01 Einführung in Microsoft .NET

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Agenda

- .NET-Grundlagen
- Die Common Language Specification (CLS)
- Das Common Type System (CTS)
- Die Common Language Runtime (CLR)
- Das .NET-Framework
- Die .NET-Klassenbibliothek
- Namensräume
- .NET-Assemblies
- Globaler Assembly Cache (GAC)
- .NET/C# im Vergleich zu Java
- .NET Standard

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Merkmale von .NET (1/4)

- **Objektorientierung**
  - .NET ist 100% objektorientiert
  - Alle Elemente lassen sich auf Objekte zurückführen, selbst Datentypen wie Integer oder Float
  - Zugriffe aufs Betriebssystem werden durch .NET-Klassen gekapselt
- **Ersatz des Win32-APIs**
  - Langfristig beabsichtigt Microsoft, das Win32-API durch die Klassen des .NET-Frameworks zu ersetzen
  - Es ist egal, ob eine Anwendung mit Visual Basic .NET programmiert wird oder mit C# oder C++.
    Alle Sprachen greifen auf die gleiche Bibliothek zurück.
    Sprachspezifische Bibliotheken sind nicht mehr vorhanden.

Merkmale von .NET (2/4)

- **Plattformunabhängigkeit**
  - Anwendungen, die auf .NET basieren, laufen in einer Umgebung, die mit der virtuellen Maschine von Java verglichen werden kann.
  - Erst zur Laufzeit einer Anwendung wird der Maschinencode erzeugt.
  - Die Spezifikation der Laufzeitumgebung (***Common Language Runtime – CLR***) hat Microsoft offen gelegt.
  - Die *Common Language Runtime* lässt sich auf Plattformen portieren, die nicht Windows sind, z.B. UNIX oder Linux.
  - Mit dem Projekt Mono wurde .NET erfolgreich auf die Linux-Plattform portiert.
  - .NET Core ist eine plattformunabhängig Neuimplementierung von .NET und Open Source

Merkmale von .NET (3/4)

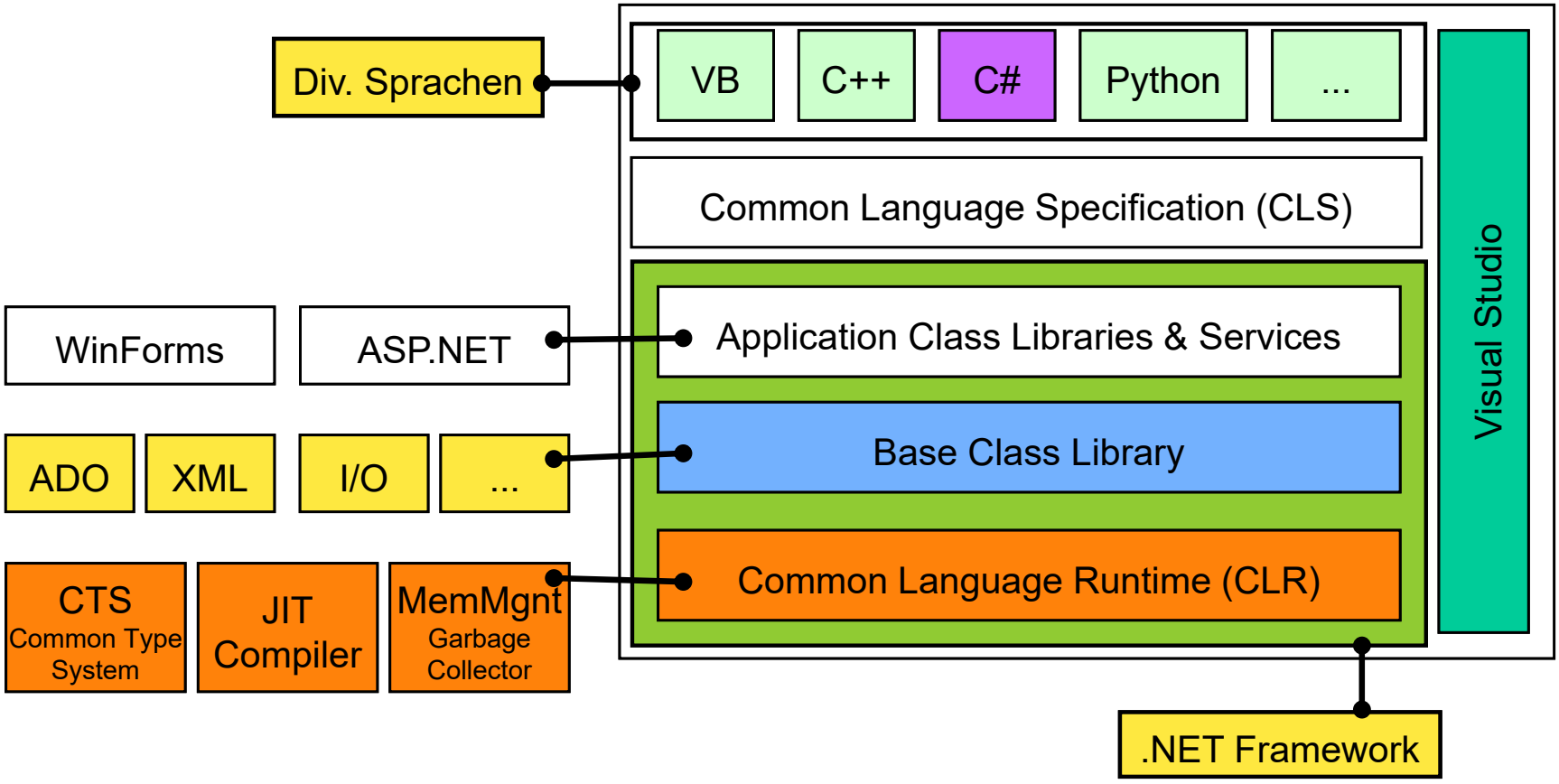
- **Sprachunabhängigkeit**
  - Es spielt keine Rolle, in welcher Programmiersprache eine Komponente entwickelt wird. Eine in C# geschriebene Klasse kann aus VB.NET, J# oder jeder anderen .NET-konformen Sprache heraus aufgerufen werden.
  - Beispielsweise kann eine in C# implementierte Klasse auch von einer VB.NET-Klasse ableiten – oder umgekehrt.

- **Speicherverwaltung**
  - Die Freigabe von nicht mehr benötigtem Speicher war schon immer ein Problem. Unter .NET braucht sich ein Entwickler darum nicht mehr zu kümmern, da der im Hintergrund arbeitende Prozess des *Garbage Collectors (GC)* diese Aufgaben übernimmt und nicht mehr benötigte Objekte erkennt und automatisch aus dem Speicher entfernt.

Merkmale von .NET (4/4)

- **Weitergabe (Deployment)**
  - Ein .NET-Programm weiterzugeben ist viel einfacher geworden, insbesondere im Vergleich zu einem auf COM basierenden Programm, das Einträge in die Registrierungsdatenbank vornehmen muss.
  - Im einfachsten Fall reicht es vollkommen aus, ein .NET-Programm (EXE- oder DLL-Datei) in das dafür vorgesehene Verzeichnis zu kopieren.
  - Darüber hinaus ist aber auch die Verteilung mit einem Installationsassistenten möglich.
  - Ab .NET 2.0 ist ClickOnce ermöglicht.

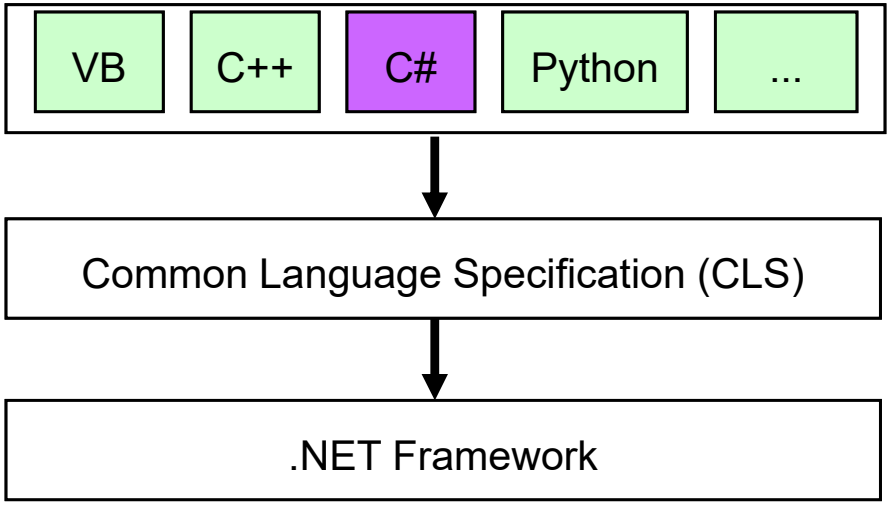
Übersicht der .NET Komponenten



Die Common Language Specification (CLS)

Sprachunabhängiger Code durch CLS

- Um sprachunabhängigen Code erzeugen zu können, muss es Richtlinien geben, an die sich alle .NET-Sprachen halten müssen.
- Diese Richtlinien, in denen die fundamentalen Eigenschaften einer .NET-kompatiblen Sprache festgelegt sind, werden durch die **Common Language Specification (CLS)** beschrieben.
- Die Common Language Specification ist ein offener Standard.



