**Agenda**

**Algemene kennis .NET & C#**

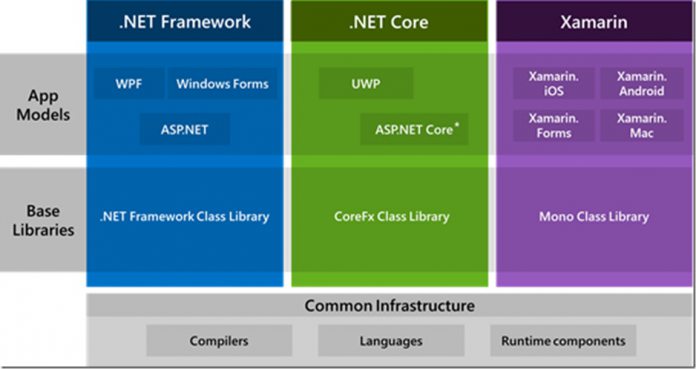
* De .NET stack
  + Wat is .NET?
  + De CLI
    - .NET implementatie (CLR) en MONO
  + CLR
    - IL language
    - Virtuele machine
    - Type system
    - Garbage collection
    - Memory safety
    - Exceptions

**Resources**

[**https://github.com/dotnet/coreclr/blob/master/Documentation/botr/intro-to-clr.md**](https://github.com/dotnet/coreclr/blob/master/Documentation/botr/intro-to-clr.md)

**Intro**

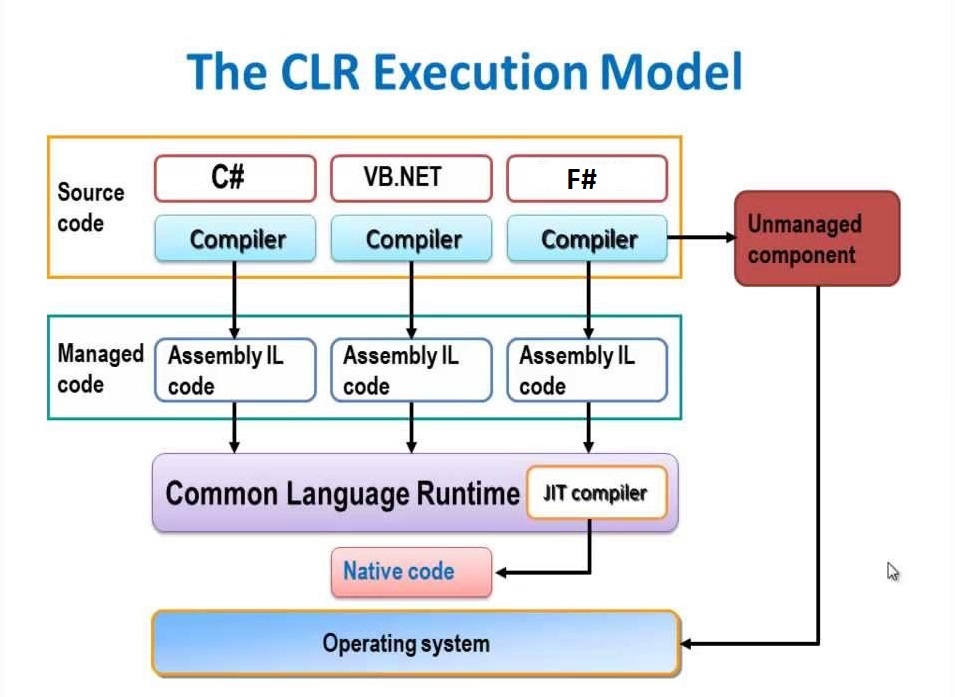
Wat is .NET?  
.NET is een applicatieframework van Microsoft. Het geeft developers de tools om verschillende soorten applicaties te ontwikkelen, zonder dat we ons zorgen hoeven te maken over de low level details van de applicatie. In 2002 kwam de eerste officiële versie uit.

****

De architectuur van .NET bestaat uit een aantal belangrijke zaken.

* De talen (C#, VB.NET en F#).
* De compiler (Roslyn).
* De framework class library
* De CLI

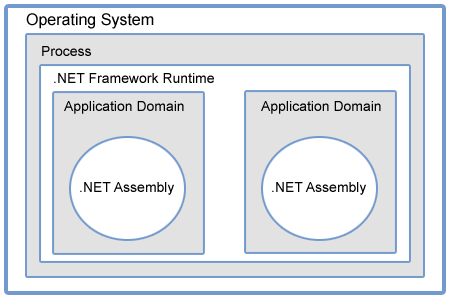
De CLI  
De CLI (Common Language Infrastructure) is de specificatie van de runtime en vormt de kern van de architectuur. Er bestaan meerdere implementaties van de CLI. De Microsoft implementatie heet de CLR. Ook de open source implementatie MONO is populair.  
In het onderstaande overzicht kan je zien hoe de verschillende onderdelen zich tot elkaar verhouden.

****

IL (Intermediate language)

De code die wij schrijven wordt door de compiler omgezet naar IL. Dit is de taal van de CLR. De taal is leesbaar voor mensen. Voordeel van een “tussentaal” is dat je meerdere high level languages zoals C# en VB.NET kan ondersteunen. De CLR zet IL via de JIT (Just In Time) compiler uiteindelijk om in native code.

Virtuele machine  
  
In de onderstaande afbeelding kun je duidelijk zien hoe .NET code (gecompiled tot een assembly) zich verhoud tot de CLR. De CLR voegt een extra niveau van abstractie toe. Het functioneert als een *virtuele machine*. De code die jij schrijft hoeft niets te weten over low level details van het filesystem, maar spreekt slechts een API als File.Open() aan.  
  
We spreken daarom ook vaak van managed code. In talen als C en C++ heb je deze abstractie veel minder.



Functies van de CLR  
<https://github.com/dotnet/coreclr/blob/master/Documentation/botr/intro-to-clr.md#fundamental-features-of-the-clr>

Naast dat onze code draait binnen een virtuele machine, brengt de CLR een aantal belangrijke functies met zich mee.

1. Garbage collector
2. Managed code
3. Memory and type safety
4. OOP
5. Value Types
6. Exceptions
7. Generics
8. Reflection