

# Einführung in .NET und C#

---

Fach: SOP  
Informatik-Techniker  
Wirtschaftsinformatiker

WS 2006/07

© 2006 Dagmar Walddobler

1

# Was ist .NET?

---

## Begriffsdeklarationen

<b>.Net-Framework</b>	stellt Anwendungen alle benötigten Dienste zur Verfügung, die wichtigsten: <ul style="list-style-type: none"><li>- eine umfangreiche Klassenbibliothek</li><li>- der CLR (siehe unten)</li><li>- verschiedene Compiler für unterschiedliche Programmiersprachen</li></ul>
<b>IL-Code</b>	die Compiler erzeugen aus dem Quellcode der .NET-Programme einen Zwischencode, den sog. IL-Code (Intermediate Language, Managed Code)
<b>JIT-Compiler</b>	wandelt den IL-Code jeweils einer Funktion in Betriebssystemcode um
<b>NGen.exe</b>	(Native Code Generator) wandelt den IL-Code dauerhaft in nativen Maschinencode um
<b>CLR</b>	(Common Language Runtime) stellt als Laufzeitumgebung die grundlegenden Technologien zur Verfügung und führt .NET-Programme aus
<b>Garbage Collector</b>	läuft als Thread im Hintergrund und beseitigt nicht mehr benötigte Objekte

## Was ist .NET?

---

### Begriffsdeklarationen

#### **XML**

ein Standard zur Speicherung und Verarbeitung von strukturierten Daten in Textform. In .NET werden Daten stets in XML-Format gespeichert oder über ein Netzwerk übertragen.

#### **SOAP**

ist ein textbasiertes Protokoll für den Zugriff auf Objekte in entfernten Komponenten

#### **ASP.NET**

(Active Server Pages) zum Erstellen von Web-Anwendungen

#### **Assemblierung**

enthält den kompilierten Code einer oder mehrerer Klassen und Typen, aller verwendeter Ressourcen (Bitmaps, Icons, etc.), und Metadaten, die die Assemblierung beschreiben.

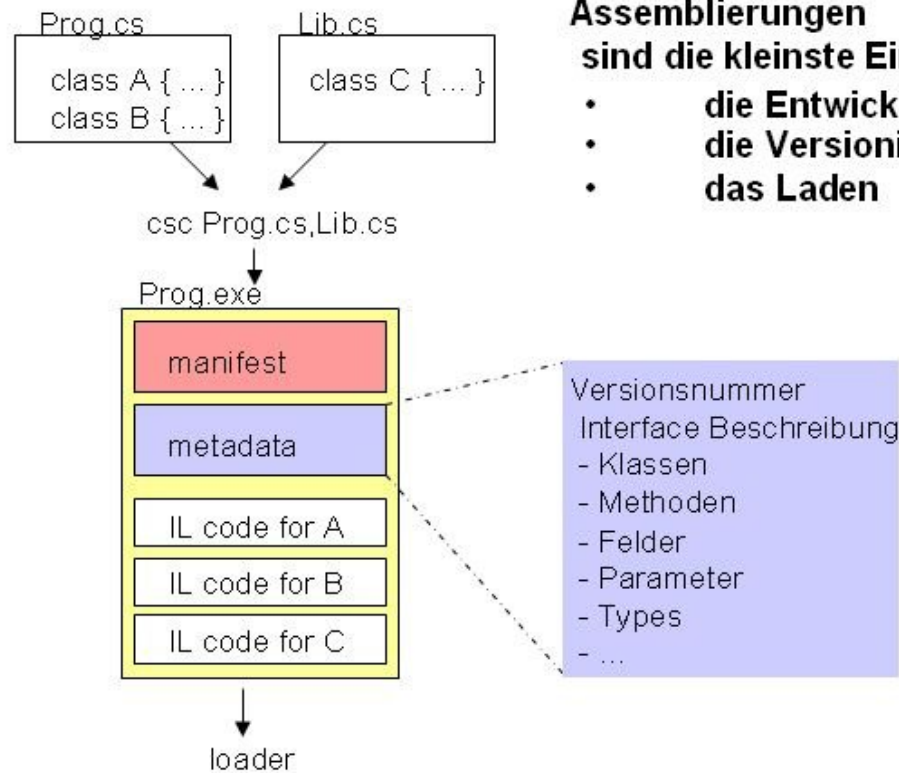
Eine Assemblierung besteht aus mindestens einer kompilierten Datei. Eine einzelne Datei wrd als Modul bezeichnet (siehe unten)

