26 de agosto de 2021

Diego L. Alexandre

Avanade Brasil

**Rua Alexandre Dumas, 2.051Ch. Santo AntonioSão Paulo - SP 04717-004Brasil**

Primeiros passos com net

Decola tech Bootcamp

1. **O que é .NET?**

**Para início de conversa**

**Gabriel Faraday –** nome do facilitador do curso

**Contato –** about.me/gabrielfaraday

**Objetivos da aula**

* **Conhecendo o framework**
  + Passado, presente e futuro do .NET
    - Microsoft iniciou nos anos 70 criando linguagens de programação: **Basic**
    - Nos anos 80, surge o **DOS**, que foi utilizado como OS padrão para computadores IBM
    - Nos próximos anos, a Microsoft atua fortemente na criação do **SO Windows**
    - No fim dos anos 90 a Microsoft tentou consolidar as ferramentas de desenvolvimento (IDES e runtimes) com o **Visual Studio 97,** exemplo: Visual Basic 5, Visual ProFox 5, C++ 5, J++
    - Scott Guthrie criou uma ferramenta web com Java, e a chamou ASP+ (Depois chamou ASP Next e depois ASPX)
    - Jason Zander ajudou na criação de um common runtime para VB e C++ (CLR)
    - Java ia bem! Então a Sun Microsystems fez um acordo para a Microsoft não mexer mais com Java.
    - Com isso, Anders Hejlsberg começou a trabalhar no C#
    - Ano 2000 a Microsoft lança o novo ambiente de desenvolvimento chamado .NET 1.0 – inicialmente chamado de Next Generation Windows Services (NGWS)
    - 2001 Miguel de Icaza começa a trabalhar no projeto Mono, uma reimplementação black box do .NET, open Source e multiplataforma.(ele não trabalhava para a Microsoft)
    - 2002 Lançamento do Visual Studio .NET com C# 1.0, conhecido como 21 linguagens, 1 plataforma: C#.net, C++.net, VB.net, entre outros.
    - 2003 Lançamento do .NET 1.1 com **Visual Studio 2003.** Trabalham em melhorias na CLR para lançar o CLR2
    - 2005 Lançamento do .NET 2.0 com C# 2.0 no **Visual Studio 2005**
    - Microsoft começa a atingir o seu objetivo inicial, inclusive evoluindo na web.
    - 2007 e 2008 Lançamento do .NET 3.5 com C# 3.0 no Visual Studio 2008, com Silverlight, WPF e WCF
    - Microsoft contrata um time de pessoas com uma pegada open source e começam a atuar na criação do ASP.NET MVC.
    - Começam a falar em Windows Azure.
    - 2010 Lançamento do .Net 4.0 com C# 4.0 no **Visual Studio 2010**, também com F#
    - Microsoft lança comercialmente o Azure
    - Anders Hejlsberg começa a trabalhar no Typescript
    - 2011 Miguel de Icaza inicia o Xamarin, basicamente ele desenvolvia e C# aplicativos que rodam em Android e iOS
    - 2012 Lançamento do .NET 4.5 com C# 5.0 no **Visual Studio 2012**
    - Lançamento do Typescript
    - 2013 Lançamento do .NET 4.5.1 no **Visual Studio 2013**
    - Início do Roslyn, um novo compilador para C# e VB.NET
    - Microsoft continua atuando mais na frente do Js e aumenta também a incorporação de ferramentas open source ao ambiente.
    - Já temos aqui o ASP.NET mais consolidado com MVC, Web API e SignaIR mas tudo ainda muito Windows.
    - 2014 Satya Nadella se torna CEO da Microsofte direciona o foco da empresa para cloud.
    - Criação do .NET Foundation para gestão de projetos open source.
    - 2014 O Windows Azure passa a se chamar Microsoft Azure
    - É introduzido o conceito do ASP.NET vNext, posteriormente chamado de ASP.NET Core
    - 2015 Lançamento do .NET 4.6 com C# 6.0 no Visual Studio 2015
    - Lançamento do **Visual Studio Code**
    - Microsoft adquire a Xamarin e adiciona o produto como parte de sua stack .NET e projetos open source
    - Lançamento do Visual **Studio for Mac**
    - **2016**  Lançamento do .Net Core 1.0 – Totalmente novo, open source e multiplataforma.
    - **2017** Lançamento do .NET Framework 4.7 com C# 7.0 no **Visual Studio 2017**
    - Lançamento do .NET Core 2.0 com C# 7.0 no **Visual Studio 2017**, **Visual Studio Code** ou **Visual Studio for Mac 2017**
    - 2019 Lançamento do .NET Framework 4.8 com C# 7.3 no **Visual Studio 2019**
    - 2019 Lançamento do .NET Core 3.0 com C# 8.0 no **Visual Studio 2019**, **Visual Studio Code** ou **Visual Studio for Mac 2019**
    - 2020 .NET Framework está pronto na versão 4.8! E deixa de ser evoluído – junto com ele WCT e ASP.NET Webforms.
    - Previsto o lançamento do .NET 5
  + O que é, Como e Onde usar o .NET
    - O que é?
      * O .NET é uma estrutura para desenvolvimento de software criada pela Microsoft
      * Uma aplicação .NET é desenvolvida para e roda em uma das seguintes **implementações** do .NET: **.NET Core**, **.NET Framework**, **Mono**, **Universal Windows Platform (UWP)**
    - Como?
      * Cada implementação inclui outras mais .NET runtimes (ambientes de execução): **.NET Core**: CoreCLT e CoreRT, **.NET Framework**: CLR, **Mono**: Mono Runtime, **UWP:** .NET Native
      * Atualmente a Microsoft desenvolve e suporta 3 linguagens para .NET: **C#, F# e VB**
    - Onde?
      * **Desktop**: WPS, Windows Forms, UWP
      * **WEB**: ASP.NET
      * **CLoud**: Azure
      * **MOBILE**: Xamarin
      * **GAMING**: Unity
      * **IoT**: ARM32, ARm64
      * **AI**: ML NET, NET for Apache Spark
      * **Tools**: Visual Studio, Visual Studio for Mac, Visual Studio Code, Command Line (CMD)
  + Quem usa o .NET
    - Vidalink, Licks Attorneys, Casas Bahia, Saraiva, Petrobras, Takeda, CTC, Rodobens, Press Ganey, Techs, Unimed, Gol, Azul, Walmart Argentina, Itau, Magalu, stackoverflow, Microsoft
    - Links úteis:
    - Download .NET: <https://dotnet.microsoft.com/download>
    - Documentação do .Net: <https://docs.microsoft.com/en-us/dotnet/>
    - Net Foundation: https://dotnetfoundation.org/

