



# Microsoft Ignite The Tour

Learn. Explore. Connect.

Sao Paulo, Brazil





# Migrate your existing ASP.NET application to ASP.NET Core



Eduardo Pires

Microsoft Regional Director – MVP

[www.eduardopires.net.br](http://www.eduardopires.net.br)



**desenvolvedor.io**



# 1 – Você realmente precisa migrar sua aplicação?

Pontos a observar:

- Antecipação de necessidades futuras não suportadas no ASP.NET Clássico
- Estabilidade da aplicação, atende ao negócio e não exige uma migração
- Atender as expectativas e desejos dos desenvolvedores em trabalhar em uma tecnologia mais nova.
- Não precisa migrar? Não migre! O ASP.NET MVC 5 terá suporte durante um bom tempo ainda.

## 2 – Migre para ASP.NET Core se...

- Performance é sua maior prioridade
- Precisa de suporte para trabalhar em outras plataformas (Linux/Mac) ou rodar a aplicação em containers Linux.
- Precisa rodar diversas aplicações trabalhando com frameworks diferentes em um mesmo servidor.
- Se o suporte open-source e updates a cada 6 meses é coisa aceitável.
- Se sua aplicação é MVC, WebAPI ou SignalR

# 3 – Não migre para ASP.NET Core se...

- Por que existem tem projetos usando e isso causa uma certa pressão.
- Você considera que containers é o futuro!
- Você precisa usar o recurso de DI (dependency injection)
- Você não gosta de web.config

## 4 – Se você está utilizando ASP.NET WebForms

É hora de reescrever sua aplicação!


# Atenção!




Isto não é uma tarefa rápida, pode demorar muito tempo até uma migração completa

# 5 – .NET Portability Analyzer



Utilize o .NET Portability Analyzer para descobrir o que precisa de migração e qual serão suas incompatibilidades de código

 Visual Studio | Marketplace

Visual Studio > Tools > .NET Portability Analyzer



## .NET Portability Analyzer

Microsoft |  130,926 installs |  (15) | Free

Evaluates portability of assemblies across .NET platforms

[Download](#)

[Overview](#) [Q & A](#) [Rating & Review](#)



### How portable is your application?

The .NET Portability Analyzer helps you determine how flexible your application is across .NET platforms.



# 6 – .NET Standard

Converta suas bibliotecas para o .NET Standard

 AutoMapper 4.2.1 

A convention-based object-object mapper. AutoMapper uses a fluent configuration API to define an object-object mapping strategy. AutoMapper uses a convention-based matching algorithm to match up source to destination values. Currently, AutoMapper is designed for model projection scenarios to flatten complex object models to DTOs and other simple objects, whose design is better suited for serialization, communication, messaging, or simply an anti-corruption layer between the domain and application layer.

① There is a newer version of this package available. See the version list below for details.

Package Manager

.NET CLI

Paket CLI

PM> Install-Package AutoMapper -Version 4.2.1

▼ Dependencies

.NETFramework 4.5

No dependencies.

.NETPlatform

Microsoft.Extensions.DependencyInjection.Abstractions (>= 4.0.0)

Microsoft.Extensions.DependencyInjection (>= 4.0.10)

Microsoft.Extensions.DependencyInjection.Abstractions.Concurrent (>= 4.0.10)

Microsoft.Extensions.DependencyInjection.Abstractions.Specialized (>= 4.0.0)

System.ComponentModel.DataAnnotations.Model.TypeConverter (>= 4.0.0)

System.Diagnostics.Debug (>= 4.0.10)

System.Dynamic.Runtime (>= 4.0.10)

System.Globalization (>= 4.0.10)

System.Linq (>= 4.0.10)

System.Linq.Expressions (>= 4.0.10)

System.Linq.Queryable (>= 4.0.10)

System.ObjectModel (>= 4.0.10)

System.Reflection (>= 4.0.10)

System.Reflection.Emit (>= 4.0.0)

System.Reflection.Emit.ILGeneration (>= 4.0.0)

System.Reflection.Extensions (>= 4.0.0)

System.Reflection.Primitives (>= 4.0.0)

System.Reflection.TypeExtensions (>= 4.0.0)

System.Runtime (>= 4.0.20)

System.Runtime.Extensions (>= 4.0.10)

System.Runtime.InteropServices (>= 4.0.10)

System.Threading

Portable Class Library (.NETFramework 4.5, Windows 8.0, WindowsPhoneApp 8.1)

No dependencies.

# 6 – .NET Standard

Converte suas bibliotecas para o  
.NET Standard



AutoMapper 8.0.0

A convention-based object-object mapper.

Package Manager

.NET CLI

Paket CLI

PM> Install-Package AutoMapper -Version 8.0.0



## Dependencies

.NETFramework 4.6.1

System.ValueTuple (>= 4.5.0)

.NETStandard 2.0

Microsoft.CSharp (>= 4.5.0)

System.Reflection.Emit (>= 4.3.0)



## Version History

Version	Downloads	Last updated
8.0.0	82,603	17 days ago
7.0.1	2,454,858	6 months ago
7.0.0-alpha-0001	29,706	8 months ago
6.2.2	3,640,849	06/12/2017
6.2.1	520,319	16/11/2017