#### **Namensräume**

(Namespaces) ist eine logische Gruppierung aller Typen. Vereinfacht nachfolgend die Schreibweise (**mehr nicht!**) zum Ansprechen von Mitgliedern des eingebundenen Namespaces

# Was ist .NET?

## Assemblierung



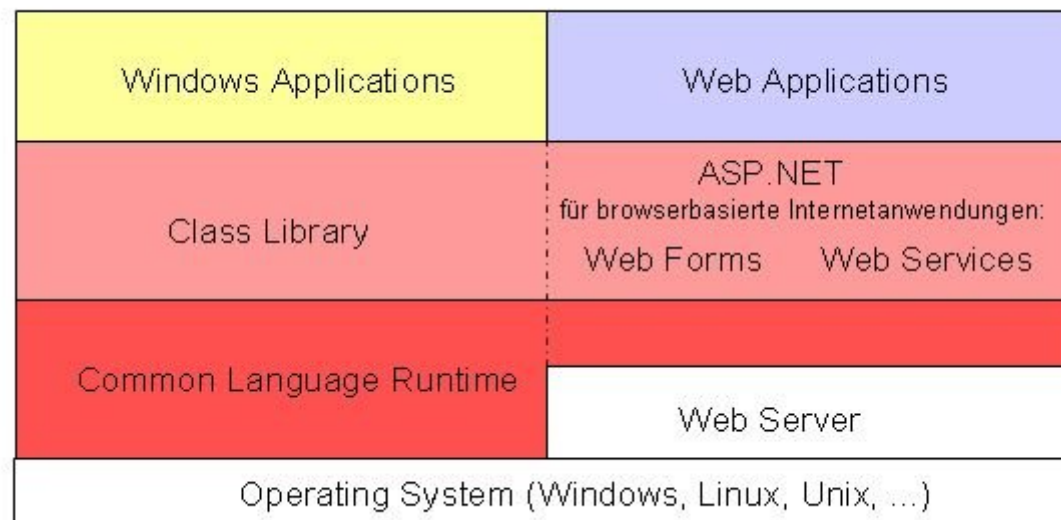
**Assemblierungen sind die kleinste Einheit für**