Common Type System (CTS)

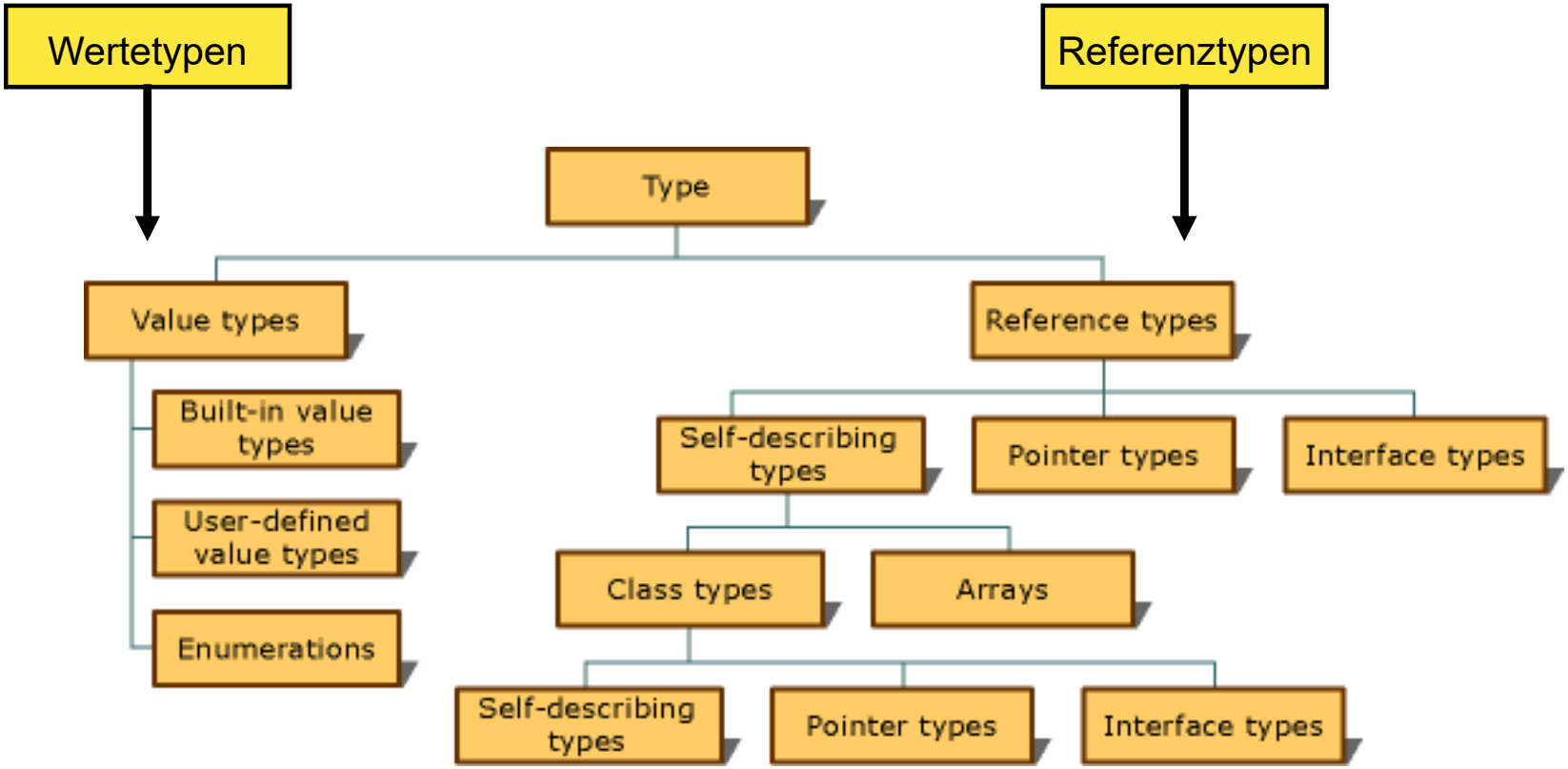
Merkmale

- CTS spezifiziert das Datentypsystem für sprachübergreifende Programmentwicklung
- Die CTS-Spezifikation stellt sicher, dass Programmcode unabhängig von der zugrundeliegenden Sprache miteinander interagieren kann.
- CTS ist die Grundlage für die Sprachunabhängigkeit.
- Alle Typen, die unter .NET zur Verfügung gestellt werden, lassen sich in zwei Kategorien aufteilen:
  - **Wertetypen:** Direkte binäre Wert-Abbildung auf dem Stack (Variablen vom Typ: int, float,...)
  - **Referenztypen:** Abbildung auf dem Heap (Instanzen von Klassen, Arrays,...)

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Common Type System (CTS)

Die CTS-Struktur

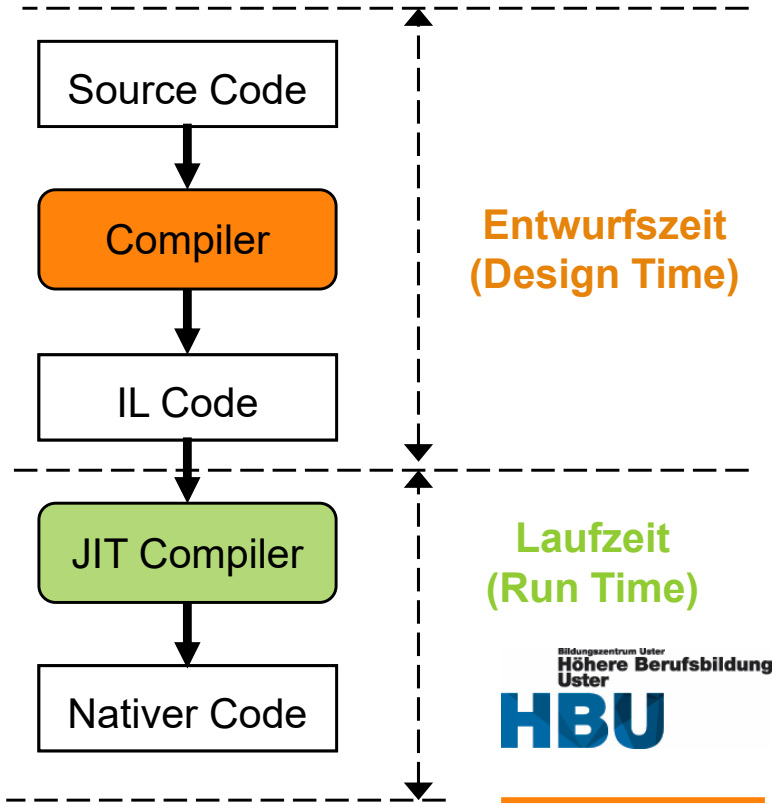


Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Die Common Language Runtime (CLR)

Übersicht: Von der Entwurfs- zur Laufzeit (1/2)

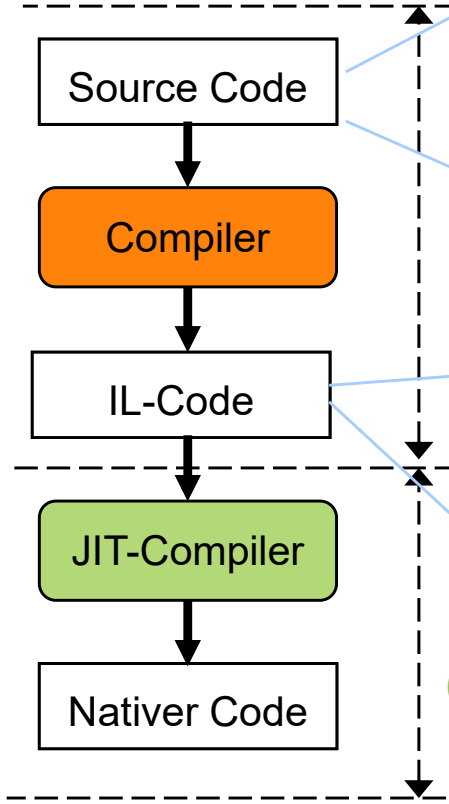
- Die Entwicklung .NET-basierter Anwendungen ähnelt dem Ablauf der Kompilierung bis zum Start der Laufzeit unter Java.
- Zuerst wird ein Zwischencode erzeugt, der CPU-unabhängig ist. Die Dateiendung lautet *.exe*, wenn wir eine eigenstartfähige Anwendung entwickelt haben.
- Allerdings ist diese Datei nicht ohne weiteres lauffähig. Sie benötigt zur Laufzeit einen »Endcompiler«, der den Zwischencode in nativen, plattformspezifischen Code übersetzt.
- Der Zwischencode einer .NET-Anwendung wird als **MSIL**-Code (*Microsoft Intermediate Language*) oder nur kurz als **IL** bezeichnet.
- Der Endcompiler wird als **JIT**-Compiler *(Just-In-Time)* oder auch nur kurz als **JITter** bezeichnet.



Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Die Common Language Runtime (CLR)

Beispiel zu IL-Code



C# Source Code

```
using System;
class HelloWorld
{
  public static int Main(String[] args)
  {
    Console.WriteLine("Hello, World");
    return 0;
  }
}
```

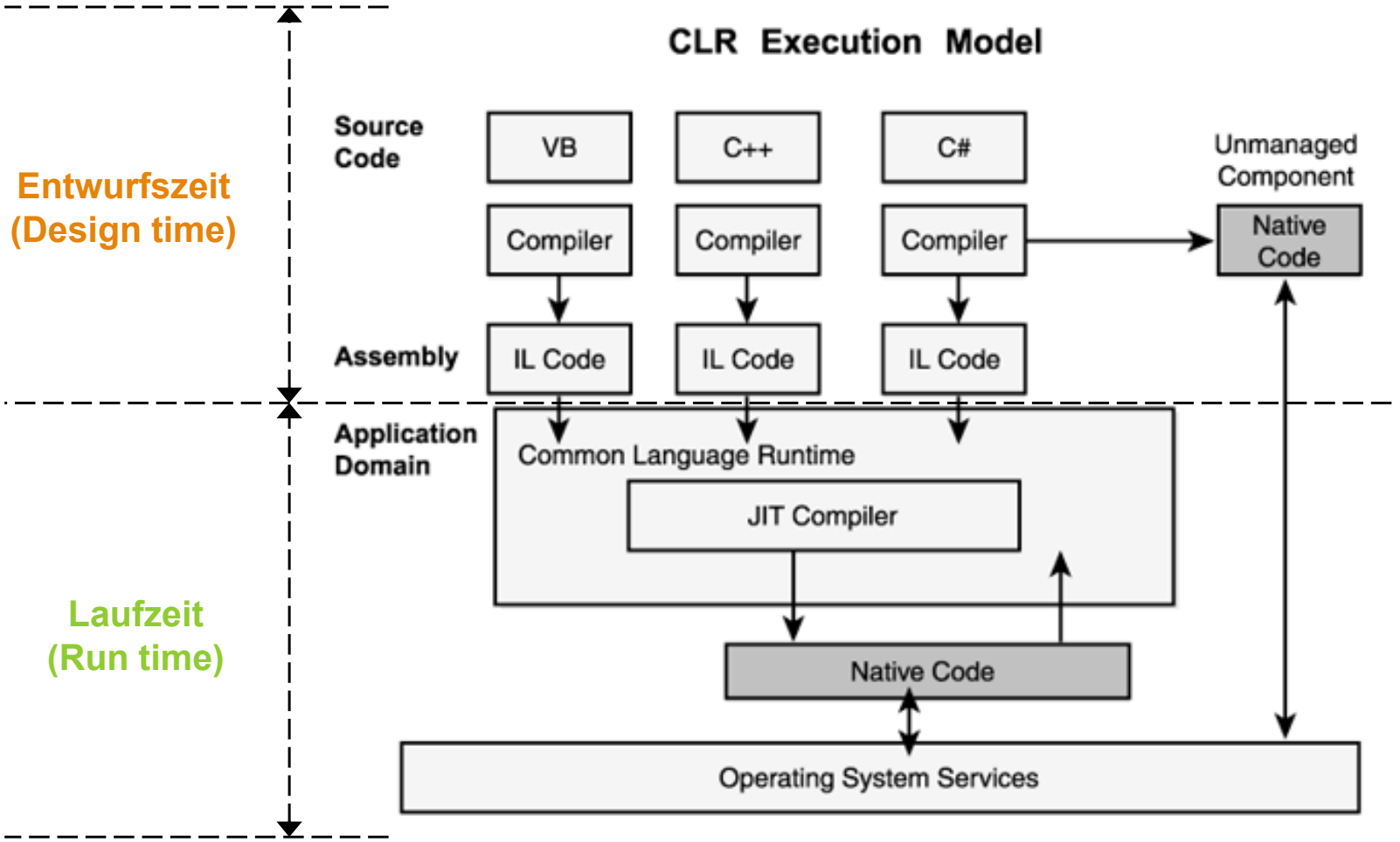
IL-Code

::Main : int32(string[])

```
.method private hidebysig static int32  Main(string[] args) cil managed
{
  .entrypoint
  .custom instance void [mscorlib]System.STAThreadAttribute::.ctor() = ( 01 00
  // Code size       16 (0x10)
  .maxstack  1
  .locals init (int32 V_0)
  IL_0000:  ldstr      "Hello World"
  IL_0005:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_000a:  ldc.i4.1
  IL_000b:  stloc.0
  IL_000c:  br.s       IL_000e
  IL_000e:  ldloc.0
  IL_000f:  ret
} // end of method HelloWorld::Main
```

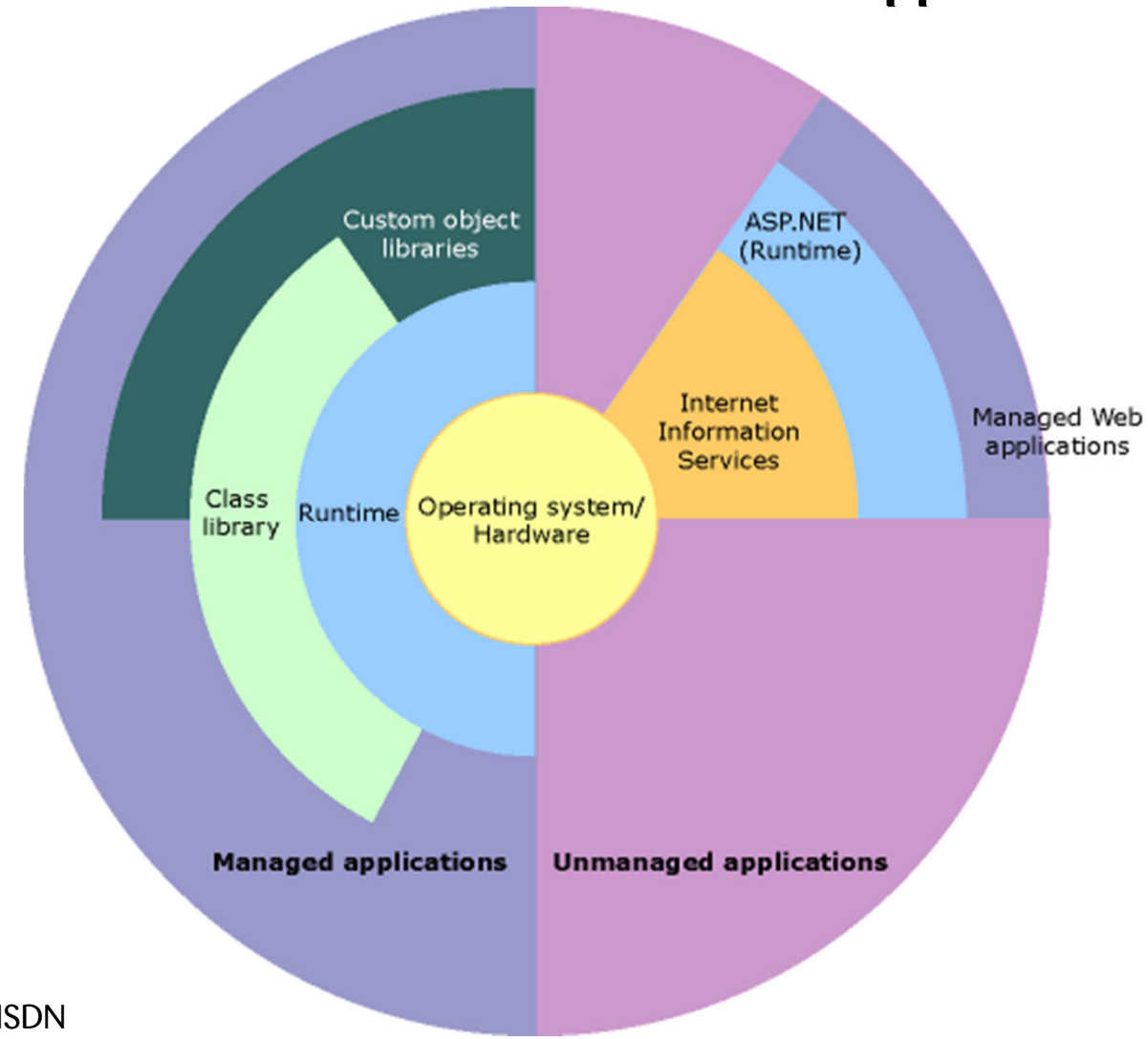
Die Common Language Runtime (CLR)

