Introduction

Dienstag, 22. November 2016

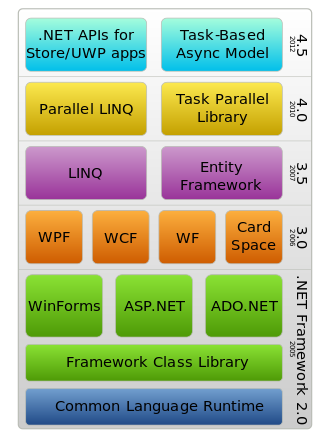
12:46

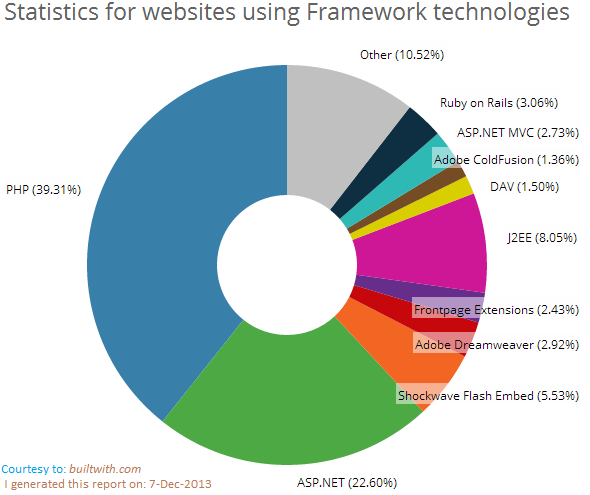
**Microsoft .NET Platform**

[](https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjgztXIlsnSAhUD7BQKHSuXDdwQjRwIBw&url=https%3A%2F%2Fworldcompressed.blogspot.com%2F2016%2F06%2Fdownload-microsoft-net-framework-all.html&psig=AFQjCNFN-WTg4re8nPn9-_o6Bp1-jprxmA&ust=1489140340450668)

* 2002 eingeführt
* Hauptprogrammierplattform für Windows

**.NET Versionen und Komponenten**

[](https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=imgres&cd=&cad=rja&uact=8&ved=0ahUKEwjSpf-LwsnSAhWJMhoKHY6iDE0QjRwIBw&url=https%3A%2F%2Fde.wikipedia.org%2Fwiki%2F.NET_Framework&psig=AFQjCNFvNcpuHgYgsFvqmz8gVkI9FErEPQ&ust=1489152123324209)



**Features**

* Komponentenbasiert: Unterschiedliche Komponenten aus unterschiedlichen Programmiersprachen:
  + C#, Visual Basic, F#, JavaScript, C++ (in Visual Studio)
  + Ruby, Python, COBOL, Pascal (externe Compiler)
  + <https://en.wikipedia.org/wiki/List_of_CLI_languages>
  + Vererbung, Exception Handling, Debugging über mehrere Programmiersprachen hinweg
* Versionskompatibilität: mehrere Versionen der gleichen .dll können nebeneinander existieren
* .NET Distributionen:
  + Microsoft, included in Visual Studio
  + Andere Distributionen ([www.mono-project.com](http://www.mono-project.com), [www.dotgnu.org](http://www.dotgnu.org)) stellen Compiler und Runtime Environment für Linux Android, IOS, …
  + .NET Core (plattformunabhängig, schlank)
* Erstellte Anwendungen sind Plattform-unabhängig, und laufen unter verschiedenen Betriebssystemen (Windows, MacOS, Linux, Android, … )

**Warum mehrere Programmiersprachen ?**

**Wie tauschen sich die unterschiedlichen Programmiersprachen untereinander aus ?**

**Compilierung in .NET**

* Erzeugt \*.dll Bibliotheken und \*.exe Dateien -> heißen Assembly
* Diese Dateien haben bis auf ihre Endung nichts mit normalen .exe, .dll Dateien zu tun
* Dateien enthalten Code in einer Intermediate Language (IL) + Metadaten + Manifest
* Enthält keine Plattform-spezifischen Informationen ->plattformunabhängig
* Metadaten beschreiben Typen, bspw. von wem eine Klasse erbt, welche Methoden sie hat, Version, …

C# Compiler 
Source Code 
Perl .NET 
Source Code 
COBOL .NET 
Source Code 
C++/CLI 
Source Code 
Perl 
COBOL 
. NET Cornpiler 
.NET Compiler 
IL 
and 
( • $611 or •.exe) 
C++/CLI Compiler 

**C# Code**

class Calc 
public int Add(int x, 
{ return x + y; } 
int y) 

**Visual Basic Code**

Class Calc 
Public Function Add(ByVa1 x As Integer, ByVa1 y As Integer) As Integer 
Return x + y 
End Function 
End Class 