1. **Iniciando com o .NET**

* Preparando o ambiente
  + Download do .NET
  + Instalando o Visual Studio Code
* Conhecendo o CLI do .NET
  + Comandos úteis: dotnet - - version, dotnet - - help
  + Comando **dotnet - - help**
    - Ele te dá algo como se fosse a documentação da interface do .NET, na linha de comando.
  + Comando **dotnet - - version**
    - Mostra a versão do .NET que está instalado.
  + Comando **dotnet - - info**
    - Mostra informações do .NET Core
  + Comando SDK nuget
    - Ele é um gerenciador de pacotes para dotnet.
  + Comando SDK test
    - Ele realiza os unit tests para projeto .NET
  + Comando SDK tool
    - Ele permite que você instale e gerencie ferramentas de extensão para .NET.
  + bbbbrrrrrlll
* Criando uma aplicação console
  + Use o comando **dotnet new console –n nome\_arquivo**
  + No exemplo da aula, foi criado o arquivo **dotnet new console –n DigitalInnovationOne**
  + Comando **code .**(code espaço ponto) ele já vem configurado na máquina e vai abrir o conteúdo da página no **Visual Studio Code**
  + Comando **explorer .** (explorer ponto) ele abre a pasta do explorer aonde está o arquivo/projeto
  + Comando **dotnet build** testa/analisa/cria as condições antes de rodar o programa. Bom para checar se está tudo ok antes de jogar no run.
  + Comando **dotnet run** ele roda o programa

1. **Conhecendo o C#**
   1. **O que é C#?**

É uma linguagem elegante, orientada a objeto e fortemente tipada.

