Microsoft .NET Core 2.2

Utilizando .NET Core 2.2 para aplicação RESTful
-William R. N.

Observações

- Microsoft .NET Framework 5.0 foi renomeado para .NET Core 1.0 e é free e open-source
- A última versão do Framework .NET oficialmente é a 4.7.2
- Foram utilizados os pacotes NuGet NHibernate para conexão com o Banco de Dados, e conversão de dados para Objetos .NET.
- O banco de dados é montado em PostgreSQL.
- Os testes de POST e GET foram realizados utilizando a ferramenta Fiddler 4.

Diferenças entre .NET Framework e .NET Core 2.2:

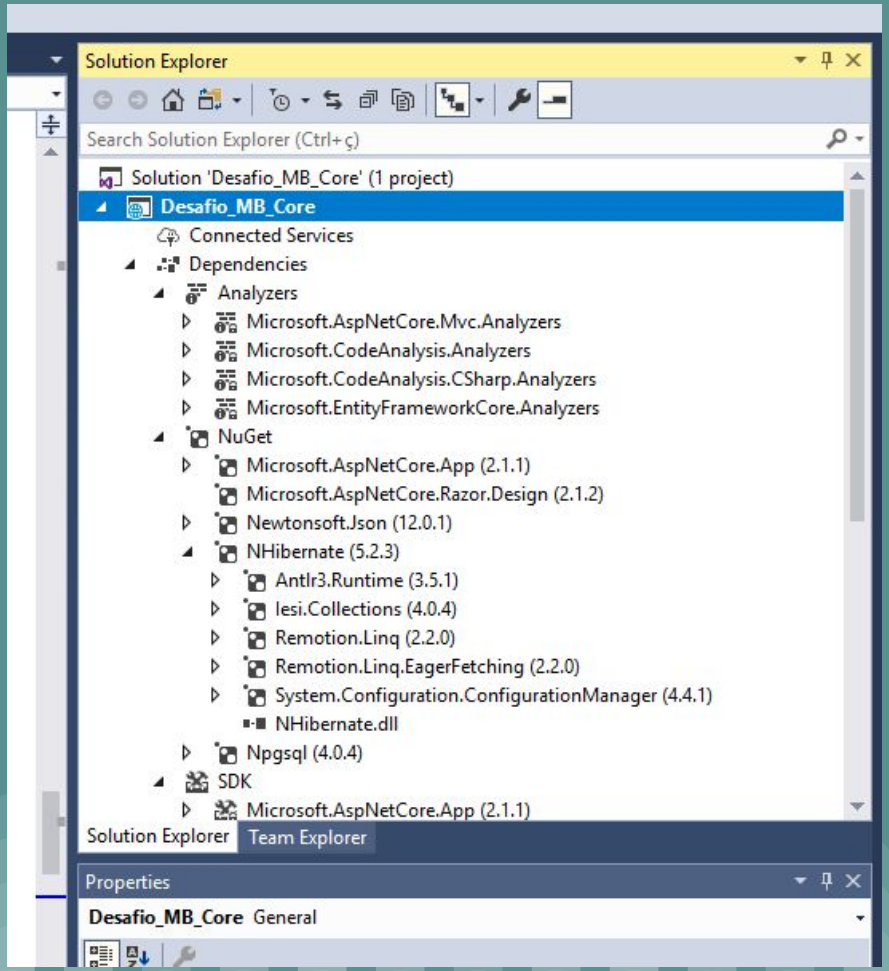
1. Aplicações para Windows.
2. Windows Presentation Foundation (WPF) e Universal Windows Platform (UWP) para aplicações Windows.
3. Pacote fechado da Microsoft
4. Grande biblioteca de código, requer várias dependências.
5. Poucas alterações no Framework a cada update.
6. Extensão de código via DLLs.

1. Aplicações para Windows, Linux e Mac.
2. Universal Windows Platform (UWP) para aplicações Windows.
3. Free e Open-source, colaboração por uma equipe de desenvolvedores.
4. Biblioteca mais ‘enxuta’ de código, mais leve e otimizado.
5. Código constantemente em desenvolvimento, alterações frequentes.
6. Extensão de código via pacotes *NuGet*.

Semelhanças entre .NET Framework e .NET Core 2.2:

1. Aplicações web via model–view–controller (MVC).
2. Como ambos são similares é possível fazer uma migração de um projeto .NET Framework para .NET Core, basta verificar se alguma API não é suportada. (Existem ferramentas como *.NET Portability Analyzer* e *.NET API analyzer* para ajudar na transição).
3. Linguagem de programação idêntica.
4. Grande parte das bibliotecas utilizadas no .NET Framework já existem como pacotes NuGet em .NET Core (Como o NHibernate, por exemplo).