Übersicht: Von der Entwurfs- zur Laufzeit (2/2)



Die Common Language Runtime (CLR)

Übersicht - .NET im Kontext zu OS und Applikationen



Quelle: MSDN

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Das .NET-Framework

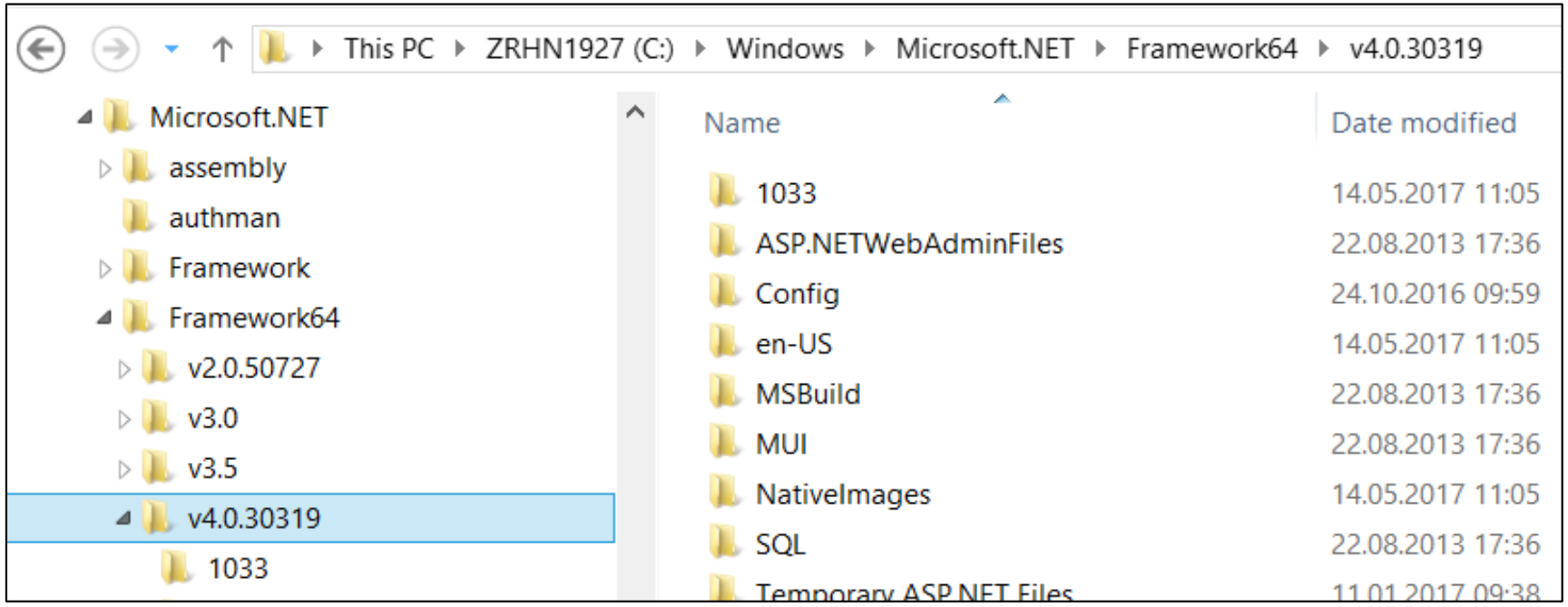
.NET-Framework-Komponenten

- Common Language Specification (CLS)
- Common Type System (CTS)
- Common Language Runtime (CLR)
- .NET-Klassenbibliotheken
  - Windows Forms (WinForms)
  - Active Server Pages .NET (ASP.NET)
  - Active Data Objects .NET (ADO.NET)
  - Entity Framework (EF)
  - Workflow Foundation (WF)
  - Windows Presentation Foundation (WPF)
  - Windows Communication Foundation (WCF)
  - und weitere

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

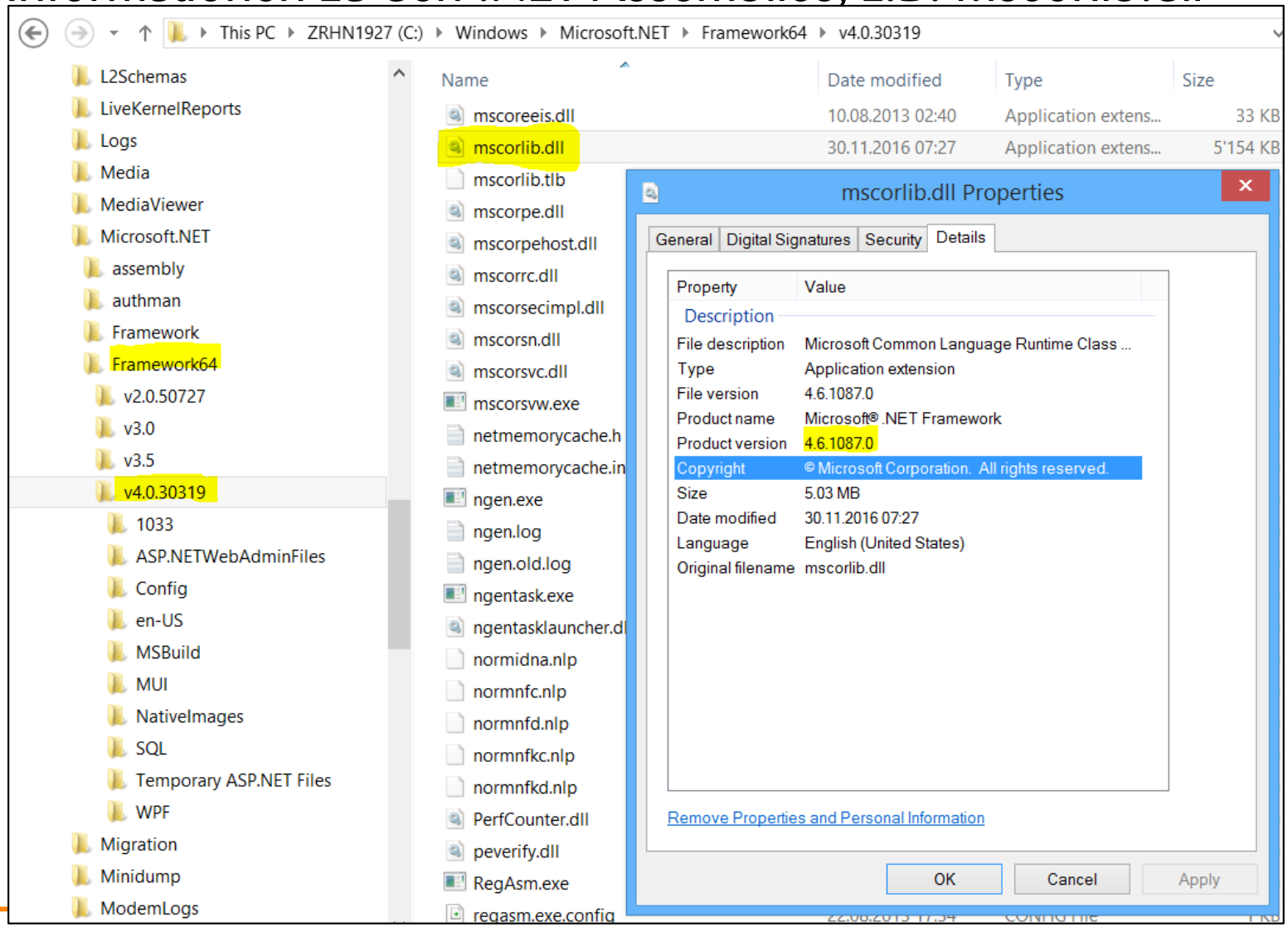
Die .NET-Klassenbibliothek

- Das .NET-Framework ist objektorientiert
- Aktuelle Version ist 4.8 (18.04.2019); keine Weiterentwicklung mehr
- Die .NET-Klassen stehen in einer engen Beziehung (.NET-Klassenhierarchie)
- Verzeichnis: C:\Windows\Microsoft.NET\...



