

# TI 2

.NET Core e ASP.NET Core

# .NET - História

- A versão 1.0 da .NET Framework foi introduzida em 2002, estando disponível apenas para Windows.
- Projetos como o [Mono](#) tinham como objetivo trazer a .NET Framework a macOS e Linux (mas não **era** suportado oficialmente pela Microsoft).
- Em 2016, a Microsoft introduziu o **.NET Core**, uma versão **open-source** do .NET, com o objetivo de ser **multi-plataforma**.
- A versão 4.8 da .NET Framework é a última versão que vai receber novas funcionalidades (só vai receber bug fixes).

# .NET Core

- .NET Core é a versão multi-plataforma da .NET Framework da Microsoft.
- A vasta maioria das classes, funções e interfaces da .NET Framework são iguais no .NET Core.
  - Exceções: Coisas específicas a Windows (e são cada vez menos com cada versão)
- Tal como a .NET Framework, as dependências são instaladas com o NuGet.
- Versão atual: 2.2
  - Versão 3 em *preview*, com melhorias ao nível de IoT, Machine Learning, e aplicações *desktop*.

# .NET Core - Instalação

- Se se tem o Visual Studio  $\geq$  2017, não é preciso fazer nada (excepto certificar que está atualizado).
- Em Linux e macOS: Instalar o [.NET Core SDK](#) (ver instruções sobre como instalar no site)
  - Pode-se instalar também o .NET Core SDK em Windows, mesmo que não se tenha o Visual Studio instalado.
- Para Visual Studio Code, seguir [este artigo](#), e opcionalmente ver [este vídeo](#) para fazer o set up.
- Para macOS, pode-se usar o [Visual Studio for Mac](#).

ASP.NET Core

# ASP.NET Core

- O ASP.NET Core é a framework da Microsoft para desenvolver aplicações web.
- 95% dos conceitos são iguais ao ASP.NET Framework (MVC):
  - Controllers, Views, Models, Entity Framework, Identity, Model Binding, Validação, ...  
(Podem existir algumas diferenças nestes componentes, mas são poucas)
- A principal diferença é que foi concebido para ser multi-plataforma (com o .NET Core).

# ASP.NET Core e ASP.NET Framework - Diferenças

Funcionalidade	ASP.NET Core	ASP.NET Framework
Web Forms (ASPX)	Ausente, substituído por <a href="#">Razor Web Pages</a>	Presente, opcional.
Web.config	<a href="#">Substituído pelo ficheiro appsettings.json, entre outros</a>	Obrigatório
Configuração no arranque	<a href="#">Main() e classe Startup</a>	Global.asax, App_Start
Guardar ficheiros de forma privada	Tudo o que não está na pasta <a href="#">wwwroot</a> é privado	Os ficheiros privados devem ir para o App_Data
Hosting e deployment	Opcional, pode-se usar Apache, nginx, ou o programa diretamente	Obrigatório usar o IIS

# ASP.NET Core e ASP.NET Framework - Diferenças

Funcionalidade	ASP.NET Core	ASP.NET Framework
Controllers	Presente quando se usa <a href="#">MVC</a>	Presente quando se usa MVC
API Controllers	<a href="#">Usam-se os mesmos controllers que os de MVC</a> , unificados	É recomendado usar o <a href="#">ASP.NET Web API</a> , mas é limitada
Ficheiros	Usam a interface <a href="#">IFormFile</a>	Usam a classe <code>HttpPostedFileBase</code>
Uso de BDs e outros serviços	Recomenda-se usar <a href="#">Dependency Injection</a> (DI)	DI limitada, objetos instanciados diretamente (new).
Sessões	Desligadas por defeito, <a href="#">têm que ser ligadas</a> no Startup	Ligadas por defeito



# Diferença principal (ou, dica para o Google)

- Se se está a fazer uma pesquisa específica ao .NET Core, ou a alguma das seguintes funcionalidades:
  - ASP.NET Core
  - Entity Framework Core (ou EF Core)
  - ASP.NET Identity Core
  - .NET Core
- Deve-se meter “Core” na pesquisa (como se vê na lista acima):
  - “ef core migrations”
  - “asp.net core configure identity”
- Caso contrário, poderão aparecer resultados para o ASP.NET Framework, que poderão ser incompatíveis.

# ASP.NET Identity Core e Identity - Diferenças

Funcionalidade	Identity Core	Identity
Identity no template com autenticação	Escondida por defeito, <a href="#">mas pode-se fazer o scaffold dos ficheiros</a>	Todas as classes e views estão disponíveis no projeto quando este é criado
Classe do User personalizada	<a href="#">Criar uma classe que estende IdentityUser e mudar a classe da base de dados para a referenciar.</a> Poderão ser necessárias migrações para refletir as alterações.	A classe é criada no projeto por defeito.

# EF Core vs. Entity Framework - Diferenças

Funcionalidade	EF Core	Entity Framework
Migrações	<a href="#">Apenas manuais</a> , recomenda-se apagar a pasta das migrações e BD (se possível) e recriar a migração	Manuais e automáticas (não se recomenda usar migrações automáticas em produção)
Seed	<a href="#">Seed de dados possível com HasData</a> , mas é limitado. Cenários mais complexos (ex: autenticação) ainda requerem <a href="#">classes dedicadas</a>	Possível na classe de configurações das migrações.
Convenções (ex: ON DELETE CASCADE)	<a href="#">Tem que se desligar no método de configuração</a> para cada relacionamento	Pode-se desligar removendo a convenção.

