Trecho 1 – Visão geral da plataforma .net

O Que é C#

C# é uma linguagem de programação orientada a objetos com a qual é possível construir diversos tipos de programas como aplicativos para windows, web, celulares, jogos, entre outros.

Ela roda na plataforma .net Framework

Microsoft .net Framework

.net Framework é um Programa que é capaz de executar outros programas, funciona como um adaptador entre o sistema operacional do computador e o seu programa escrito em C#.

Um dos motivos dele existir é para tornar os seus programas Multi-plataforma, o que significa que ele torna seus programas capazes de serem executados em ambientes diferentes como Windows, Linux ou OSX.

Multi-linguagem (Common Language Specification, Common Type System)

O .net Framework também suporta outras linguagens além do C#, o que o torna multi-linguagem além de multi-plataforma.

Como o .net framework é capaz de funcionar com várias linguagens?

Através da common language specification (CLS)

A CLS é uma documentação que dita o que uma linguagem precisa ter, quais regras ela precisa respeitar para poder ser compatível com o ambiente .net, ou seja, para que um compilador seja capaz de transformar um código escrito nesta linguagem em MSIL (linguagem intermediária).

Outro requisito que uma linguagem precisa respeitar para ser compatível com .net é trabalhar com os tipos de dados especificados no Common Type System.

Compilador – em geral é um programa que transforma uma linguagem de programação em linguagem de máquina, porém, no ambiente .net ele transforma uma linguagem compatível com .net em linguagem intermediária

MSIL – Microsoft Intermediate Language: É uma linguagem gerada pelos compiladores capaz de ser transformada em linguagem de máquina pelo JIT

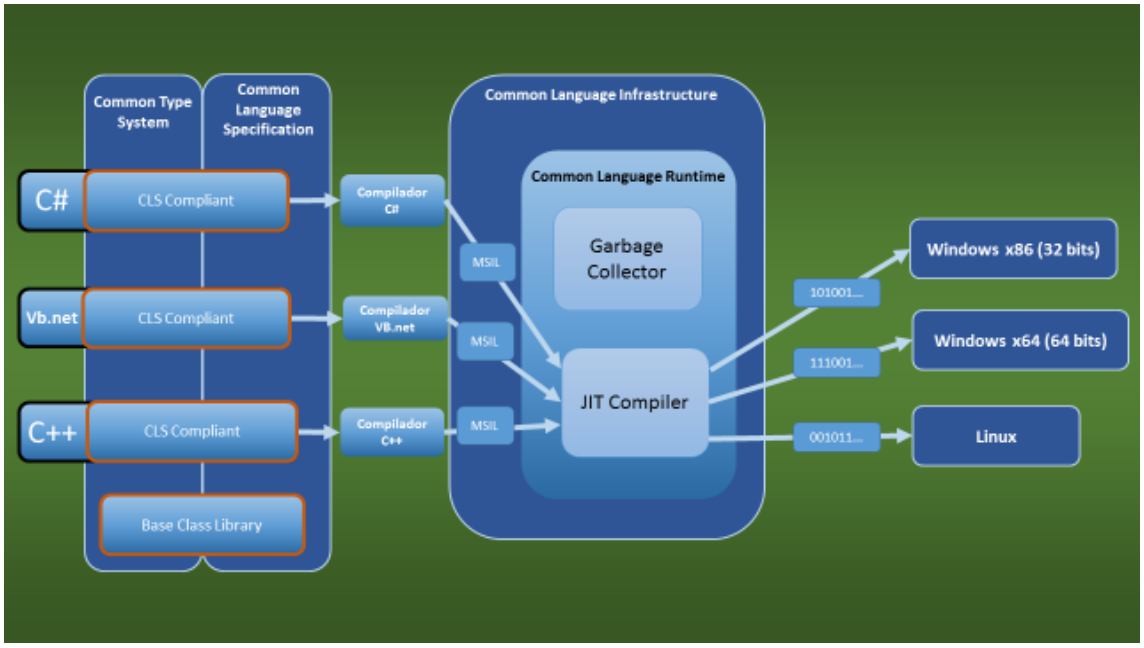
Common Type System – É um conjunto de tipos de dados padronizados que as linguagens devem suportar para serem compatíveis com .net

Execução multi-plataforma (Intermediate Language, Common Language Runtime, JIT)

Como funciona a infraestrutura multi-plataforma do .net?

Common Language Runtime – é o ambiente capaz de executar um programa .net, ele cuida de gerenciamento de memória, bem como da compilação para linguagem de máquina propriamente dita através do JIT

JIT – é um componente do common language runtime que compila a linguagem intermediária na linguagem nativa da arquitetura do processador onde o programa está sendo executado



Gerenciamento de memória (Garbage Collector)

O Garbage collector é um processo automático que roda dentro do CLR responsável por gerenciar a memória disponível para o seu programa, ele remove automaticamente os objetos que não estão mais em uso nas situações onde é identificado que a memória disponível está baixa.

É possível também forçar a chamada manualmente do G.C. porém por ser uma operação muito custosa para a máquina, deve-se utilizar este recurso com moderação.

