

Microsoft .NET Framework

Anita Sosnecki

Universität Bonn

Institut für Informatik

Seminar Softwaretechnologie WS 2003



Übersicht

- Einleitung
- Was ist .NET?
- .NET Framework
- .NET Komponenten
- Assemblies
- Deployment
- Sprachen in .NET
- Vor- und Nachteile von .NET
- Beispiele

Einleitung

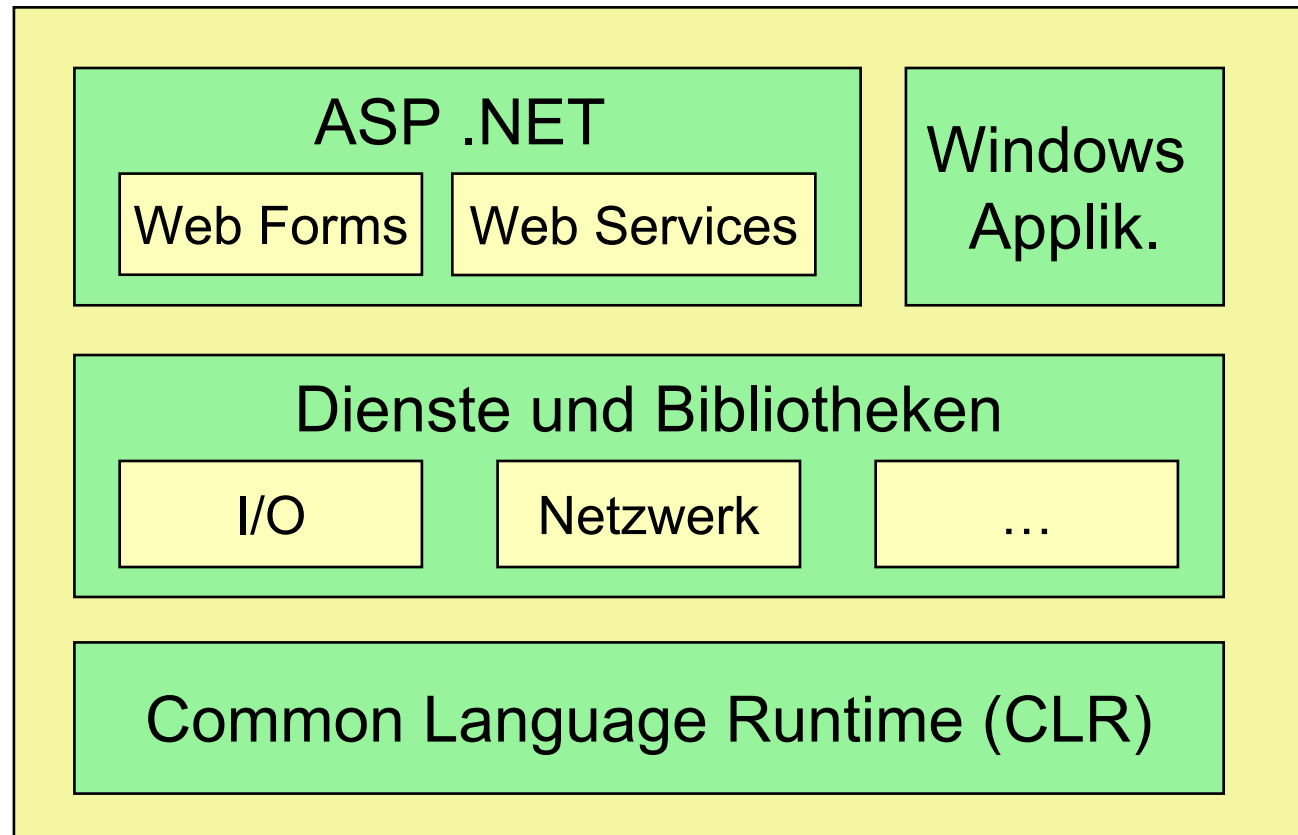
- .NET steht für eine Strategie von Microsoft
 - Benutzer sollen eine Software jederzeit, an jedem Ort und auf jedem Gerät mit optimalem Nutzen einsetzen können
 - Verschiedene Programme und Programmbausteine sollen nur mit geringem Aufwand integriert werden
- Komplette Entwicklungsplattform
 - .NET Framework
 - Mächtige Entwicklungsumgebung: *Visual Studio .NET*

Was ist .NET?