- **die Entwicklung**
- **die Versionierung**
- **das Laden**

## Was ist .NET?

---

Eine neue Software-Plattform für Desktop- und Web-Anwendungen

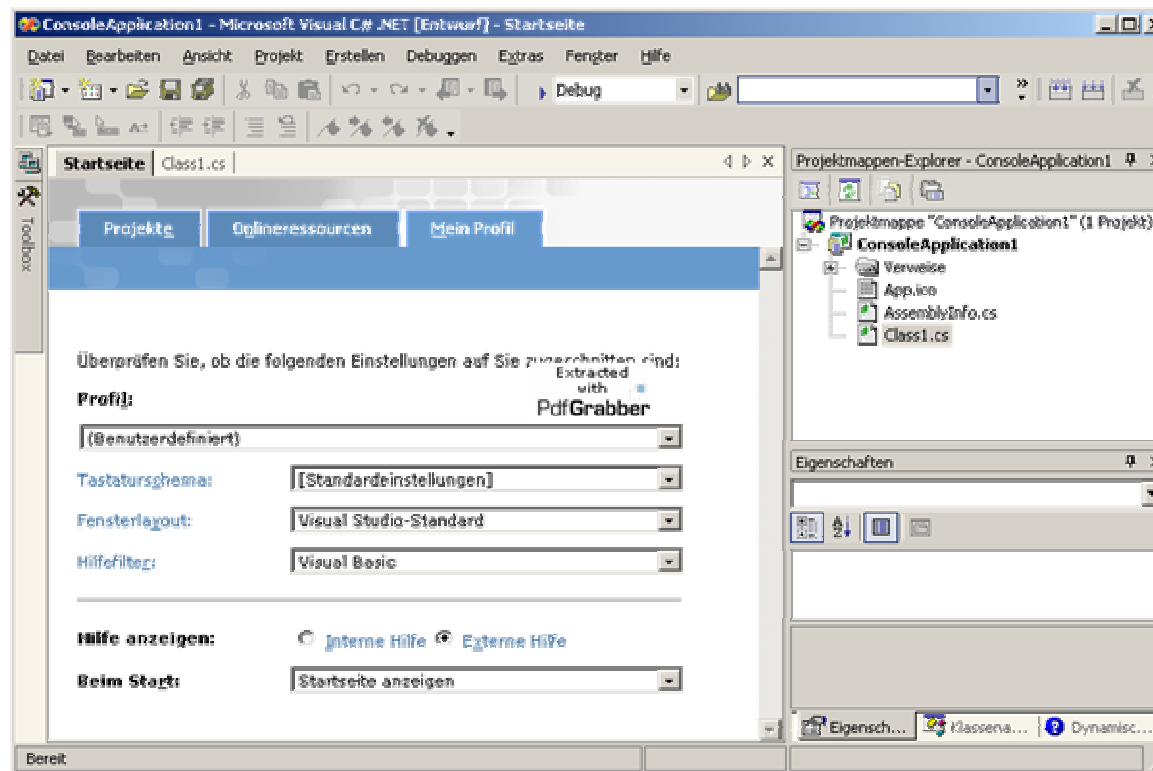


## Das .NET Framework



© 2006 Dagmar Walddobler

## Visual Studio.NET (Screenshot)



© 2006 Dagmar Walddobler

7

# Visual Studio.NET

---

## **I**ntegrated **D**evelopment**E**nvironment (IDE) (Integrierte Entwicklungsumgebung)

Bereitstellung / Integration von Hilfsmitteln jeglicher Art zur (möglichst ganzheitlichen) Unterstützung des gesamten Softwareentwicklungsprozesses.

### **Elementare Leistungsmerkmale:**

Projektverwaltung; Code-Editor (mit IntelliSense); GUI-Designer; Compiler-Steuerung; Debugger; diverse Assistenten und Zusatzwerkzeuge

### **ist das Standardwerkzeug zur Softwareentwicklung unter Windows überhaupt**

VS.NET eignet sich auch zur Entwicklung von **nicht-.NET**-Anwendungen; Plug-In-Konzept zur Einbindung „ganz fremder“ Sprachen und Compiler. (*Beispiel: Fortran*)

### **Visual Studio (.NET) weist selbst eine gewisse Komplexität auf!**

## Ziele des .NET

---

Um die Desktop- und Web-Programmierung zu vereinheitlichen

### Bisher

#### Desktop-Programmierung

objekt-orientiert  
compiliert (C, C++, Fortran, ...)  
Class Library

#### Web-Programmierung

ASP (nicht objekt-orientiert)  
interpretiert (VBScript, Javascript, PHP, ...)  
spezielle Library

### Unter .NET

#### Desktop- und Web-Programmierung

objekt-orientiert (ASP.NET)  
compiliert (C#, C++, VB.NET, Fortran, ...)  
Einheitliche Class Library

## Ziele des .NET

---

### Interoperabilität von Programmiersprachen

bisher

- Millionen von Codezeilen in C++, Fortran, Visual Basic, ...
- sehr begrenzte Interoperabilität

unter .NET

- binäre Kompatibilität unter mehr als 20 Sprachen (C#, C++, Java, Eiffel, Fortran, Cobol, Pascal, VB.NET, Perl, Python, ...)

Class in VB.NET

```
Public Class A
  Public X As Integer
  Public Sub Foo() ...
End Class
```