A sintaxe C# é similar a do C, C++ ou Java

Suporta os conceitos de encapsulamento, herança e polimorfismo (OO)

Os programas C# são executados no .NET, que inclui: CLR (Common Language Runtime), Conjunto unificado de bibliotecas de classes.

Atualmente o compilador do C# é o **Roslyn**

* 1. **Como funciona?**

O código-fonte do C# é compilado em uma linguagem intermediária (IL).

O código e os recursos de IL são armazenados no disco em um arquivo executável chamado assembly, geralmente com uma extensão **.exe** ou **.dll**

Quando o programa C# é executado, o assembly é carregado no CLR

Em seguida o CLR executará a compilação *just in time* (JIT) para converter o código IL em instruções de máquinas nativas.

O CLR também fornece outros serviços: Garbage Collector, Exception Handler, Resources Manager

Além dos serviços de tempo de execução, o .NET também inclui uma extensa biblioteca com milhares de classes que fornecem uma ampla variedade de funcionalidades úteis, desde entrada e saída de arquivos, manipulação de cadeias de caracteres, análise XML, etc.

* 1. **Estrutura de programa**

Os principais conceitos organizacionais em C# são:

* **programas**
  + São os projetos em si.
* **namespaces**
  + estão dentro do programa
* **tipos**
* **membros**
* **assemblies**

Programas em C# consistem em um ou mais arquivos

Os programas declaram tipos, que contêm membros e podem ser organizados em namespaces

As classes e interfaces são exemplos de tipos

Campos, métodos, propriedades e eventos são exemplos de membros.

Quando os programas em C# são compilados, eles são fisicamente empacotados em assemblies

Os assemblies geralmente tem a extensão de arquivo .exe ou .dll, dependendo se são aplicações ou bibliotecas

using *System*;

namespace *DigitalInnovationOne*

{

    class Program

    {

        static *void* Main(*string*[] *args*)

        {

*int* numeroDeVezes = 5;

            for (*int* i = 0; i < numeroDeVezes; i++)

            {

                Console.WriteLine($"Bem vindo ao curso de .Net {i}");

            }

        }

    }

}

Github do professor Gabriel faraday:

<https://github.com/gabrielfbarros/csharp-examples>

1. **Conhecendo variáveis e instruções**

Tipos de valor

Numéricos: **sbyte**, **short**, **int**, **long**, **byte**, **ushort**, **uint**, **ulong**

Caracteres Unicode: **char**

Pontos Flutuantes: **float, double, decimal**

Booleano: **bool**

**Enum, structs,** tipos **nullable** (Exemplo int?)

Tipos de Referência

Variáveis de tipos de referência armazenam referência a seus dados.

É possível que duas variáveis façam referência ao mesmo objeto e, portanto, que operações em uma variável afetem o objeto referenciado pela outra variável.

Tipos de Classe: **class, object, string**

Tipo de Arrays: **int[], int[,]**, etc...

**interface, delegate**

**Instruções**

Ações de um programa são expressas usando instruções

**{**

**Um bloco permite que várias instruções sejam descritas sejam descritas em contexto**

**}**

* **Declarações de variáveis e constantes locais**
* **If**
* **Switch**
* **While**
* **Do**
* **For**
* **Foreach**
* **Break**
* **Continue**
* **Return**
* **Throw**
* **Try... catch... finally**
* **Using**

1. **Classes e objetos essenciais em C#**

**Classes são os tipos mais fundamentais em C#**

**Mas o que é uma Classe?**

Uma classe é uma estrutura de dados que combina estado (campos) e ações (métodos)

**O que são Objetos?**

Objetos são instâncias de uma classe.