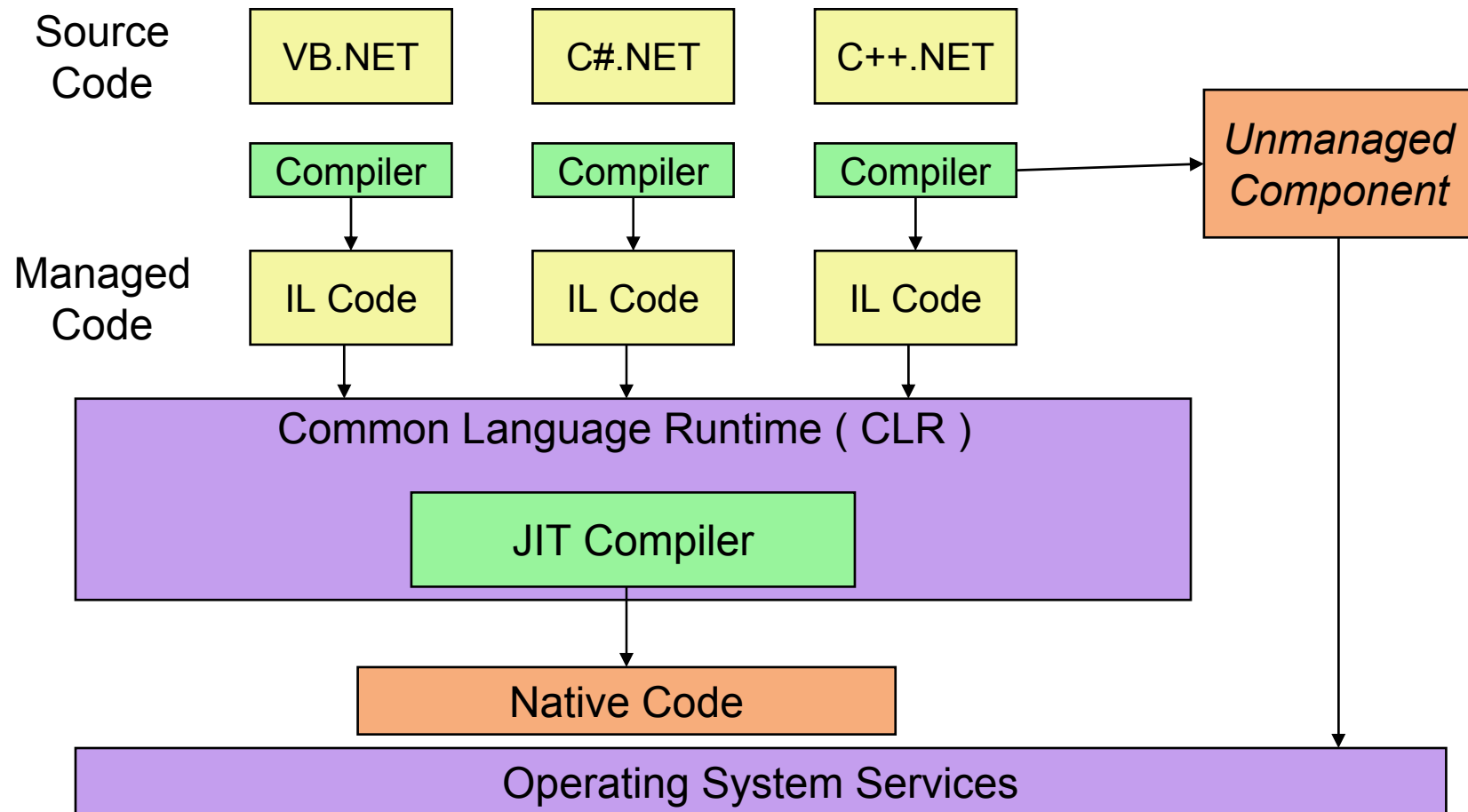
- .NET besteht aus folgenden Komponenten:
 - .NET Framework
 - (Web-basierte) Applikationen, Web Services, Windows Services
 - Internetdienste (Web Services)
 - Komponenten, die Dienste und Funktionalitäten zur Verfügung stellen
 - Verwendung: speziell im Netzwerk
 - .NET Enterprise Server
 - (Windows Server 2003, SQL Server, BizTalk Server ...)
- Unterstützt viele Sprachen
 - C++, C#, VB ...

