

# Introduzione ad ASP.NET Core 1.0

Andrea Saltarello

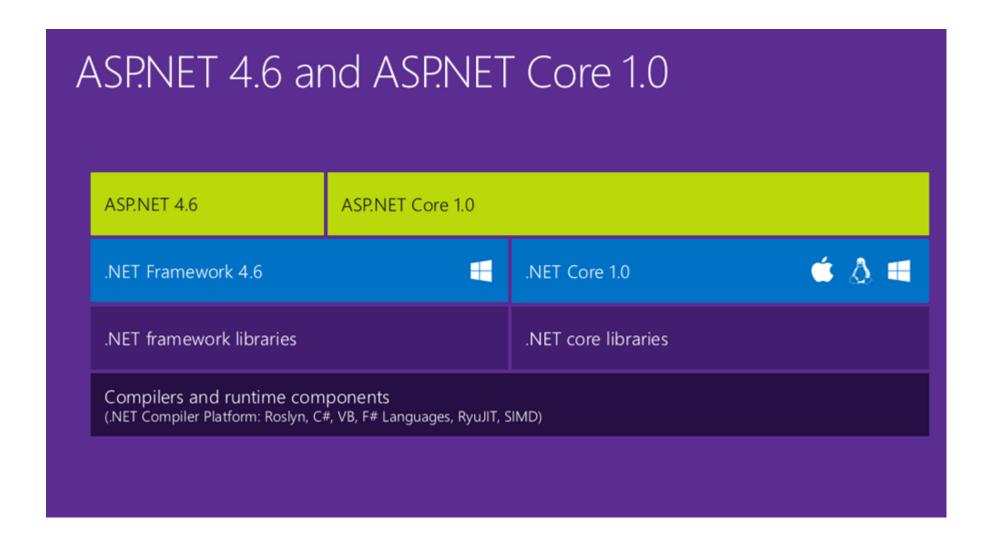


## Talk.About();

- Anatomia di una applicazione ASP.NET Core
  - Hosting
  - Configurazione
  - Building blocks
- Tooling
- MVC



#### ASP.NET Core 1.0 at a glance





### Anatomia di una applicazione ASP.NET

#### Una applicazione ASP.NET Core:

- 1. È una «normale» applicazione console
- 2. Si configura mediante codice
- 3. «gira» su (in rigoroso ordine alfabetico): Linux, OS X e Windows



```
O references | Andrea Saltarello, 7 days ago | 1 author, 1 change
public class Program
    O 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();
```



## Configurazione

Per configurare una applicazione usiamo:

- Startup.cs (o omologa)
  - 1. ConfigureServices()
  - 2. Configure()
- [OPT] appsettings.json per «informazioni» di configurazione
- (temporaneamente) project.json per specificare le reference



ASP.NET Core everywhere Routes



### **Building blocks**

ASP.NET Core fornisce alcuni building block:

- Dependency Injection
- Logging

Sono forniti come *interfacce* dotate di una implementazione, sostituibile, implementazione di default



### Dependency Injection

Supportata in Controller, Filtri, View sostanzialmente \*dappertutto\*

- 1. Registrare i tipi in **ConfigureServices**, indicando lo scope:
  - AddInstance
  - AddSingleton
  - AddScoped
  - AddTransient
- 2. Esporre le dipendenze
  - Ctor
  - Parametri action: FromServicesAttribute
  - View: @inject

E' possibile sostituire il container built in con uno di terze parti: <a href="https://github.com/aspnet/DependencyInjection/blob/dev/README.md">https://github.com/aspnet/DependencyInjection/blob/dev/README.md</a>



**Dependency Injection** 



## **Tooling**

I tool per ASP.NET Core supportano Visual Studio 2015, ma sono in release *preview 2* e sappiamo che saranno modificati sostanzialmente. Al momento gli asset sono:

- .xproj omologo del .csproj
- project.json per info progetto e asset server side
- bower.json per asset client side



Tooling



#### Controller

Una singola, ed opzionale, classe base:

Microsoft.AspNet.Mvc.Controller

Sono Action tutti i metodi pubblici di un controller.

Se restituiscono:

- IActionResult, il "giro" è quello di MVC
- != IActionResult, il "giro" è quello di WebAPI (quindi content negotiation o ProducesAttribute)

Se costruiamo degli helper, decoriamoli con NonActionAttribute



MVC

WebAPI

**JSON** Serialization



#### POCO Controller

Nessuna classe base; può essere conveniente dichiarare ed attivare:

- public ActionContext ActionContext { get; set; } \*
- public ViewDataDictionary ViewData { get; set; } \*
- public IUrlHelper Url { get; set; }

- \* Solo con container di terze parti:
- https://github.com/aspnet/Mvc/issues/2151#issuecomment-104541719
- https://github.com/aspnet/Announcements/issues/28



#### View 1-2-3

- Ye ol' good **Razor** view engine ©
  - \_ViewImports.cshtml
- WebFormsViewEngine non è più supportato
- TagHelpers
  - Package Microsoft.AspNet.Mvc.TagHelpers (o cmq contenente classi derivate da TagHelper)
  - @addTagHelper AssemblyName (es: "Microsoft.AspNet.Mvc.TagHelpers")
  - @removeTagHelper
  - @tagHelperPrefix
  - |



\_ViewImports.cshtml

TagHelper

Environment



#### Grazie!

#### Contatti:

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



# Thank you! Questions?

https://twitter.com/ugidotnet