As classes suportam herança e polimorfismo, mecanismos pelos quais as classes derivadas podem estender e especializar as classes base.

public class Ponto

{

public int x,y;

public Ponto(int x, int y)

{

this.x = x;

this.y = y;

}

}

Instâncias de classes(objetos) são criadas usando o operador new, que aloca memória para uma nova instância, chama um construtor para iniciar a instância e retorna uma referência à instância.

Ponto p1 = new Ponto(0,0);

Ponto p2 = new Ponto(10,20);

A memória ocupada por um objeto é recuperada automaticamente quando o objeto não está mais acessível. Não é necessário nem possível desalocar implicitamente objetos em C#.

**Membros**

Os membros de uma classe podem ser estáticos ou membros da instância.

Os membros estáticos pertencem a classe e membros de instância pertencem ao objeto.

Exemplos de membros:

Constantes, variáveis, métodos, propriedades, construtores, etc.

**Acessibilidade**

Cada membro de uma classe tem uma acessibilidade associada, que controla as regiões do texto, do programa que podem acessar o membro

Podem ser:

Public

Protected

Internal

Private

**Herança**

Uma declaração de classe pode especificar uma classe base, herdando os membros public, internal e protected da classe base.

Omitir uma especificação de classe base é o mesmo que derivar do tipo object.

**Métodos**

Um método é um membro que implementa uma computação ou ação que pode ser executada por um objeto ou classe.

Os métodos podem ter uma lista de parâmetros, que representam valores ou referências de variáveis passados para o método, e um tipo de retorno, que especifica o tipo do valor calculado e retornado pelo método.

Boa prática: nome de método tem que ser um verbo, pois indica uma ação.

Como aplicar classes e objetos em projetos

1. **Trabalhando com structs, interfaces e enums**

**O que são Structs**

Como as classes, as Strucs são estruturas de dados que podem conter membros de dados e membros de ação, mas, diferentemente das classes, as Structs são tipos de valor e não requerem alocação de heap

Heap = local da memória aonde realmente está alocado o dado na memória

Uma variável de um tipo de struct armazena diretamente os dados de estrutura, enquanto uma variável de um tipo de classe, armazena uma referência a um objeto alocado na memória.

O uso de structs em vez de classes para pequenas estruturas de dados pode trazer uma grande diferença no número de alocações de memória.

Construtores de structs são chamados com o operador new, semelhante ao construtor de classes mas em vez de alocar dinamicamente um objeto no heap gerenciado e retornar uma referência a ele, um construtor struct simplesmente retorna o próprio valor struct (normalmente em um local temporário na stack), e esse valor é copiado conforme necessário.

**O que são interfaces**

Uma interface define um contrato que pode ser implementado por classes e structs.

Uma interface pode conter métodos, propriedades, eventos e indexadores.

Uma interface não fornece implementações dos membros que define – apenas suas “assinaturas”.

As interfaces podem empregar herança múltipla.

**Enums**

Um enum é um tipo de valor distinto com um conjunto de constantes nomeadas.

Você define enumerações quando você define um tipo que pode ter um conjunto de valores discretos.

Eles usam um dos tipos de valor integral como armazenamento subjacente. Eles fornecem significado semântico aos valores discretos.

1. **Uma síntese do que é .NET**

**Revisão introdutória**

O que é .Net (Passado, Presente e Futuro do .Net)

O que é, Como e Onde usar .Net

Quem usa .Net

Ambiente de desenvolvimento

- Preparando o ambiente

- Conhecendo a CLI do .Net

- Criando uma aplicação Console

Conhecendo o C#

- O que é o C#

- Como funciona

- Estrutura de programa

**-** Tipos e variáveis

**-** Tipo de valor

**-** Tipo de referência

Instruções

- if - break

- switch - continue

- while - return

- do -throw

- for - try ..catch.. finally

- foreach - using

- Arrays

- Classes e Objetos

- Membros

- Acessibilidade

- Herança

- Métodos

- Structs

- interfaces

- Enums