.NET Framework

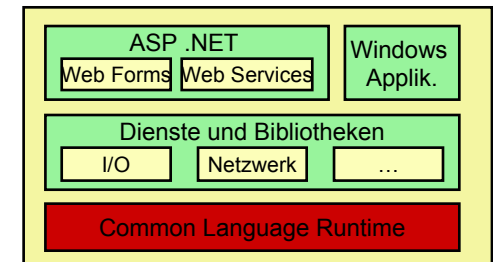
.NET Framework



.NET Framework



Common Language Runtime

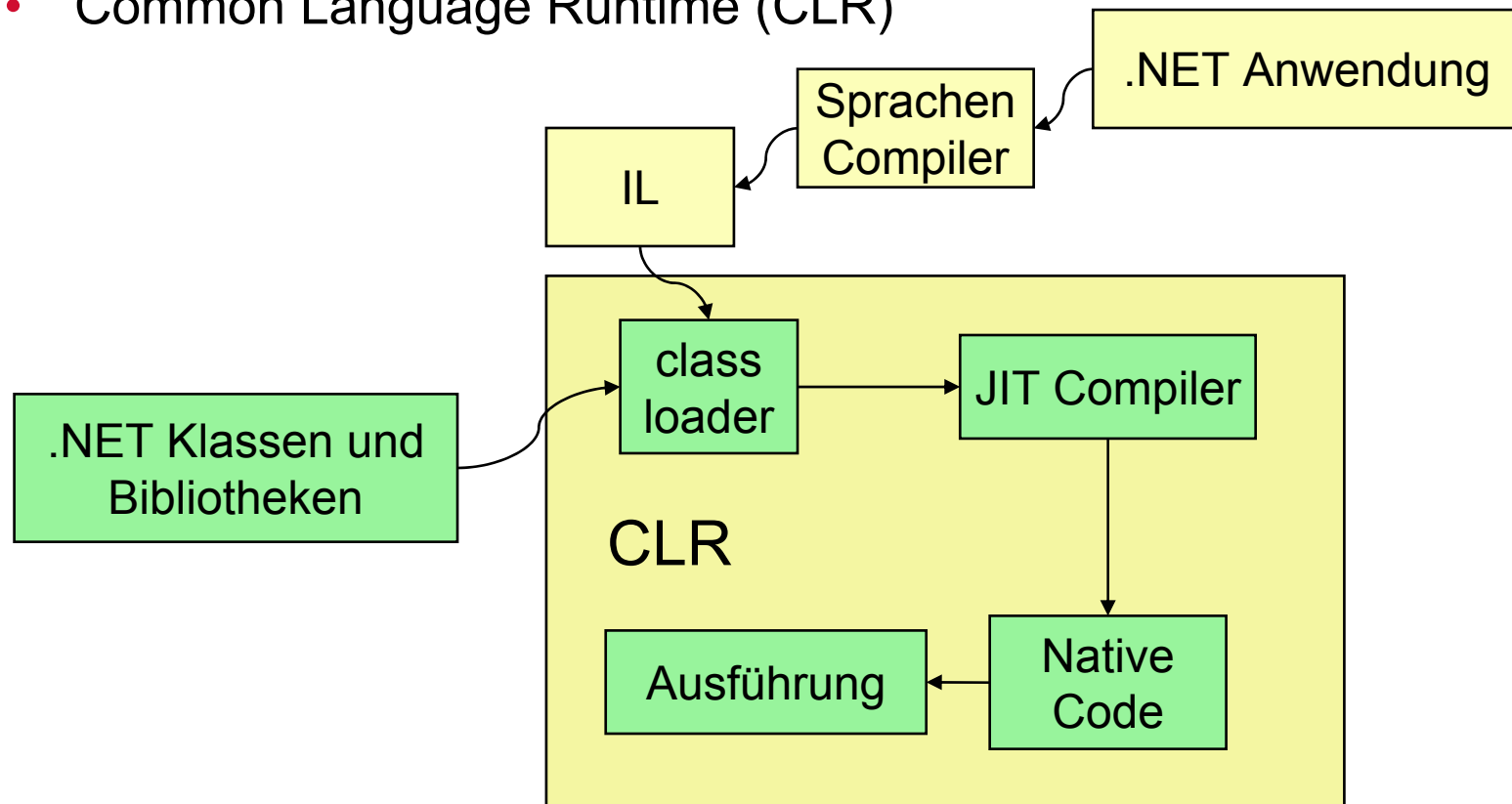


Aufgaben:

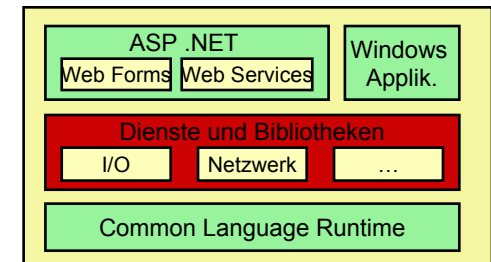
- Compiler (JIT)
- Codeverwaltung (Laden + Ausführen)
- Fehlerbehandlung
- Speicherverwaltung (Garbage Collector)
- Nebenläufigkeit
- Überprüfung von Typensicherheit
- Überprüfung der Herkunft und Identitäten von Komponenten
- Bietet einen einheitlichen Satz von Datentypen (CTS Common Type System)

Common Language Runtime (CLR)

- Common Language Runtime (CLR)

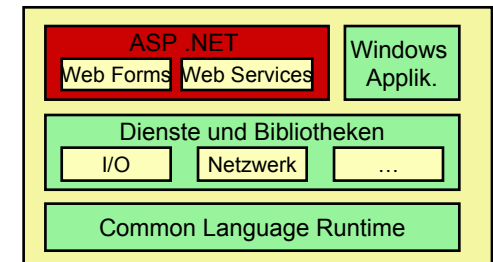


Dienste und Bibliotheken



- Basisklassen des Frameworks
 - I/O, Security Management, Network Communication, Thread Management, Text Management, User Interface Design Features
- Daten und XML Klassen
 - ADO.NET (ActiveX Data Objects)
→ Verwaltung von Daten
 - XML Klassen → XML-Datenbearbeitung
- Klassen, die Zugriff auf Dienste des Betriebssystems haben
- Von jeder .NET Sprache aufrufbar
- Klassen in Namespaces strukturiert und verwaltet

ASP.NET



- Erstellen von Web-Anwendungen
 - Web Forms (dynamisch erzeugte Webseiten)
 - Web Services (Dienste für Prozesse im Internet)
- Erweiterung von ASP (Active Server Pages)
- Unterstützt alle .NET-Sprachen
- ASP.NET viel schneller als ASP → Code wird kompiliert und nicht mehr interpretiert

Web Forms

- Erstellen von dynamischen Webseiten
- Dateien mit der Endung .aspx
- Trennung von Layout und Logik (Code Behind)
 - Layout → MyForm.aspx
 - Code → MyForm.aspx.cs (Code in C#)
- Wird nur einmal kompiliert