Subclass in C#

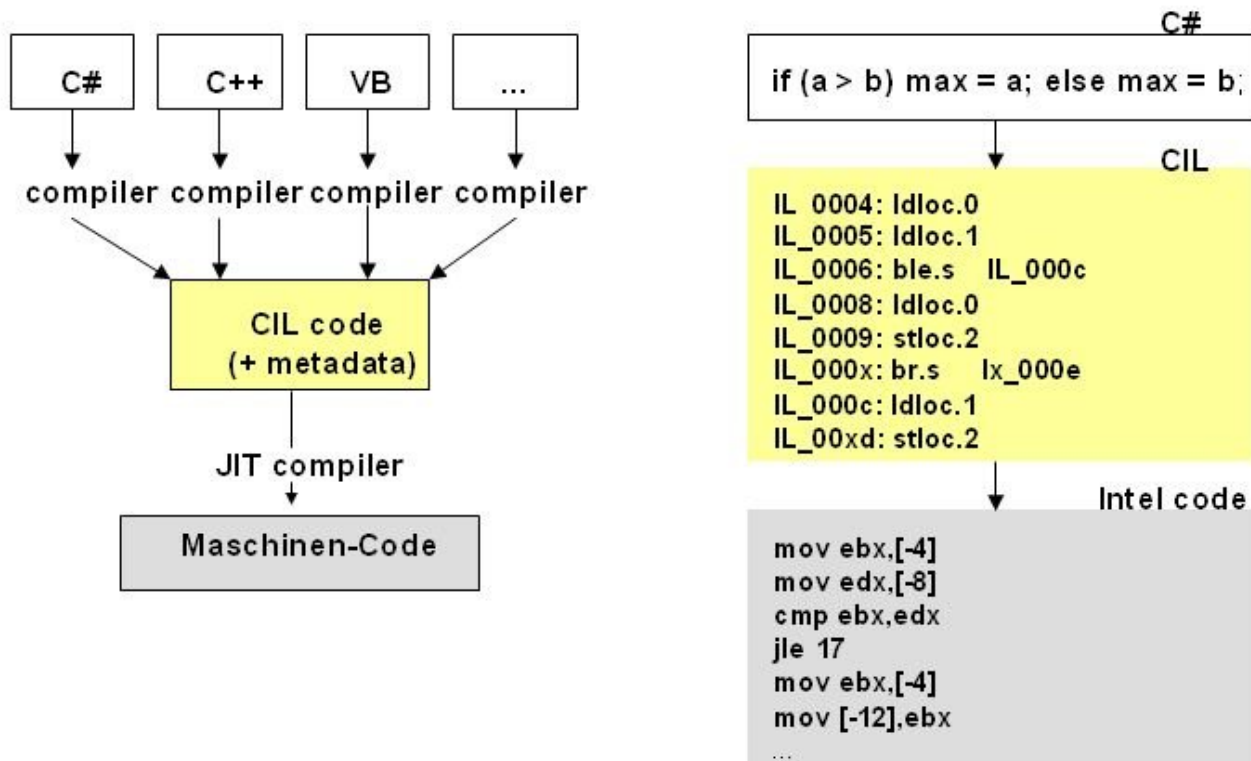
```
class B : A {
  public string s;
  public void Bar()
  {...}
}
```

verwendet in Eiffel

```
class Client feature
  obj: B;
  ...
  create obj;
  obj.Bar;
  ...
end
```

## Ziele des .NET

### Interoperabilität von Programmiersprachen - Beispiel



© 2006 Dagmar Walddobler

## Ziele des .NET

---

### Einfachere Erstellung von dynamischen Webseiten

#### bisher:

- ASP (Mischung aus HTML und VBScript oder Javascript)

#### unter .NET

- ASP.NET (klare Trennung von HTML und Script Code)
- objekt-orientiert
- ereignisgesteuert
- benutzerdefinierte GUI-Elemente
- effizient (compilierter Servercode)

## Ziele des .NET

---

### Mehr Qualität und Komfort

- Sicherheit:** öffentliche key-Signaturen  
codebasierte Zugriffsrechte
- Zero-impact Installation:** keine Registry-Einträge erforderlich  
saubere Deinstallation
- Side-by-Side Execution:** Die CLR ist in der Lage, mehrere Dateien selbst dann in denselben Adressraum zu laden, wenn sie den gleichen Dateinamen tragen ( "Parallelbetrieb" ).
- Verschiedene Clients:** Unterstützung von Smart Devices  
(Handy, PDA, Notebooks, Desktop-PCs)

## Warum C# als neue Sprache?

---

- Keine evolutionären „Altlasten“ vorhanden
  - Konsequente Ausrichtung auf .NET
  - Zusammenführung der Vorteile bereits etablierter Programmiersprachen:
    - so einfach zu Erlernen wie Visual Basic,
    - so mächtig und flexibel wie C++
    - viele Vorteile von Java
  - rein objektorientierte Programmiersprache
  - Sicherstellen innerer Konsistenz und Logik
  - Universalsprache mit hoher Leistungsfähigkeit
  - Erhöhung der Entwicklungsproduktivität
- ➔ Keine wirklich neuen Konzepte: C# ist keine Revolution!

