



ASP.NET Core essentials

Andrea Saltarello

Solution Architect @ Managed Designs

<https://twitter.com/andysal74>

18 NOVEMBRE 2016



Talk.About();

- Anatomia di una applicazione ASP.NET Core
- Architettura di ASP.NET Core
- Cenni a MVC Core

P.S.: demo scaricabili qui: <http://nsk.codeplex.com> (appena mi convertono il progetto da TFSVC a Git 😊)





Anatomia di una applicazione ASP.NET Core

Una applicazione ASP.NET Core:

1. Può «girare» sia su .NET Core sia su .NET Framework
2. Limitatamente a .NET Core, può essere sviluppata su: Linux, macOS, Windows
3. È una «normale» applicazione console “ospitata” da un *server* e configurata mediante codice

ASP.NET Core at a glance

ASP.NET 4.6 and ASP.NET Core 1.0

ASP.NET 4.6	ASP.NET Core 1.0
.NET Framework 4.6 	.NET Core 1.0   
.NET framework libraries	.NET core libraries
Compilers and runtime components (.NET Compiler Platform: Roslyn, C#, VB, F# Languages, RyuJIT, SIMD)	

La Dev story: gli editor

Per sviluppare, abbiamo varie opzioni:

	Linux	macOS	Windows
CLI + OmniSharp ¹	solo .NET Core	solo .NET Core	solo .NET Core
Xamarin Studio		solo .NET Core ²	
Visual Studio Code	solo .NET Core	solo .NET Core	X
Visual Studio ³		solo .NET Core	X

1. Supporto per: Sublime, Atom, Emacs, Vim e Brackets
2. Solo supporto xproj
3. macOS: solo csproj; Windows: xproj per v2015, csproj per v2017

La Dev story: il deploy

(ASP).NET Core è distribuito in due versioni:

1. LTS: supportate per 3 anni *o* per 1 anno dopo il rilascio della LTS successiva.
L'attuale versione LTS è la 1.0.1
2. Current: supportate, all'interno della corrispettiva LTS, per 3 mesi a partire dal rilascio della «current» successiva

.NET Core only:

- il deploy permette sia di usare l'installazione del FX a livello di macchina, sia una locale alla applicazione
- non esistono drop release, tutte le versioni sono installate side-by-side

[<https://www.microsoft.com/net/core/support>]

1 - demo

Dev Story

Una «normale» applicazione Console...

0 references | Andrea Saltarello, 7 days ago | 1 author, 1 change

```
| public class Program
```

```
{
```

0 references | Andrea Saltarello, 7 days ago | 1 author, 1 change

```
| public static void Main(string[] args)
```

```
{
```

```
    var host = new WebHostBuilder()
```

```
        .UseKestrel()
```

```
        .UseContentRoot(Directory.GetCurrentDirectory())
```

```
        .UseIISIntegration()
```

```
        .UseStartup<Startup>()
```

```
        .Build();
```

```
    host.Run();
```

```
}
```

```
}
```


...ospitata da un server...

Sono i «container»: ASP.NET Core fornisce i seguenti:

- Kestrel (managed, xplat)
- WebListener (httpsys, 1.1+)

```
0 references | Andrea Saltarello, 7 days ago | 1 author, 1 change
| public class Program
| {
|     0 references | Andrea Saltarello, 7 days ago | 1 author, 1 change
|     public static void Main(string[] args)
|     {
|         var host = new WebHostBuilder()
|             .UseKestrel()
|             .UseContentRoot(Directory.GetCurrentDirectory())
|             .UseIISIntegration()
|             .UseStartup<Startup>()
|             .Build();
|
|         host.Run();
|     }
| }
```

...e configurata mediante codice

Per configurare una applicazione usiamo:

- **project.json** per specificare le reference, da VS 2017/Mac GA in poi *ye olde* csproj
- **Startup.cs** (o omologa) per configurare i *middleware*
 1. ConfigureServices()
 2. Configure()
- [OPT] **appsettings.json** per «informazioni» di configurazione
- ASPNETCORE_ENVIRONMENT

Middleware

Sono **funzioni** che:

- implementano la delegate **RequestDelegate**
- vengono aggiunti alla request pipeline di ASP.NET Core

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Hello World!");
    });
}
```

2 - demo

Middleware

Middleware

ASP.NET Core fornisce out of the box alcuni middleware, tra i quali:

- Identity
- MVC
- ResponseCaching (1.1+)
- ResponseCompression (1.1+)
- Rewrite (1.1+)

Per convenzione, vengono *registrati* in **ConfigureServices** ed *attivati* in **Configure**

[<https://blogs.msdn.microsoft.com/webdev/2016/11/16/announcing-asp-net-core-1-1/>]

ConfigureServices(), parte 2

La *registrazione* dei middleware serve, sostanzialmente, a configurarne le *dipendenze* (nel senso IoC) tipicamente mediante extension method ad hoc (es: **AddMvc**, **AddIdentity**, ...)

E' basata su un IoC container built-in, che permette di:

1. Registrare dei tipi indicando lo scope delle istanze da generare:
 - **AddInstance**
 - **AddSingleton**
 - **AddScoped**
 - **AddTransient**
2. Esporre le dipendenze di:
 - Ctor
 - Parametri action: **FromServicesAttribute**
 - View: **@inject**

E' possibile sostituire il container built in con uno di terze parti:

<https://github.com/aspnet/DependencyInjection/blob/dev/README.md>

MVC

ASP.NET MVC Core

E' il middleware fornito out of the box da ASP.NET Core per implementare applicazioni basate sulla variante **Model 2** del pattern MVC (quella di **Struts**, **Rails**, **Monorail**, ...)

API molto simile a MVC 5 di ASP.NET 4, ma con alcune differenze strutturali:

- pipeline MVC-WebAPI integrata
- supporto a controller POCO
- dependency ~~Resolver~~ Injection

Per vederlo in azione, guardare qui:

<http://www.ugidotnet.org/video/189329326/NET05--ASP-NET-MVC-Core-1>

Grazie!

Contatti:

- <https://twitter.com/andysal74>
- <http://blogs.ugidotnet.org/pape>
- andrea@ugidotnet.org

Thank you! Questions?

<https://twitter.com/ugidotnet>