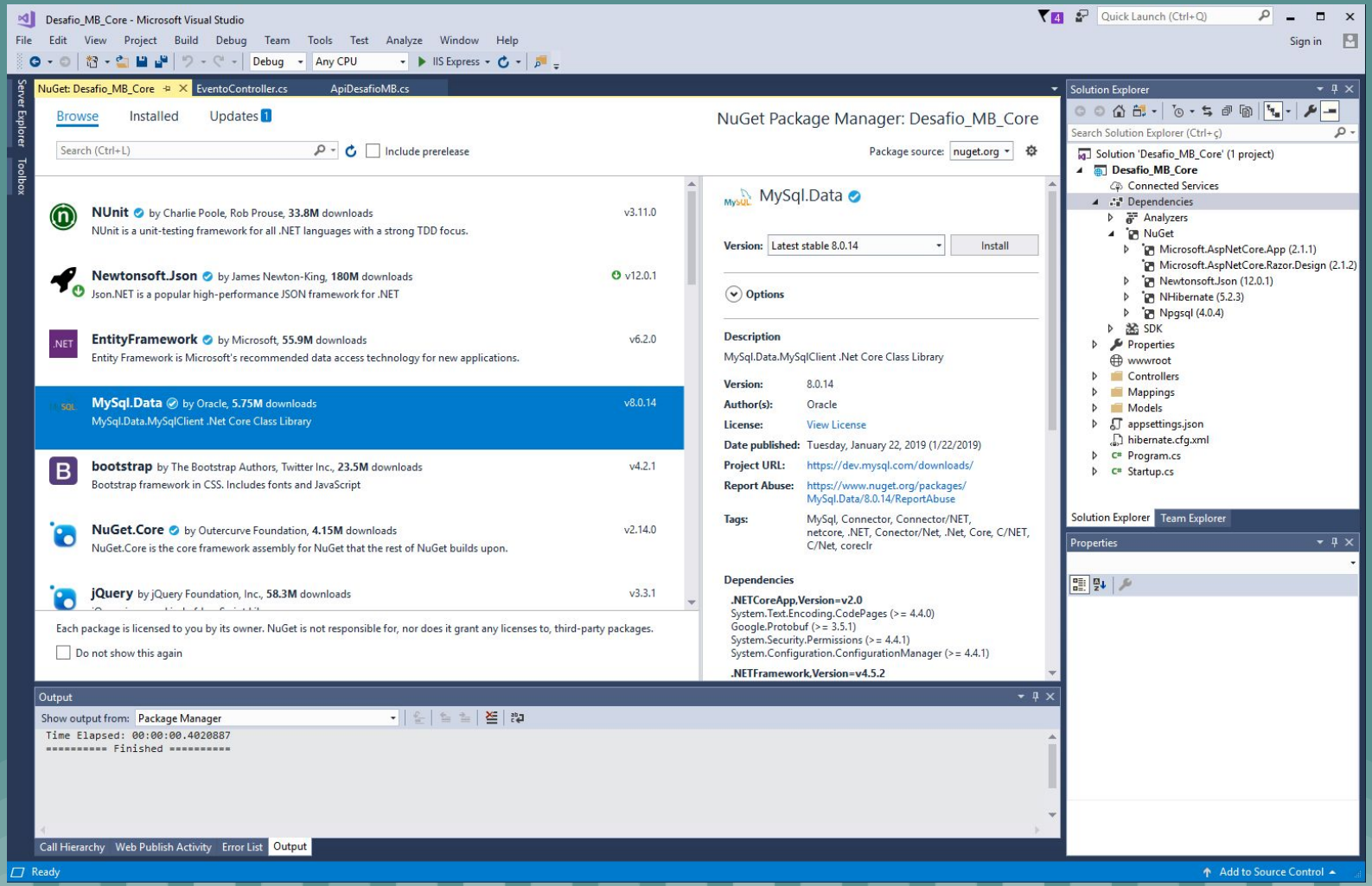
Universo de aplicações Windows para Universal Windows Platform (UWP)



Dependências NuGet no projeto

Encapsulamento de DLL e suas respectivas dependências.

Qualquer dependência adicional é avisada pelo instalador.