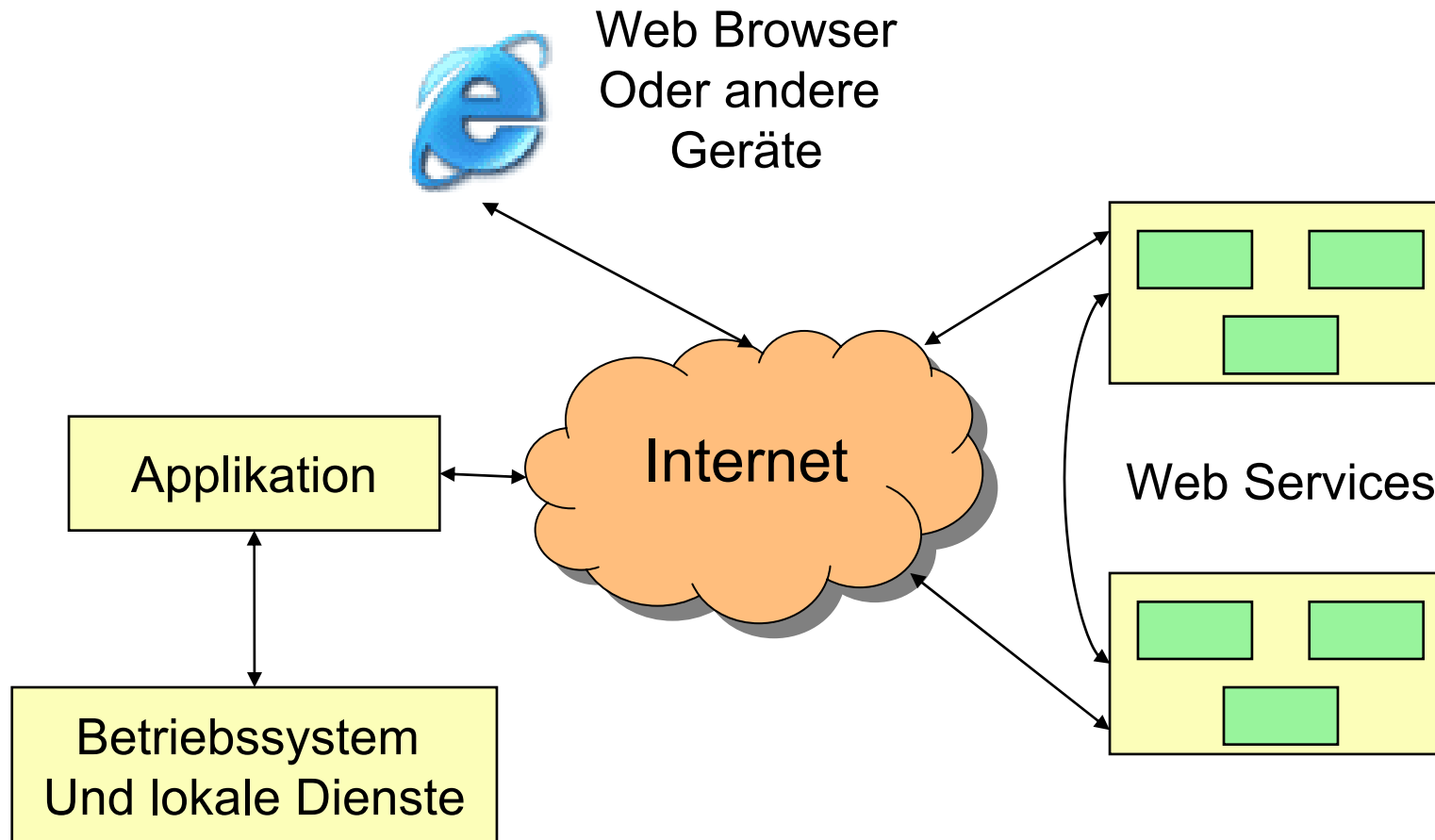
Web Services

- Web Services (Dienste für Prozesse im Internet und Intranet)
 - Nutzung durch unterschiedliche Clients (Laptop, PDA, ...)
 - Basieren auf offenen Standards (XML, HTTP, SOAP)
 - Adressierung über URL
 - Black Boxes (Implementierung nicht einsehbar)
 - Implementierung
 - Kommunikationsprotokoll muss allen Beteiligten bekannt sein →HTTP
 - Asynchrone Kommunikation mittels Versendung von Nachrichten →SOAP
 - Microsoft Internet Information Server (IIS)
 - Textdatei mit der Endung .asmx
- Programmierbares Web

