NET Framework Redistributable ist csc.exe. Er wird installiert im Verzeichnis C:\Windows\Microsoft.NET\Framework64\v4.0.30319. Alternativ kann er in der .NET Framework-Klassenbibliothek im sogenannten "CodeDOM" durch die Klasse Microsoft.CSharp.CSCodeProvider angesprochen werden.

Wenn Sie heute ein aktuelles Microsoft .NET Framework (z.B. 4.7.2) verwenden, so ist dort der ursprüngliche C#-Compiler immer noch in der Version 5.0 enthalten.

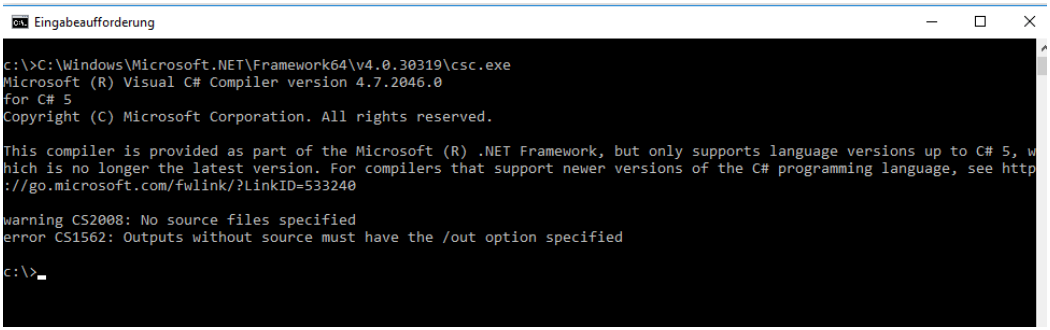


Abbildung: In .NET Framework 4.7.1 ist der C#-Compiler für C# 5.0 enthalten.

7.1.1 Kompilierung mit csc.exe

Der Befehl

```
csc.exe Dateiname1.cs Dateiname2.cs DateinameX.cs
```

oder

```
csc Dateiname1.cs Dateiname2.cs DateinameX.cs
```

übersetzt die angegebenen Dateien in eine Konsolenanwendung. Eine Datei, die als Konsolenanwendung oder Windows-Anwendung kompiliert wird, muss genau eine Klasse mit folgendem Einstiegspunkt besitzen: public static void Main().

Listing: »Hello World« in C#

```
class Hauptprogramm
{
        public static void Main()
        {
                System.Console.WriteLine("Hello World!");
        }
}
```

7.1.2 Kommandozeilenparameter

Der Kommandozeilencompiler bietet zahlreiche Optionen. Die wichtigsten davon sind:

- /target:winexe Der Compiler erzeugt eine Windows-Anwendung
- /target:library Der Compiler erzeugt eine DLL (kein Main() notwendig)