Desafio_MB_Core - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
Any CPU
IIS Express
Quick Launch (Ctrl+Q)
Sign in
Server Explorer
Toolbox
NuGet: Desafio_MB_Core
EventoController.cs
ApiDesafioMB.cs
Browse
Installed
Updates 1
Search (Ctrl+L)
Include prerelease
NuGet Package Manager: Desafio_MB_Core
Package source: nuget.org
NUnit by Charlie Poole, Rob Prouse, 33.8M downloads
v3.11.0
NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.
Newtonsoft.Json by James Newton-King, 180M downloads
v12.0.1
Json.NET is a popular high-performance JSON framework for .NET
EntityFramework by Microsoft, 55.9M downloads
v6.2.0
Entity Framework is Microsoft's recommended data access technology for new applications.
MySql.Data by Oracle, 5.75M downloads
v8.0.14
MySql.Data.MySqlClient .Net Core Class Library
bootstrap by The Bootstrap Authors, Twitter Inc., 23.5M downloads
v4.2.1
Bootstrap framework in CSS. Includes fonts and JavaScript
NuGet.Core by Outercurve Foundation, 4.15M downloads
v2.14.0
NuGet.Core is the core framework assembly for NuGet that the rest of NuGet builds upon.
jQuery by jQuery Foundation, Inc., 58.3M downloads
v3.3.1
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Do not show this again
MySql.Data
Version: Latest stable 8.0.14
Install
Options
Description
MySql.Data.MySqlClient .Net Core Class Library
Version: 8.0.14
Author(s): Oracle
License: View License
Date published: Tuesday, January 22, 2019 (1/22/2019)
Project URL: https://dev.mysql.com/downloads/
Report Abuse: https://www.nuget.org/packages/MySql.Data/8.0.14/ReportAbuse
Tags: MySql, Connector, Connector/NET, netcore, .NET, Conector/Net, .Net, Core, C/NET, C/Net, coreclr
Dependencies
.NETCoreApp,Version=v2.0
System.Text.Encoding.CodePages (>= 4.4.0)
Google.Protobuf (>= 3.5.1)
System.Security.Permissions (>= 4.4.1)
System.Configuration.ConfigurationManager (>= 4.4.1)
.NETFramework,Version=v4.5.2
Solution Explorer
Search Solution Explorer (Ctrl+ç)
Solution 'Desafio_MB_Core' (1 project)
Desafio_MB_Core
Connected Services
Dependencies
Analyzers
NuGet
Microsoft.AspNetCore.App (2.1.1)
Microsoft.AspNetCore.Razor.Design (2.1.2)
Newtonsoft.Json (12.0.1)
NHibernate (5.2.3)
Npgsql (4.0.4)
SDK
Properties
wwwroot
Controllers
Mappings
Models
appsettings.json
hibernate.cfg.xml
Program.cs
Startup.cs
Solution Explorer
Team Explorer
Properties
Output
Show output from: Package Manager
Time Elapsed: 00:00:00.4020887
========== Finished ==========
Call Hierarchy
Web Publish Activity
Error List
Output
Ready
Add to Source Control

Ubuntu 18.04

Register Microsoft key and feed

Before installing .NET, you'll need to register the Microsoft key, register the product repository, and install required dependencies. This only needs to be done once per machine.

Open a command prompt and run the following commands:

```
~$ wget -q https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb
~$ sudo dpkg -i packages-microsoft-prod.deb
```

Install the .NET Runtime

Update the products available for installation, then install the .NET Runtime.

In your command prompt, run the following commands:

```
~$ sudo add-apt-repository universe
~$ sudo apt-get install apt-transport-https
~$ sudo apt-get update
~$ sudo apt-get install aspnetcore-runtime-2.2
```

Veja
https://dotnet.microsoft.com/download/archives

Publish and copy over the app

Configure the app for a framework-dependent deployment.

Run dotnet publish from the development environment to package an app into a directory (for example, *bin/Release/<target_framework_moniker>/publish*) that can run on the server:

```
dotnet publish --configuration Release
```

The app can also be published as a self-contained deployment if you prefer not to maintain the .NET Core runtime on the server.

Copy the ASP.NET Core app to the server using a tool that integrates into the organization's workflow (for example, SCP, SFTP). It's common to locate web apps under the *var* directory (for example, *var/www/helloapp*).

Note

Under a production deployment scenario, a continuous integration workflow does the work of publishing the app and copying the assets to the server.

Test the app:

1. From the command line, run the app: `dotnet <app_assembly>.dll`.
2. In a browser, navigate to `http://<serveraddress>:<port>` to verify the app works on Linux locally.

Veja
https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-2.2

Perguntas

Fontes:

C# Corner
https://www.c-sharpcorner.com/article/difference-between-net-framework-and-net-core/

.NET Blog
https://blogs.msdn.microsoft.com/dotnet/2014/12/04/introducing-net-core/

Microsoft.com
https://docs.microsoft.com/en-us/dotnet/core/porting/index

StackOverflow
https://stackoverflow.com/questions/29820947/whats-the-difference-between-asp-net-5-net-core-and-asp-net-core-5