# EF Core vs. Entity Framework - Diferenças

Funcionalidade	EF Core	Entity Framework
Carregar dados relacionados	<p><u>Por defeito, é necessário usar o <a href="#">Include</a></u>, caso contrário os valores das propriedades serão NULL ou arrays vazios.</p> <p>Alternativamente, <u><a href="#">configura-se para ser automático</a></u>.</p> <p>(Recomendo o Include se se sabe os dados necessários <i>a priori</i>, pode resultar em queries SQL mais otimizadas).</p>	Feito automaticamente por defeito, com “classes proxy”.
Atualizar uma entidade (o SaveChanges é igual)	<code>db.Tabela.Update(entidade);</code>	<code>db.Entry(entidade).State = EntityState.Modified;</code>

# EF Core vs. Entity Framework - Diferenças

Funcionalidade	EF Core	Entity Framework
Relacionamentos M-N	<a href="#">É obrigatório criar a classe do relacionamento</a> , com as chaves primárias e forasteiras.	A classe do relacionamento é opcional, desde que não sejam necessários mais campos que as chaves primárias e forasteiras.
Chaves primárias compostas	<a href="#">É necessário usar o HasKey no método de configuração</a> (ver segundo snippet da secção “Fluent API”).	Usam-se os atributos [Key] e [Column(Order = n)] nas propriedades afetadas.
Usar outras bases de dados (depois de instalar os NuGet packages necessários)	Configurado no <a href="#">Startup</a> e appsettings.json (mudar a Connection String).	Configurado no Web.config.

Criação de projetos

# Criação de projetos sem Visual Studio

- No .NET Core, é possível usar a linha de comandos para criar projetos.
- Para isso, usa-se o programa “dotnet”.
- O “dotnet” é usado também para:
  - Scaffolding
  - Migrações
  - Executar aplicações
  - Correr *scripts*

# Ferramenta “dotnet” - coisas úteis

Comando	Descrição
<code>dotnet [comando] --help</code>	Mostra a ajuda do [comando].
<code>dotnet new -l</code>	Mostra os “templates” disponíveis (ex: mvc, console, ...).
<code>dotnet new mvc -o ProjetoMvc</code>	Cria um projeto MVC na pasta ProjetoMvc (convém usar o “cd” para entrar na pasta do projeto).
<code>dotnet restore</code>	Restaura os NuGet packages do projeto da pasta atual.
<code>dotnet build</code>	Compila o projeto da pasta atual.
<code>dotnet run</code>	Executa o projeto da pasta atual.
<code>dotnet watch run</code>	Automaticamente reinicia o projeto quando existem alterações.



# Ou seja, para criar e iniciar um projeto...

# Criar projeto com autenticação (dotnet new mvc --help para detalhes)

> **dotnet new mvc --auth Individual -o ProjetoTeste**

# Ir para a pasta do projeto

> **cd ProjetoTeste**

# Executar o projeto

> **dotnet run**

# Abrir o browser em <http://localhost:5000> (poderá sair erro de https).

# Pode-se abrir o ficheiro .csproj no Visual Studio, ou a pasta no Visual Studio Code.

# Ferramenta “dotnet” - (mais) coisas úteis

Comando	Descrição
<code>dotnet add package [nome]</code>	Instala o <a href="#">package NuGet</a> com o nome [nome] (“Manage NuGet packages for solution” continua disponível)
<code>dotnet ef migrations add [Nome]</code>	Adiciona uma migração (“Add-Migration” e “Enable-Migrations” continuam disponíveis no Visual Studio).
<code>dotnet ef database update</code>	Aplica as migrações na base de dados. (“Update-Database” continua disponível no Visual Studio).
<code>dotnet ef migrations remove</code>	Desfaz a última migração criada (na BD) e apaga-a (do disco)
<code>dotnet ef database update 0</code>	<b>Reverte todas as migrações.</b>
<code>dotnet publish -c Release -o [pasta]</code>	Compila a aplicação para ser colocada num servidor em produção. O output fica na pasta [pasta].

# Dependency Injection

# Dependency Injection (DI)

- É uma técnica (avançada) de programação em que as dependências (ex: Bases de Dados) são disponibilizadas por parâmetro para objetos ou funções.
  - Nos controllers do ASP.NET Core, são passados por parâmetro no construtor.
- As dependências são registadas num “container”, geralmente no arranque.
  - No ASP.NET Core, isto é feito no “ConfigureServices” da classe Startup.
- Torna os programas mais modulares e fáceis de testar, porque posso substituir as dependências sem modificar as classes dependentes.
  - Especialmente útil em testes, quando quero usar uma BD “falsa” com dados dummy.

# DI for 5 year olds (<https://stackoverflow.com/a/1638961>)

*When you go and get things out of the refrigerator for yourself, you can cause problems. You might leave the door open, you might get something Mommy or Daddy doesn't want you to have. You might even be looking for something we don't even have or which has expired.*

*What you should be doing is stating a need, "I need something to drink with lunch," and then we will make sure you have something when you sit down to eat.*

*Quando vais buscar coisas ao frigorífico, podes causar problemas. Podes deixar a porta aberta, podes ir buscar alguma coisa que o Pai ou a Mãe não quer que tu comas/bebas. Tu podes até estar à procura de algo que não existe ou que já não está no prazo de validade.*

*O que deves fazer é dizer o que queres: "Eu preciso de alguma coisa para beber ao almoço", e depois nós (Pai/Mãe) arranjam-te alguma coisa adequada para ti quando fores almoçar.*

# Dependency Injection (DI)

- <https://medium.freecodecamp.org/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f>
- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.2>