Web Services

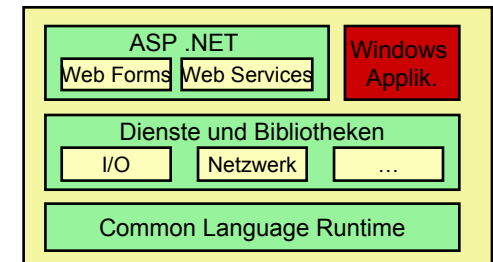
- Web Service kann mittels einer URL aufgerufen werden
- Nutzung des Web Services
 - Wo befindet sich dieser Web Service? → URL
 - Was kann der Service? → WSDL-Beschreibung wird benötigt
- Detaillierte Beschreibung des Web Services
 - Syntax: URL + "?WSDL" (Web Service Description Language)
 - Beispiel: <http://localhost/MyServices/myService.asmx?WSDL>
- Suche von Web Services
 - <http://www.uddi.org>
 - Uddi steht für: Universal Description, Discovery, and Integration

Web Services



Windows Applikationen

- Windows Forms
 - Windowsapplikation mit Benutzeroberfläche
- Windows Services
 - Dienste, die im Hintergrund laufen
 - Beispiel: Windows Update Service



Windows Services

- Windows Services
 - Programme, die im Hintergrund laufen
 - Müssen registriert werden (Einträge in der Windows-Registry)
 - Installationsroutine
 - Benötigt InstallUtil.exe für die Ausführung
 - Status
 - Ausgeführt
 - Beendet (Ressourcen wieder freigegeben, Dienst weiterhin registriert)
 - Angehalten (Ressourcen nicht wieder freigegeben)
- Implementierung
 - Muss von *ServiceBase* abgeleitet sein
 - Zusätzlich: Klasse *ProjectInstaller*

.Net Komponenten

Was ist eine .NET Komponente

- Baustein für eine Anwendung
- Unabhängige Softwareeinheit
 - Besitzt eine Reihe von Funktionen
 - Kann von vielen Anwendungen benutzt werden
- Komponentenklasse, -struct, -modul (nur in Visual Basic)
- Komponenten werden in Assemblies gepackt
- CLR beruht auf Komponenten-Architektur (Komponenten-Bibliothek)

.NET Komponenten: Nutzen

- Komponenten :
 - Wiederverwendbare Programmteile
 - Kapseln bestimmte Funktionalitäten
 - Werden in Bibliotheken angelegt
- Erleichterung der Programmierung
 - Verkürzung der Entwicklungszeit
 - Bessere Wartbarkeit des Codes
 - Kleinere Codeeinheiten

.Net Komponenten: Implementierung

- **Komponentenklasse**
 - Implementiert Interface *System.ComponentModel.IComponent* oder
 - Wird von *System.ComponentModel.Component* abgeleitet
- **Klasse *Component* als Basisklasse für alle Komponenten in *System.Windows.Forms***
 - Implementiert *IComponent* (Funktionalität, die alle Komponenten gemeinsam haben)
 - Implementiert *IDisposable* (Fähigkeit, Ressourcen aufzuräumen)
 - *Dispose()*, *Finalize()*
- **Zugriffsmodifizierer**
 - *Public*, *Private* ...