**Entsprechender IL Code**

<https://en.wikipedia.org/wiki/List_of_CIL_instructions>

.method public hidebysig instance int32 Add(int32 x, 
int32 y) cil managed 
// Code size 9 (Ox9) 
.maxstack 2 
. locals init (int32 V 0) 
IL 
0000 : 
IL 0001: 
IL - 
0002 : 
IL 
0003 : 
IL 
0004: 
IL - 
0005 : 
IL 
0007 : 
0008 : 
IL 
end of 
nop 
Idarg.l 
Idarg.2 
add 
stloc.o 
br.s IL 0007 
Idioc.o 
ret 
method Calc: :Add 

**Ausführung des Programms**:

* ruft just-in time compiler (JIT, jitter) auf der den code compiliert und ausführt
* Methoden werden nur einmal kompiliert und im Speicher gelagert, (-> kein Interpreter) so dass mehrfacher Aufruf der gleichen Methode nur eine einzige Kompilierung benötigt.

**Teile von .NET**

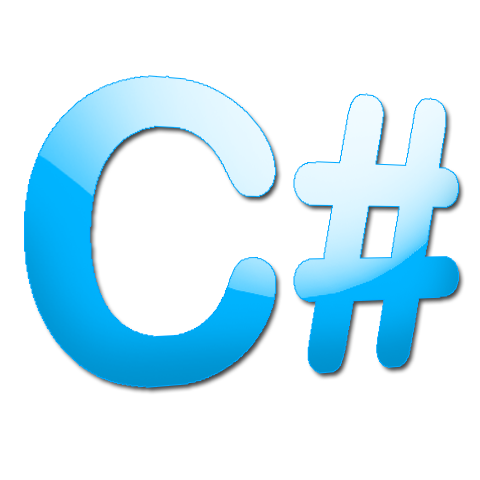
* Common Language Runtime (CLR)
  + Findet, lädt und managed .Net Objekte
  + Nutzt Metadaten der Assembly um Typen zu finden
  + Memory management, thread coordination, …
  + Compiliert nötigen IL Code und führt ihn aus
* Common Type System (CTS)
  + Spezifiziert Datentypen und Programmkonstrukte und deren Interaktion
* Common Language Specification (CLS)
  + Definiert eine Teilmenge von CTS, die über alle .NET Sprachen geteilt ist
  + Denn nicht jede .NET Sprache unterstützt alle features und Datentypen der CTS
  + Programmierung nur mit CLS features erzeugt Code Bibliotheken die garantiert mit jeder anderen .NET Sprache kommunizieren können
  + Festgelegt in Regeln
    - Erste Regel: Alle Regeln beziehen sich nur auf die Teile eines Typs, der außerhalb eines Assemblys sichtbar sind
  + C# Compiler kann prüfen, ob die Regeln eingehalten wurden, also ob der Code der CLS entspricht
* Base Class Library
  + Existiert für jede .NET Sprache
  + Beinhaltet Behandlung von I/O, Threads, Grafik, Hardware-Schnittstellen

The Base Class Libraries 
The Common Language Runtime 
Common Type System 
Common Language Specification 

**Wie behält man die Übersicht bei vielen verschiedenen Klassen ?**

**Namespaces**

* Zur Organisation aller Typen der .NET Platform über alle Sprachen hinweg
* Namespaces können subspaces (Unterräume) enthalten
  + System, System.IO, …
* Jedes Assembly kann beliebig viele Namespaces enthalten
* Bsp: System namespace enthält System.Int32, System.String, …
* Zugriff auf Typen durch
  + Namespace.Subnamespace.Subnamespace.Typname
  + using keyword: using Namespace.Subnamespace.Subnamespace Typname

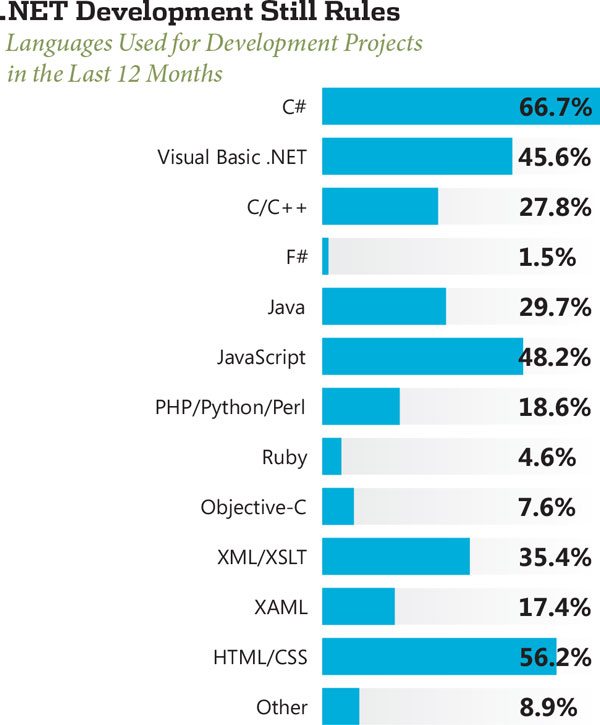
[](https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiJtrTulsnSAhWEtRQKHff7DlkQjRwIBw&url=http%3A%2F%2Fwww.pskills.org%2Fcsharp.jsp&psig=AFQjCNFYC8mUdYV-3tEzGwUgZFK1Nymh5A&ust=1489140478449392)