+ Show more

# 7 – Migre para o novo formato de projeto (csproj)

```
1 <Project Sdk="Microsoft.NET.Sdk.Web">
2
3   <PropertyGroup Label="Globals">
4     <SccProjectName></SccProjectName>
5     <SccProvider></SccProvider>
6     <SccAuxPath></SccAuxPath>
7     <SccLocalPath></SccLocalPath>
8   </PropertyGroup>
9
10  <PropertyGroup>
11    <TargetFramework>netcoreapp2.1</TargetFramework>
12  </PropertyGroup>
13
14  <ItemGroup>
15    <PackageReference Include="Hangfire" Version="1.6.21" />
16    <PackageReference Include="Microsoft.AspNetCore.App" />
17    <PackageReference Include="Microsoft.AspNetCore.Razor.Design" Version="2.1.2" PrivateAssets="All" />
18  </ItemGroup>
19
20  <ItemGroup>
21    <Content Update="appsettings.json">
22      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
23    </Content>
24  </ItemGroup>
25
26 </Project>
27
```

## 8 – Substitua os namespaces

Um simples Find Replace em seu projeto:

De: `System.Web`

Para: `Microsoft.AspNetCore`

# 9 – Converta o global.asax para usar o Startup.cs

O startup.cs é o core da configuração da sua app:

- MVC / WebAPI
- Rotas
- Autenticação
- CORS
- Filtros
- Exception Handling
- Middlewares
- Dependency Injection

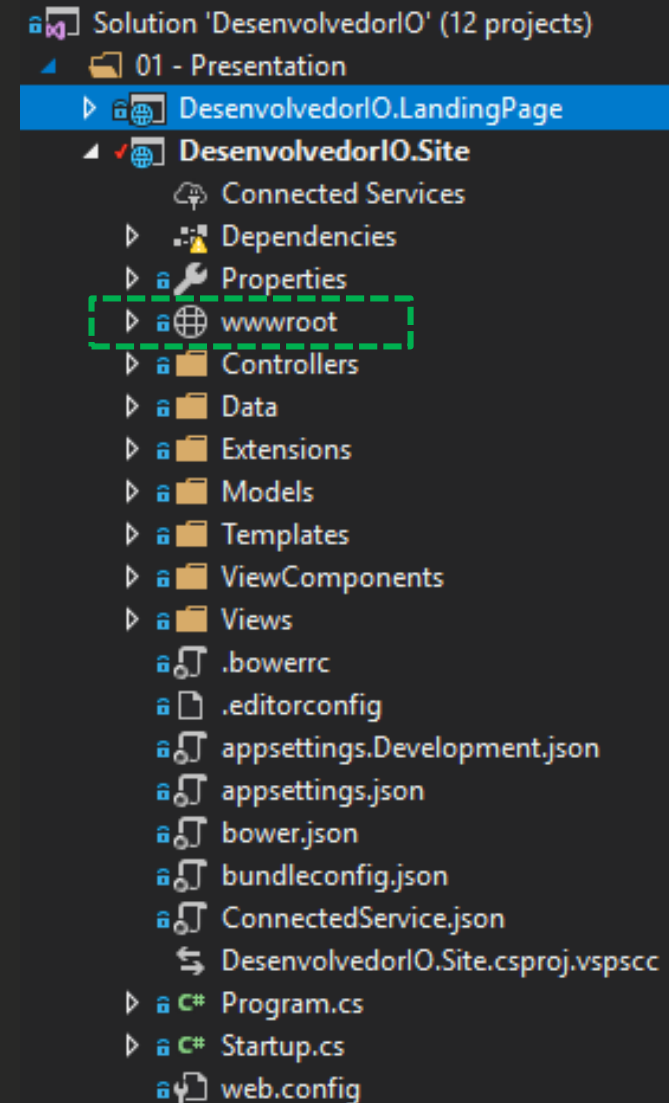


Demo Startup.cs



# 10 – Transfira o conteúdo estático

JS, CSS, Imagens e demais arquivos estáticos deverão estar dentro da pasta wwwroot



# 11 – Porte o EF6 para o EF Core

Realize um “Scaffold” uma engenharia reversa para criar seus modelos e seu contexto do EF

```
dotnet ef dbcontext scaffold “connectionstring”  
Microsoft.EntityFrameworkCore.SqlServer -o Model
```



## 12 – Remover HttpContext antigo

Remova qualquer referência ao HttpContext

Apesar do mesmo nome no ASP.NET Core é um objeto totalmente diferente e será necessário substituir os trechos que fazem uso desta referência.

# 13 – Atualize a autenticação com Identity

Se a aplicação já usa Identity é bem simples!

- Basta atender ao mesmo padrão de um novo projeto criado via template.
- Utilize o Scaffold para customizar as Views

## 14 – Bundle & Minify

Converta sua configuração para o bundleconfig.json

- Um unico arquivo de configuração
- Atualize as Views para utilizar os novos arquivos.

# 15 – Migre as Views

Copie suas Views para o novo projeto

- As Views são compatíveis com a antiga syntaxe do Razor 😊
- Utilize a `_ViewImports.cshtml` para adicionar as referências.

# Revisão:

- .NET Portability Analyzer
- .NET Standard
- Pacotes de Terceiros
- Novo csproj
- Sytem.Web > Microsoft.Asp.Net.Core
- Global.asax > Startup.cs
- Handlers, Modules > Middlewares
- Dependency Injection
- Application Configuration
- Conteúdo Estático > wwwroot
- EF > EF core
- Remover referências HttpContext
- ASP.NET Identity
- CSS JS Updates
- Migrar Views



# Muito Obrigado!

Não esqueça de avaliar esta sessão 😊

[www.eduardopires.net.br](http://www.eduardopires.net.br)

[www.desenvolvedor.io](http://www.desenvolvedor.io)