Die .NET-Klassenbibliothek

- Informationen zu den .NET Assemblies, z.B. mscorlib.dll



Namensräume

Überblick

- Ein Namensraum (namespace) ist eine logische Organisationsstruktur, um Klassen zu strukturieren, unabhängig von der Klassenhierarchie.
- Ein Namensraum ermöglicht es Klassen logisch zu gruppieren, anhand ihrer Themengebiete.

Namensräume

Einige Namensräume aus dem .NET-Framework

Namespace	Beschreibung
System.Collections	Klassen für Objektsammlungen
System.Data	Klassen für den Zugriff auf Datenbanken über ADO.NET
System.Drawing	Klassen mit grafischen Funktionalitäten
System.IO	Klassen für Ein- und Ausgabeoperationen
System.Web	Klassen im Zusammenhang mit dem HTTP-Protokoll
System.Windows.Forms	Klassen, um Windows-basierte Anwendungen zu entwickeln

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

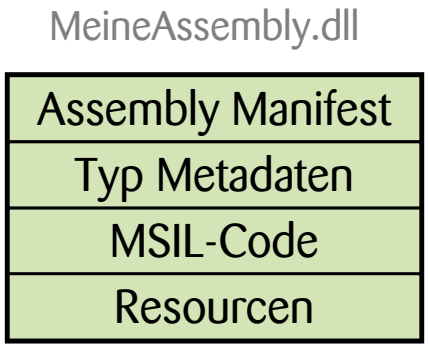
.NET-Assemblies

Grundlagen zu Assemblies

- Das Ergebnis der Kompilierung von .NET-Quellcode ist eine Assembly.
- Je nach Projekttyp wird entweder eine EXE- oder DLL-Assembly erzeugt.
- Die erstellten Assemblies sind nicht gleich denjenigen EXE oder DLL, welche mit VB 6.0 oder C/C++ erzeugt werden.
- Assemblies enthalten IL-Code, welcher zur Laufzeit einer Anwendung vom JITter in nativen Code kompiliert und anschliessend ausgeführt wird.
- Eine Assembly ist als Baugruppe einer Anwendung zu verstehen.

Assembly – Inhalte

- Im Allgemeinen kann eine Assembly vier Elemente enthalten:
  - Das Assembly-Manifest enthält die Assembly-Metadaten
  - Die Typ-Metadaten
  - MSIL-Code (Microsoft Intermediate Language), der die Typen implementiert
  - Ressourcen



.NET-Assemblies

Assembly – Manifest

Ein Manifest enthält die folgenden Informationen:

- Name und Versionsnummer der Assembly
- Angaben über andere Assemblierungen, von denen die aktuelle Assembly abhängt
- Die von der Assembly veröffentlichten Typen
- Sicherheitsrichtlinien, nach denen der Zugriff auf die Assembly festgelegt wird

Das Manifest macht es möglich, dass Assemblies sich nicht in die Registrierung eintragen zu müssen.

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Assembly – Metadaten

- Metadaten sind Daten, die eine Komponente beschreiben.
- Metadaten in der Assembly sind quasi ein Inhaltsverzeichnis des enthaltenen IL-Codes.

Globaler Assembly Cache (GAC)

Grundlagen

- Jeder Computer, auf dem die Common Language Runtime installiert ist, besitzt einen computerweiten Codecache, den so genannten globalen Assembly Cache.
- Im globalen Assembly Cache werden Assemblies gespeichert, die speziell für die gemeinsame Verwendung durch mehrere Anwendungen auf dem Computer vorgesehen sind.
- Zwei GAC Versionen
  - GAC_32 für 32bit
  - GAC_64 für 64bit

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Globaler Assembly Cache (GAC)

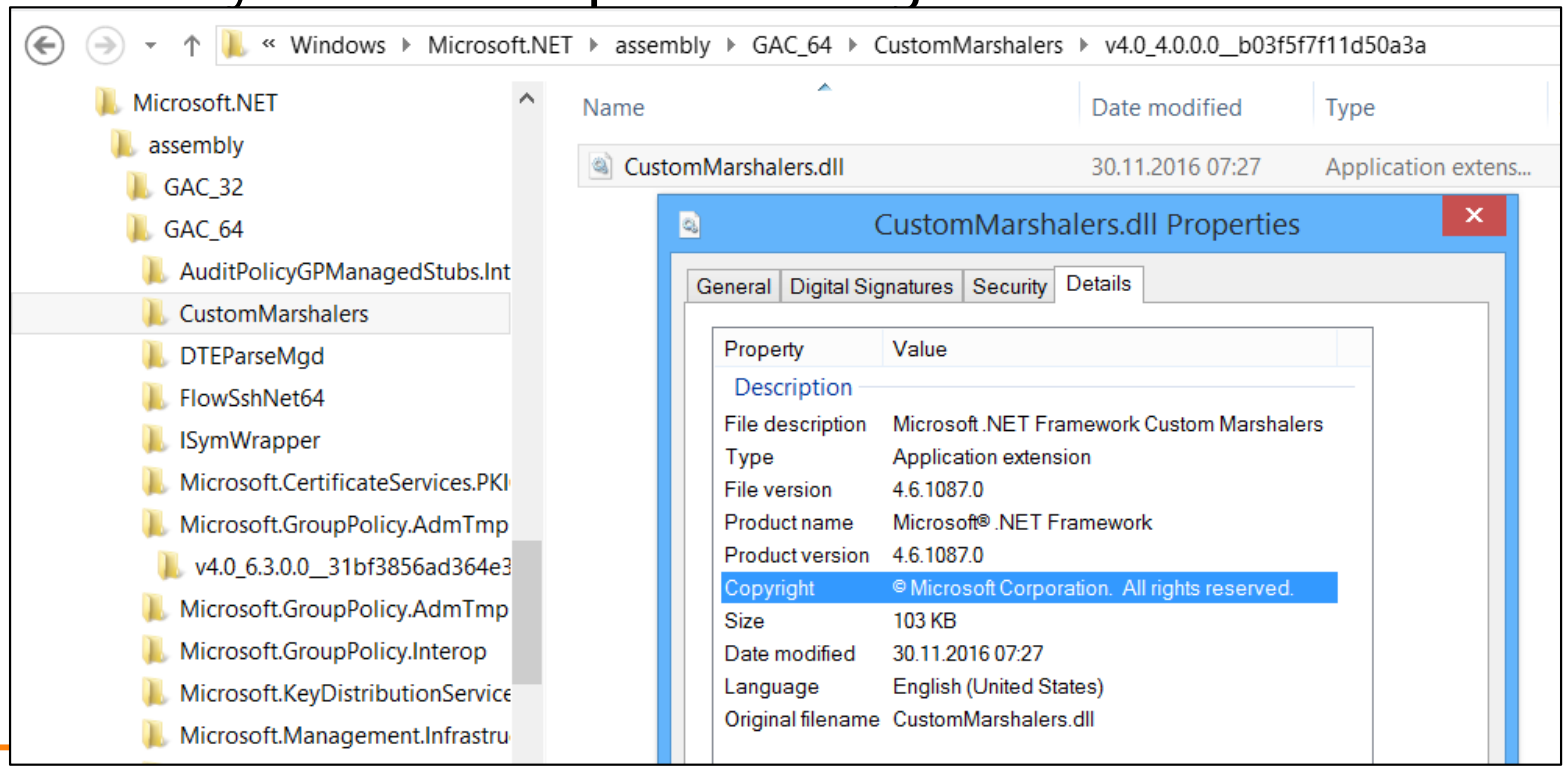
Der GAC-Inhalt

- Pfad

Ab .NET Framework 4: %windir%\Microsoft.NET\assembly

Frühere Versionen: %windir%\assembly

- Assembly Cache im Explorer anzeigen



Globaler Assembly Cache (GAC)

Strong Named Assemblies

- Alle Assemblies, die im globalen Assembly Cache bereitgestellt werden, müssen starke Namen (Strong Name) besitzen.
- Ein Schlüsselpaar für die Signatur erzeugen

```
sn -k mykeys.snk
```

- Beim Hinzufügen einer Assembly werden Integritätsprüfungen für alle Dateien vorgenommen, aus denen die Assembly besteht. Diese Integritätsprüfungen werden vom Cache durchgeführt.
- Strong Names ermöglichen eine Unique Identity für Assemblies

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Globaler Assembly Cache - GAC

Assemblies im GAC bereitstellen

Es gibt mehrere Möglichkeiten eine Assembly im GAC bereitzustellen:

- Die Verwendung eines Installationsprogramms, das für die Zusammenarbeit mit dem globalen Assembly Cache entworfen wurde. Das ist die bevorzugte Option für die Installation von Assemblies im globalen Assembly Cache.
- Die Verwendung des Entwicklertools Global Assembly Cache Tool (Gacutil.exe) aus .NET Framework SDK.
- Das Verschieben von Assemblies in den Cache durch einen Drag & Drop-Vorgang in Windows Explorer.