.NET Komponenten: Nutzung von "altem" Code

- Nutzung alter COM-Komponenten
 - Kommunikation zwischen COM und .NET Klassen mit Hilfe von RCW (Runtime Callable Wrapper)
 - Was passiert im Hintergrund
 - COM-Bibliotheken werden in Assemblies umgewandelt
 - In Visual Studio .NET übernimmt diese Aufgabe der Type Library Importer (tlbimp)
 - Unmanaged Code

→ Einschränkungen bei der Sicherheit und höherer Speicherbedarf
- Nutzung von Win32 API
 - Gewünschte DLL in ein .NET Projekt importieren (Platform invocation - PInvoke) z.B. `[DllImport("user32")]`

Assemblies

Was ist eine Assembly?

- Kleinste verteilbare Einheit in .NET
- Code in IL (Intermediate Language)
- Informationen über sich selbst
- Informationen über den Code (Manifest)
→ Teil der Metadaten
- Manifest !
 - Assembly-Identität = Name + Version + Ländercode
 - Ländercode gemäß RFC 1766, z.B. de-DE
 - Liste der Module, aus denen die Assembly besteht
 - Referenzierte Assemblies
 - Exportierte Ressourcen

Funktionen einer Assembly

- Code (IL) muss ein Assemblymanifest enthalten
- Sicherheit → Berechtigungen anfordern und erteilen
- Identität eines jeden Typen wird überprüft
- Auflösung von Typen und Bereitstellung angeforderter Ressourcen
- Versionierung
 - Alle Typen und Ressourcen in einer Assembly bilden eine Einheit mit der selben Version
 - Gleiche Assemblies verschiedener Versionen können koexistieren
- Parallele Ausführung

Arten von Assemblies

- Ausführbare Assembly → .exe
- Bibliothek → .dll

- Statisches Assembly → Persistent
- Dynamisches Assembly → Transient

- Private Assembly
 - Wird im Verzeichnis der referenzierten .NET Anwendung gespeichert
- Shared Assembly
 - Öffentlich
 - Liegt im globalen Verzeichnis GAC (Global Assembly Cache)
 - Zugriff von mehreren .NET Anwendungen möglich
 - Strong Names

Deployment

- Deployment
 - Verteilung einer Applikation
 - Erstellen einer Installationsversion inklusive evtl. notwendiger Konfigurationsdateien
 - Wichtigstes Element : *Assembly*
- XCopy-Deployment
 - Komplette Applikation kann einfach in ein anderes Verzeichnis und sogar auf einen anderen Computer kopiert werden
 - Verweise innerhalb der Verzeichnisstruktur sind relativ

Sprachen in .NET

- Über 20 Programmiersprachen werden unterstützt
- C#, eine von Microsoft neu entwickelte Sprache
 - Syntax und Semantik ähnlich wie bei Java und C++
 - Durchgehend objektorientiert

```
class MyApp {  
    public static void Main() {  
        System.Console.WriteLine ("Hello, C#!");  
    }  
}
```

Sprachen in .NET

- VB.NET
 - Aber: Redesign war nötig, um Anforderungen des Frameworks zu erfüllen
- C++
 - Erweiterung um automatische Speicherverwaltung
 - Features wie Destruktor, delete() werden nicht mehr unterstützt ...
- Viele weitere Sprachen
 - Java, JScript, Delphi, Cobol, Fortran,
 - Haskell, Pascal, Perl, Python, Scheme

Vor- und Nachteile von .NET

- + Spracheninteroperabilität, d.h. keine sprachenspezifische Runtime mehr.
- + CTS (Common Type System), ein gemeinsames Typkonzept
→ Klassen, die in unterschiedlichen Sprachen erstellt wurden, können voneinander erben
- + .NET ist vollkommen objektorientiert und komponentenorientiert
aber: hoher Lern- bzw. Einarbeitungsaufwand
- + Ende der DLL-Hölle
- Betriebssystemunabhängigkeit ??? (<http://www.go-mono.com>)
- ASP und ASP.NET nicht kompatibel (unmanaged Code)
- Entwicklungsumgebung sehr teuer (Visual Studio .NET)

Beispiele

- Windows Anwendung (Visual Studio .NET)
- Web Service