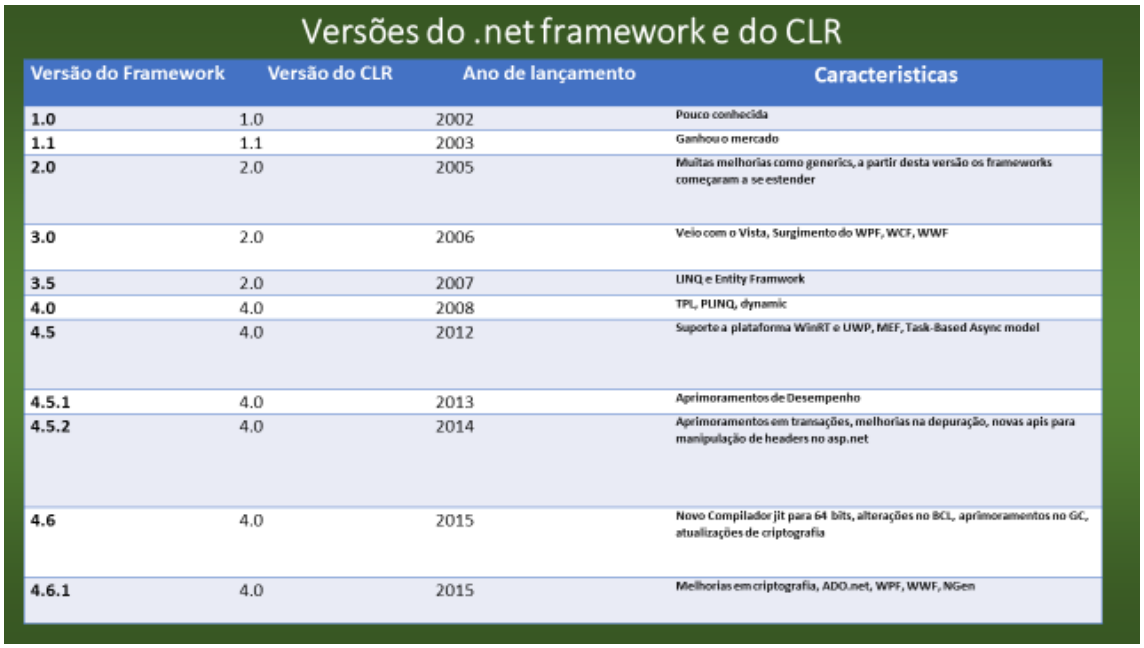
Apesar de ser capaz de identificar os objetos que não estão mais em uso, o GC não é capaz de liberar a memória utilizada por recursos que não são gerenciados pelo .net, como conexões com banco de dados, manipuladores de arquivos e manipuladores de janelas. Estes são conhecidos como recursos não gerenciados, ou seja, que não fazem parte do ecossistema .net.

Posteriormente veremos como trabalhar com este tipo de recurso.

.net Framework Class Library

O .net Framework Class Library é uma biblioteca de componentes prontos que realizam operações frequentemente necessárias como gravar e ler arquivos, manipular bancos de dados, realizar envio de e-mails e muito mais

Esta biblioteca é bem organizada, separando os componentes por assuntos e vamos explorá-la ao longo do curso. É muito importante conhecer os recursos disponíveis na Class Library para evitar "reinventar a roda" uma vez que o uso destes componentes fornece ganho de produtividade e diminui a possibilidade de erros.



Versões do .net framework e do CLR

Versão do Framework	Versão do CLR	Ano de lançamento	Caracteristicas
1.0	1.0	2002	Pouco conhecida
1.1	1.1	2003	Ganhou o mercado
2.0	2.0	2005	Muitas melhorias como generics, a partir desta versão os frameworks começaram a se estender
3.0	2.0	2006	Veio com o Vista, Surgimento do WPF, WCF, WWF
3.5	2.0	2007	LINQ e Entity Framwork
4.0	4.0	2008	TPL, PLINQ, dynamic
4.5	4.0	2012	Suporte a plataforma WinRT e UWP, MEF, Task-Based Async model
4.5.1	4.0	2013	Aprimoramentos de Desempenho
4.5.2	4.0	2014	Aprimoramentos em transações, melhorias na depuração, novas apis para manipulação de headers no asp.net
4.6	4.0	2015	Novo Compilador jit para 64 bits, alterações no BCL, aprimoramentos no GC, atualizações de criptografia
4.6.1	4.0	2015	Melhorias em criptografia, ADO.net, WPF, WWF, NGen

Short-cuts:

- C# é uma linguagem poderosa que está presente em diversos tipos de aplicações e dispositivos;
- O.net framework é um software necessário para rodar uma aplicação em C#;
- O .net framework traz diversos recursos já desenvolvidos frequentemente necessários em programas para facilitar a vida do programador e aumentar sua produtividade;

Por que escolher C# como linguagem?

Já que o .net suporta diversas linguagens, por que adortamos c#?

1. O c# foi criado especificamente para a plataforma .net, diferente de outras linguagens que foram adaptadas ou portadas para serem suportadas na plataforma;
2. Hoje o c# é amplamente adotado pelas empresas no mercado e a demanda por profissionais que conheçam a linguagem é extremamente alta;
3. Além dos aplicativos de linha de comando, Windows e web, hoje em dia o c# é utilizado no desenvolvimento de aplicativos móveis, jogos e até micro controladores;
4. É uma linguagem em constante evolução hoje está na versão 6.0 com previsão para lançamento da 7.0

Relação das versões do C# com as versões do .net

Versão do C#	Versão do .net	Principais Alterações
1.0	**1.0**	-
1.2	**1.1**	-
2.0	**2.0**	Generics, Nullable
3.0	**2.0, 3.0, 3.5**	Propriedades auto implementáveis, lambda, extension methods, Tipo implícito em variáveis local (uso de var)
4.0	**4.0**	Dynamic, generic co e contra variancia
5.0	**4.5**	Métodos assíncronos
6.0	**4.6**	Null Propagator, Roslyn, import static, string interpolation, exception filter (catch.. when)