.NET/C# im Vergleich zu Java

Codes im Vergleich

- Hello World in Java:

```
import System;

public class HelloWorld
{
    public static int main(String[] args)
    {
        out.println("Hello, World");
        return 0;
    }
}
```

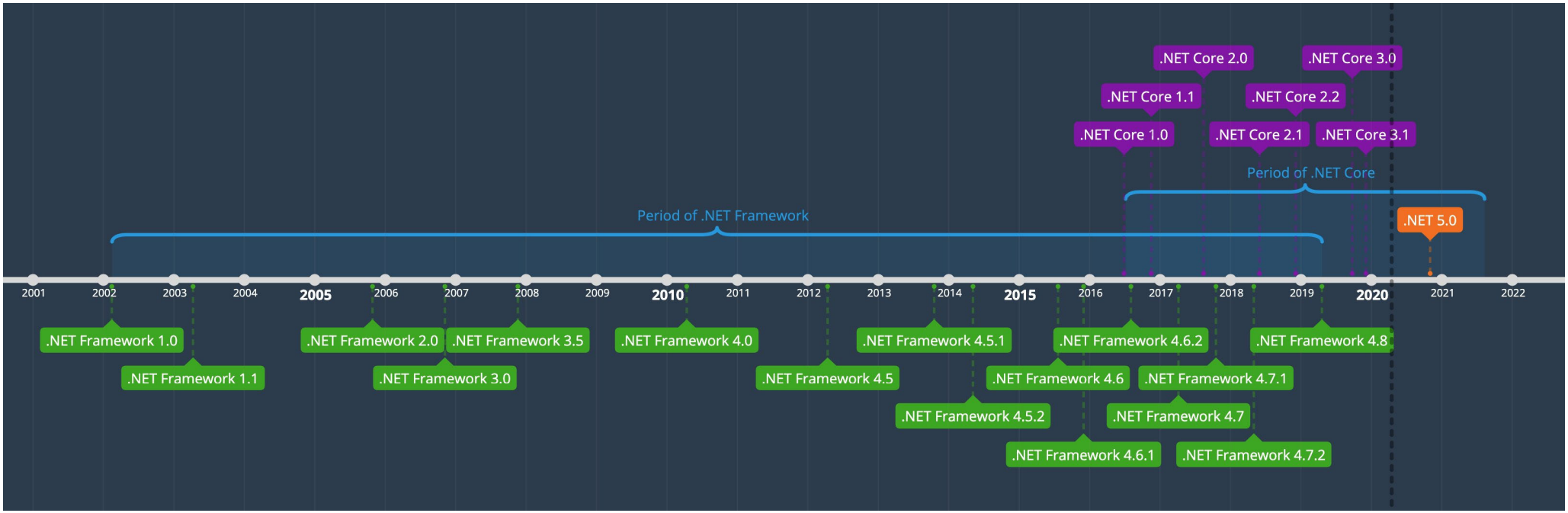
- Hello World in C#:

```
using System;

public class HelloWorld
{
    public static int Main(String[] args)
    {
      Console.WriteLine("Hello, World");
      return 0;
    }
}
```

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

.NET-Zeitachse



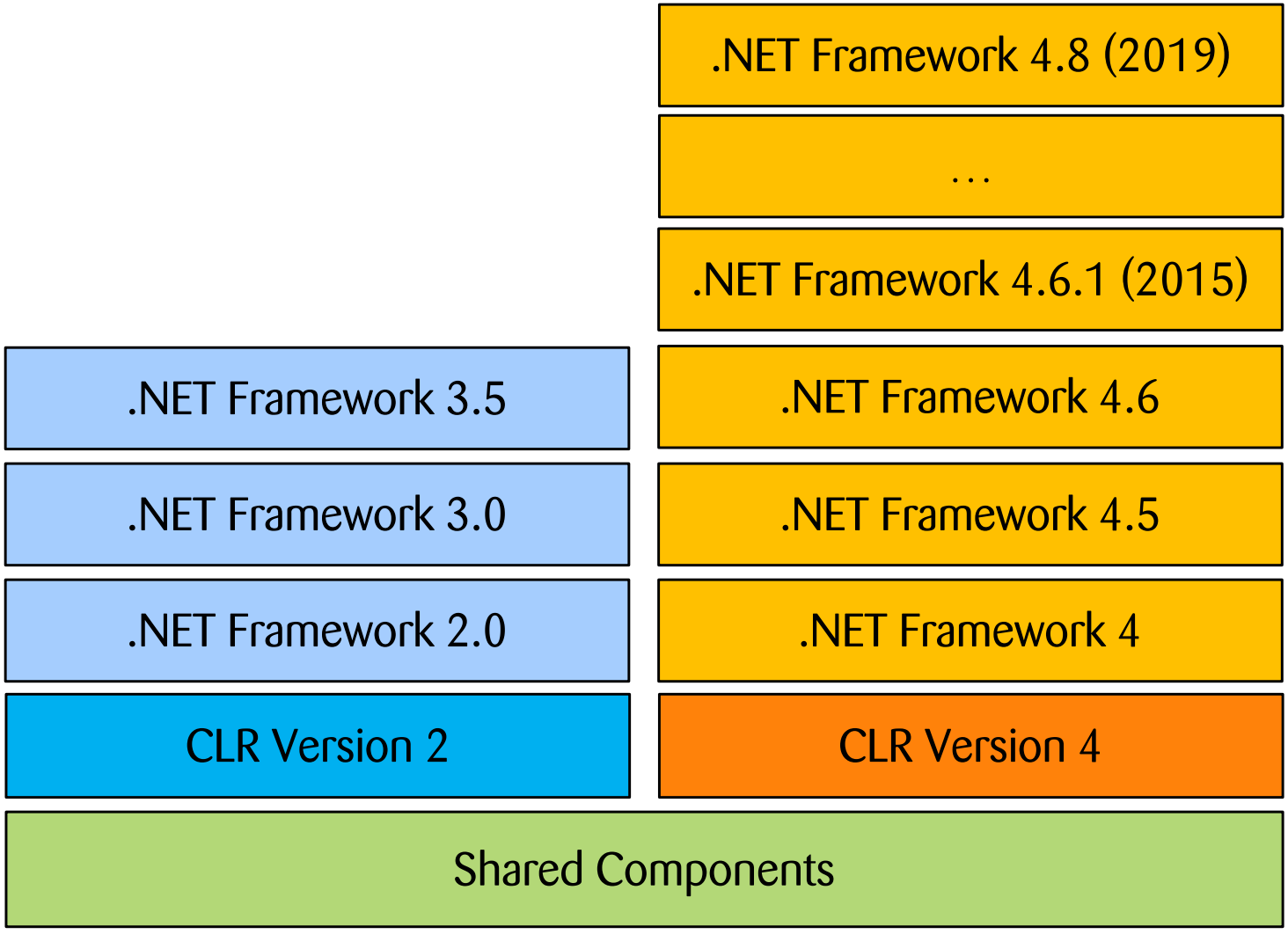
- .NET Framework 1.0 bis 4.8
- .NET Core 1.0 bis 3.1
- .NET 5.0
  - Der Begriff «Core» in «.NET Core» fällt weg; «one dot net»
  - .NET 4.0 existiert nicht, um Verwechslungsgefahr mit .NET Framework 4.x zu vermeiden

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Quelle: Twitter - https://twitter.com/buhakmeh/status/1250127850457440256/photo/1