## Merkmale von C#

---

### Sehr ähnlich zu Java:

70% Java, 10% C++, 5% Visual Basic, 15% neu

#### Wie in Java

- Objektorientierung (einf. Vererbung)
- Interfaces
- Exceptions
- Threads
- Namespaces (wie Packages)
- Strenge Typprüfung
- Garbage Collection
- Reflection
- Dyn. Laden von Code
- Spezielle String-Unterstützung

#### Wie in C++

- (Operator) Overloading
- Zeigerarithmetik in Unsafe Code
- Einige syntaktische Details

## Neue Features in C#

---

- Referenzparameter
- Objekte am Stack (Structs)
- Blockmatrizen
- Enumerationen
- Uniformes Typsystem
- goto
- Attribute
- Systemnahes Programmieren
- Versionierung

### "Syntactic Sugar"

- Komponentenunterstützung
  - Properties
  - Events
- Delegates
- Indexers
- Operator Overloading
- foreach-Iterator
- Boxing/Unboxing

## Einfachstes C#-Programm

---

### File Hello.cs

```
using System;

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

- **benutzt Namespace System**
- **Hauptmethode muß immer *Main* heißen**
- **gibt auf Konsole aus**
- **Dateiname und Klassenname müssen nicht übereinstimmen.**

### Übersetzen (im Command-Prompt)

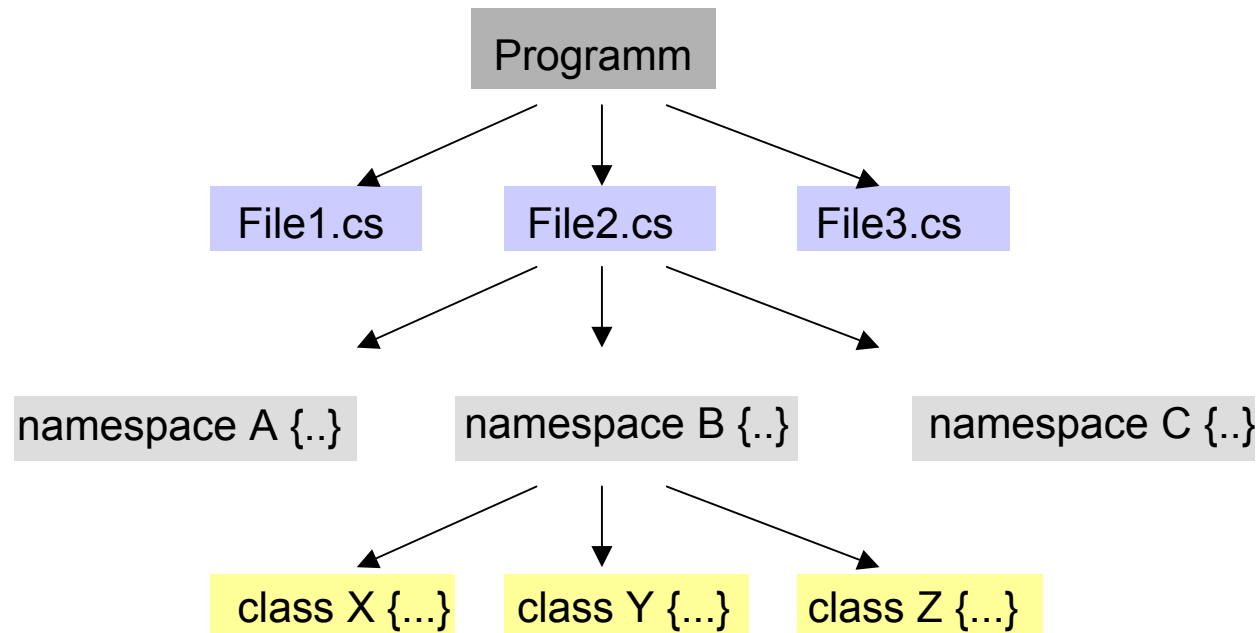
***csc Hello.cs***

### Ausführen

***Hello***

## Gliederung von C#-Programmen

---



- Wenn kein Namespace angegeben → namenloser Standardnamespace
- Namespace kann auch Structs, Interfaces, Delegates, Enums enthalten
- Namespace kann in verschiedenen Files „wiedereröffnet“ werden
- Einfachster Fall: 1 File, 1 Klasse

## Programm aus 2 Dateien

---

### Counter.cs

```
class Counter {  
    int val = 0;  
    public void Add (int x) { val = val + x; }  
    public int Val () { return val; }  
}
```

### Prog.cs

```
using System;  
  
class Prog  
{  
    static void Main() {  
        Counter c = new Counter();  
        c.Add(3); c.Add(5);  
        Console.WriteLine("val = " + c.Val());  
    }  
}
```

## Übersetzen

***csc/target:exe Counter.cs Prog.cs***

## Ausführen

***Prog***

## Arbeiten mit DLLs

***csc/t:library Counter.cs***  
**→ erzeugt Counter.dll**

***csc/r:Counter.dll Prog.cs***  
**→ erzeugt Prog.exe**