**Die Sprache C#**

* Syntax sehr ähnlich zu Java
* Vereint Konzepte von Java, Visual Basic, C++
* Idee: die Vorteile aller Sprachen vereinigen
  + Syntaktisch sauber wie Java
  + Einfach wie Visual Basic
  + Flexibel und Mächtig wie C++

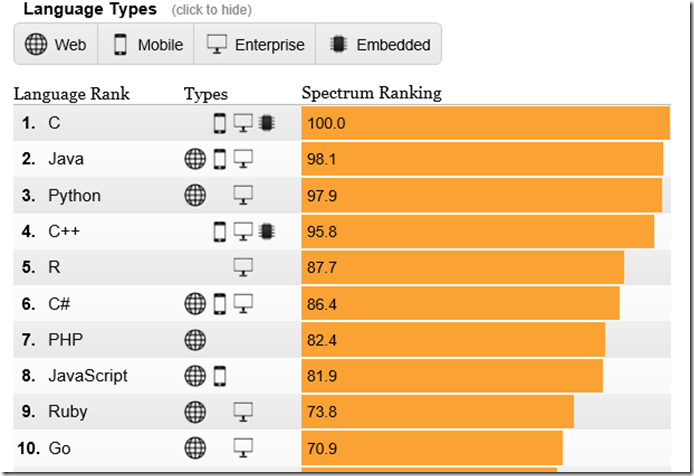
**Features**

* Keine Pointer nötig
* Automatisches Speichermanagement (Garbage Collector)
* Operatorüberladung
* Attribut-basierte Programmierung
* Generische Programmierung
* Lambda Operatoren
* Einfaches Multithreading mit Async/Await Keywords

[](https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwia_p2ducnSAhVDWxQKHakcA2cQjRwIBw&url=https%3A%2F%2Fvisualstudiomagazine.com%2Farticles%2Fsalary-surveys%2Fsalary-survey.aspx&bvm=bv.149093890,d.ZGg&psig=AFQjCNF4YjINx5DQNIfO6VtJlZsobTLvuQ&ust=1489143523387743)

Aus <<https://www.google.de/search?q=c%23+statistics&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjtiJ30ocnSAhWsIsAKHRvXAKAQ_AUIBigB&biw=1920&bih=918>>

Most popular coding languages

[](https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwiYtJOku8nSAhXHcRQKHf_hDUkQjRwIBw&url=https%3A%2F%2Fblogs.msdn.microsoft.com%2Fzxue%2F2016%2F10%2F24%2Fhow-many-developers-use-c-vs-c-vs-other-programming-languages%2F&bvm=bv.149093890,d.ZGg&psig=AFQjCNGodB1me71e-XGThMOizFH8IpqD4g&ust=1489150274544850&cad=rjt)

Aus <<https://www.google.de/search?q=c%23+f%C3%BCr+windows+anwendungen+statistik&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjsr9fgusnSAhXCFsAKHRSBC1wQ_AUICSgC&biw=1920&bih=918>>

**.NET Core** ist eine [freie und quelloffene](https://de.wikipedia.org/wiki/Free/Libre_Open_Source_Software) [Software-Plattform](https://de.wikipedia.org/wiki/Plattform_(Computer)) (innerhalb) der [.NET](https://de.wikipedia.org/wiki/.NET)-Plattform

* 2016 veröffentlicht
* Plattform-unabhängig!
* Abgespeckte Version von .NET
* Fast alles was wir hier lernen kann direkt in .NET Core verwendet werden

**Gliederung des Kurses**

* **Grundlagen C#**
* **Vererbung, Polymorphie, Schnittstellen**
* **Generische Programmierung**
* **Fehlerbehandlung Exceptions**
* **Delegates und Ereignisse**
* **Windows Forms**
* **Parallele Programmierung**
* **Test-basierte Entwicklung**
* **Agile / emergente Entwicklung**

[](https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwj1odz60dPSAhUJcBoKHTPpDtUQjRwIBw&url=http%3A%2F%2Fwww.enterprise-application-development.org%2Fresearch%2Fpartners%2Fcompanies.html&psig=AFQjCNFACNefN0HSLEq14Zu_mAY1ovJSfw&ust=1489499976654297)