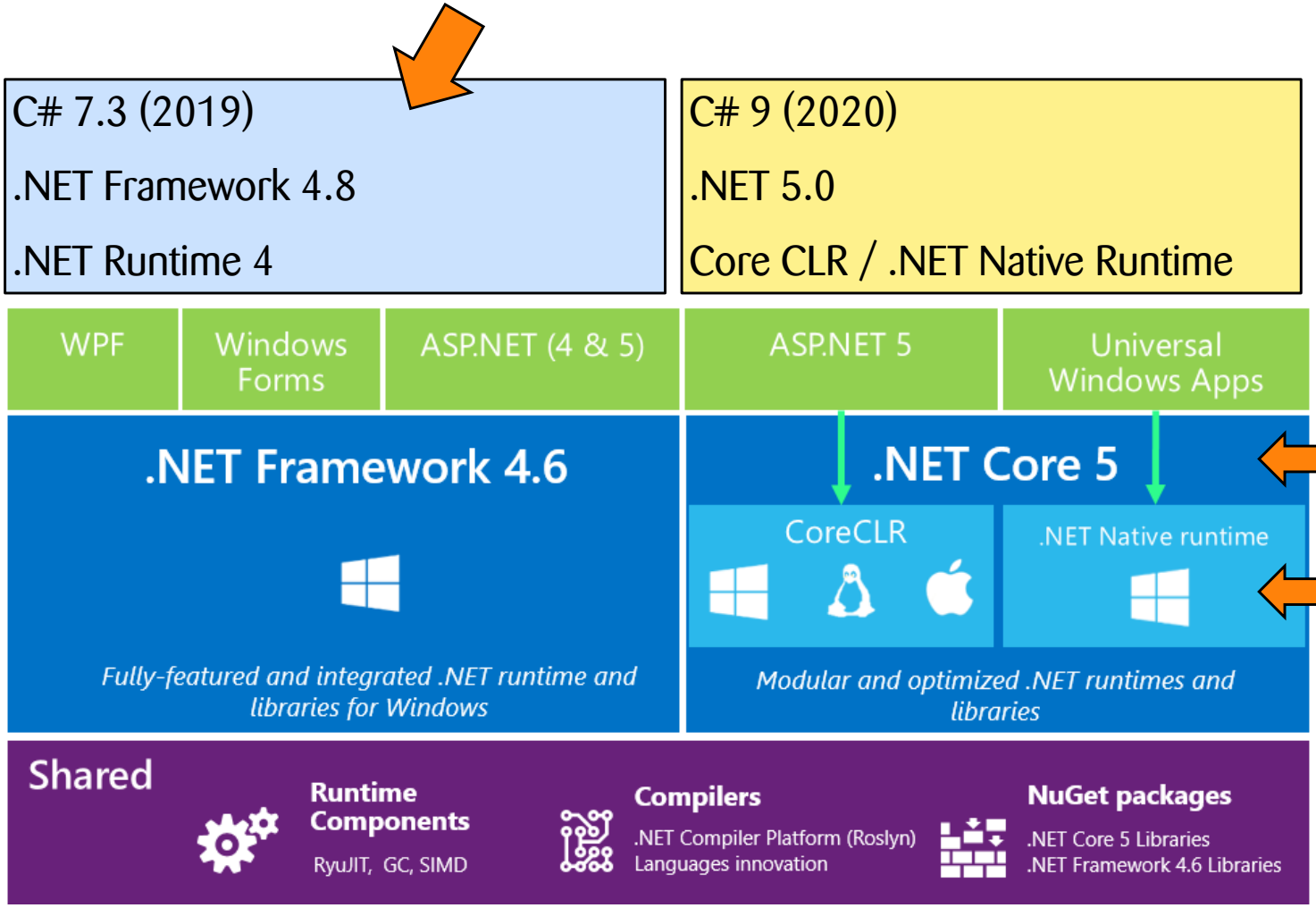
.NET-Versionen

.NET-Versionen 2.0 bis 4.8



.NET-Versionen

Aktuelle Versionen (Stand August 2021)



Neue Plattform .NET Core (reduzierte Funktionalität)

ASP.NET für Mac & Linux

Windows 10

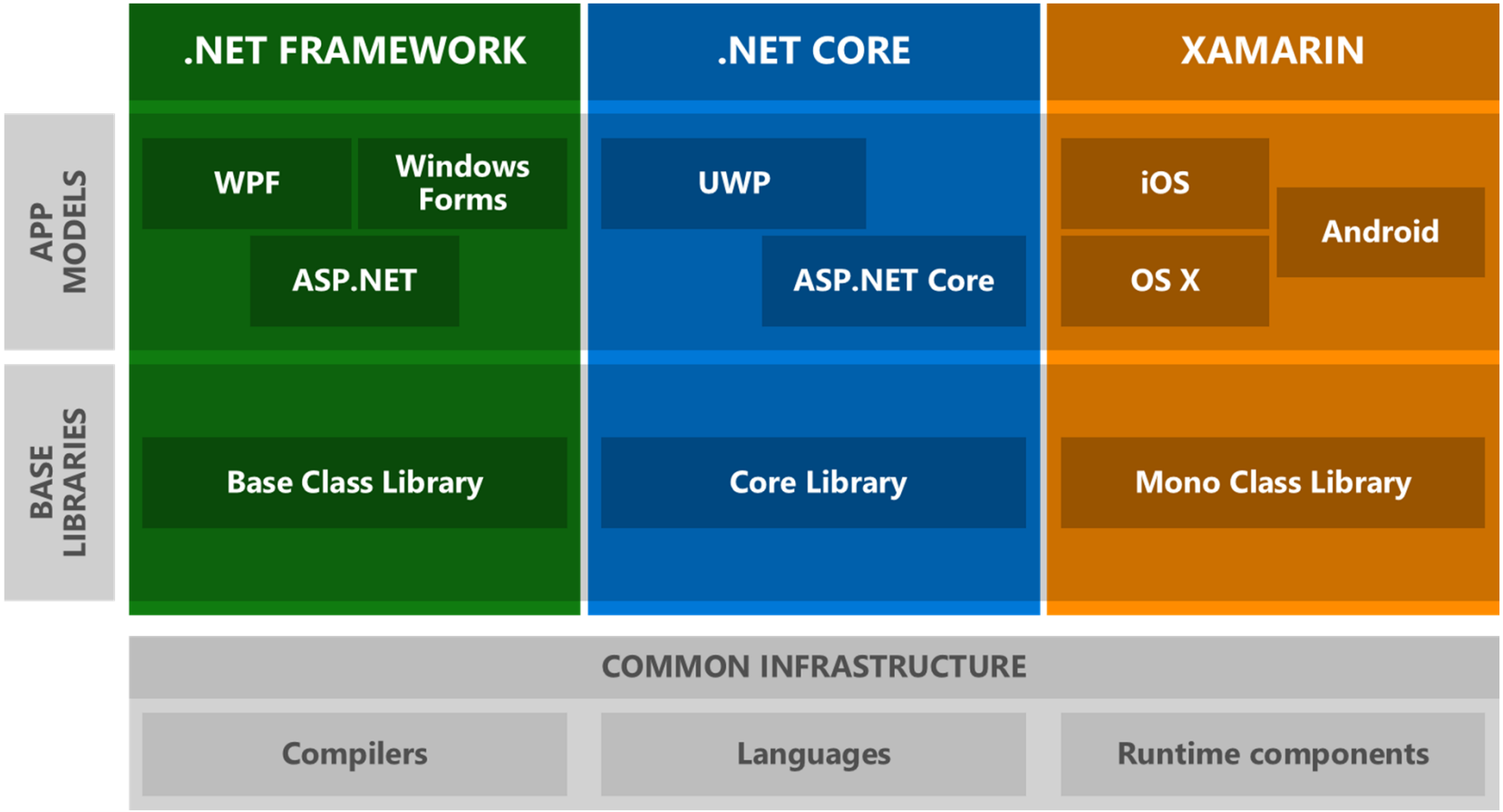
Quelle: What's New for .NET 2015 | Connect("On Demand"); | Channel 9

Quelle: Wikipedia - https://en.wikipedia.org/wiki/.NET https://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_4.8 https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

.NET Standard

Historie: Unterschiedliche Base Libraries

- Machen das Entwickeln einer Applikation für mehrere Zielplattformen schwieriger
- Welche APIs werden gemeinsam durch die Base Libraries unterstützt?

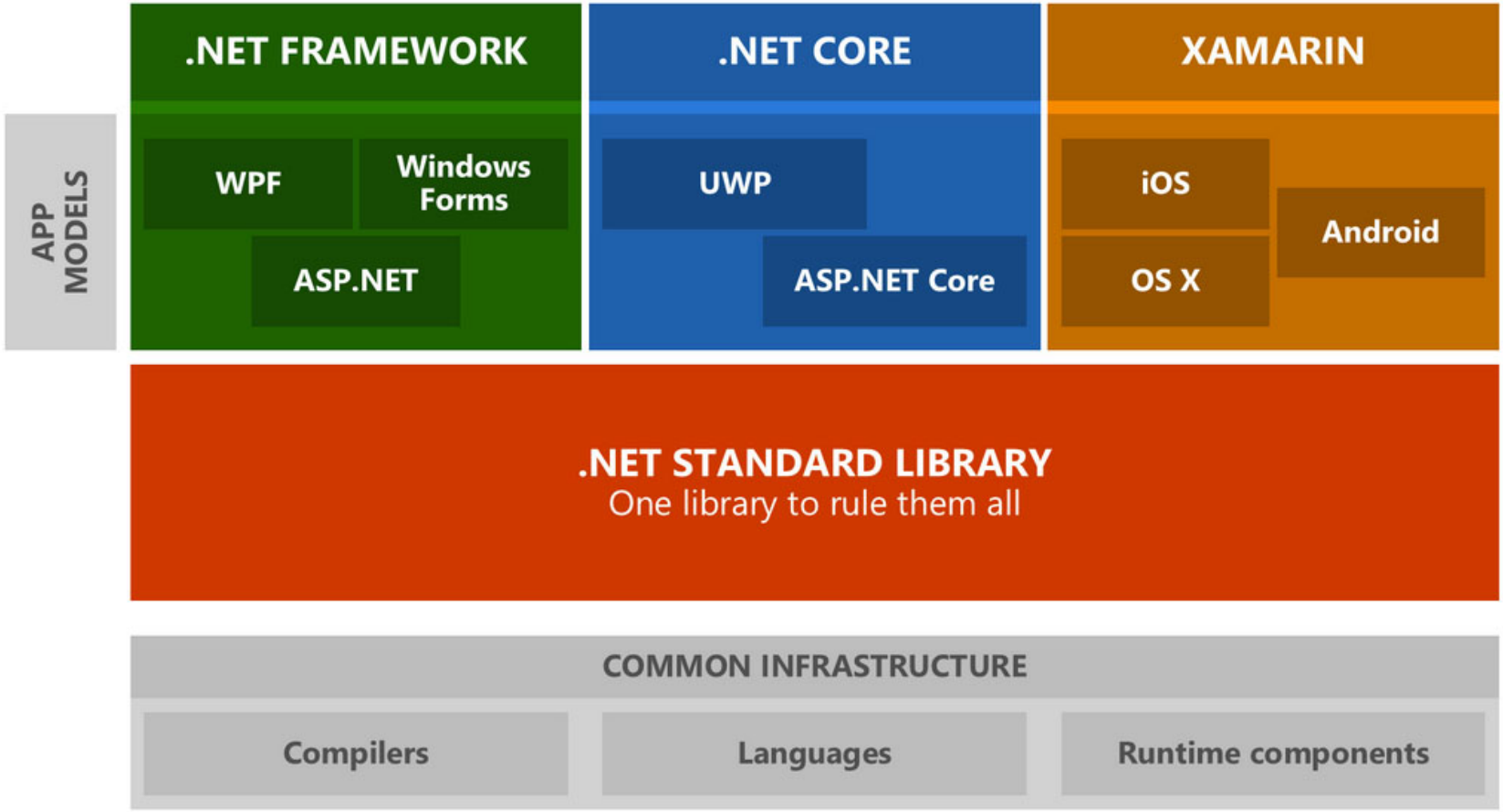


Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

.NET Standard

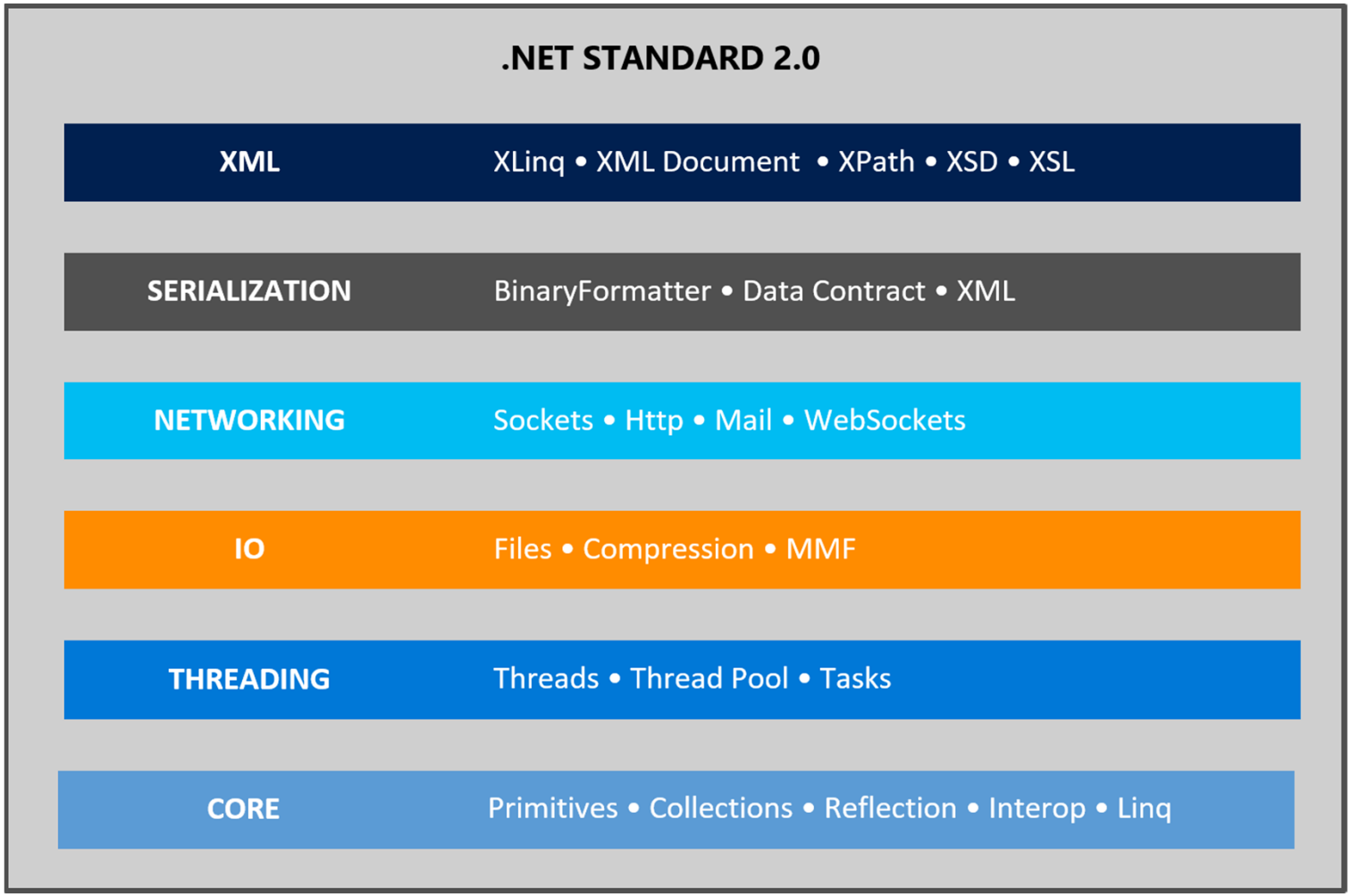
Die Lösung: Eine .NET Standard Library (aktuell v2.1)

- Soll das Entwickeln einer Applikation für mehrere Zielplattformen vereinfachen
- Einfach den kleinsten gemeinsam unterstützten .NET Standard verwenden



.NET Standard

.NET Standard 2.0 definierte APIs



.NET Standard

.NET Standards - Übersicht der .NET und .NET Standard Versionen

- Bsp.: Zielplattformen Windows Phone 8.1 und Xamarin.Android 7.0
- Gemeinsam wird max. .NET Standard 1.2 unterstützt

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	**1.0**	**2.0**
.NET Framework	4.5	**4.5**	**4.5.1**	**4.6**	4.6.1	4.6.1 ~~4.6.2~~	4.6.1 ~~vNext~~	**4.6.1**
Mono	4.6	4.6	4.6	4.6	4.6	4.6	**4.6**	**5.4**
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	**10.0**	**10.14**
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	**3.0**	**3.8**
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	**7.0**	**7.5**
Universal Windows Platform	10.0	10.0	10.0	10.0	**10.0**	vNext	vNext	**vNext**
Windows	8.0	**8.0**	**8.1**					
Windows Phone	8.1	8.1	**8.1**					
Windows Phone Silverlight	**8.0**							

fsbildung

Quelle: Microsoft - https://docs.microsoft.com/en-us/dotnet/standard/net-standard