MS going FOSS: .NET Core on Linux

Mauro Leone

HCSSLUG
www.hcsslug.org

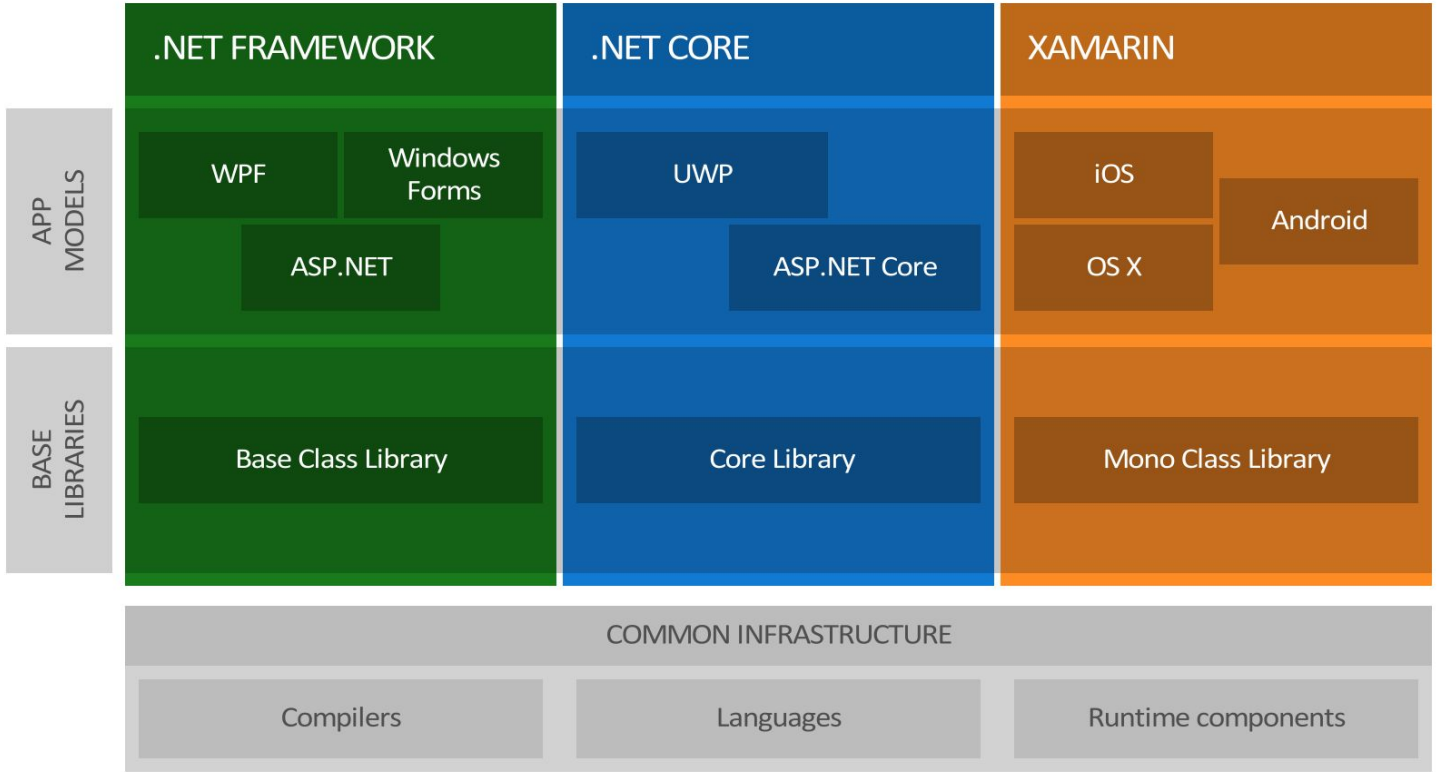
Il mondo .NET fino a poco fa!

- 15 anni di onorata attività

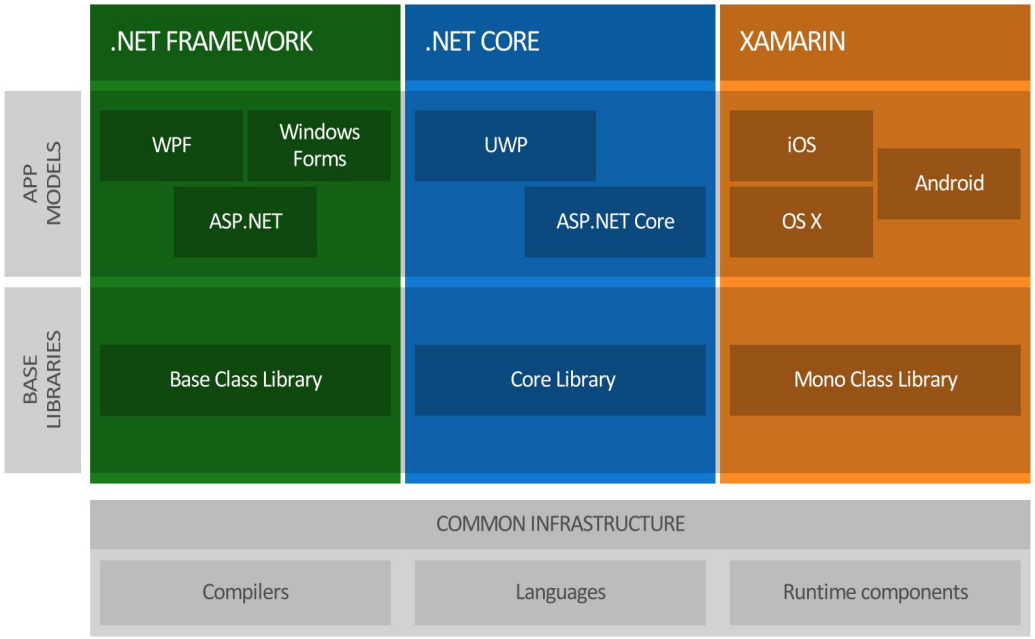
- Codebase diverse, compilazioni diverse
  - Desktop
  - Silverlight (????)
  - ARM con subset vari:
    - Compact Framework
    - Micro Framework

- Riproduzioni del framework .NET:
  - **Mono**
    - Compatibile con Linux, iOS, MacOS ecc…
    - Disponibile anche per processori ARM
  - **Xamarin**
    - .NET per Android, iOS

.NET e la sua architettura



.NET e la sua architettura

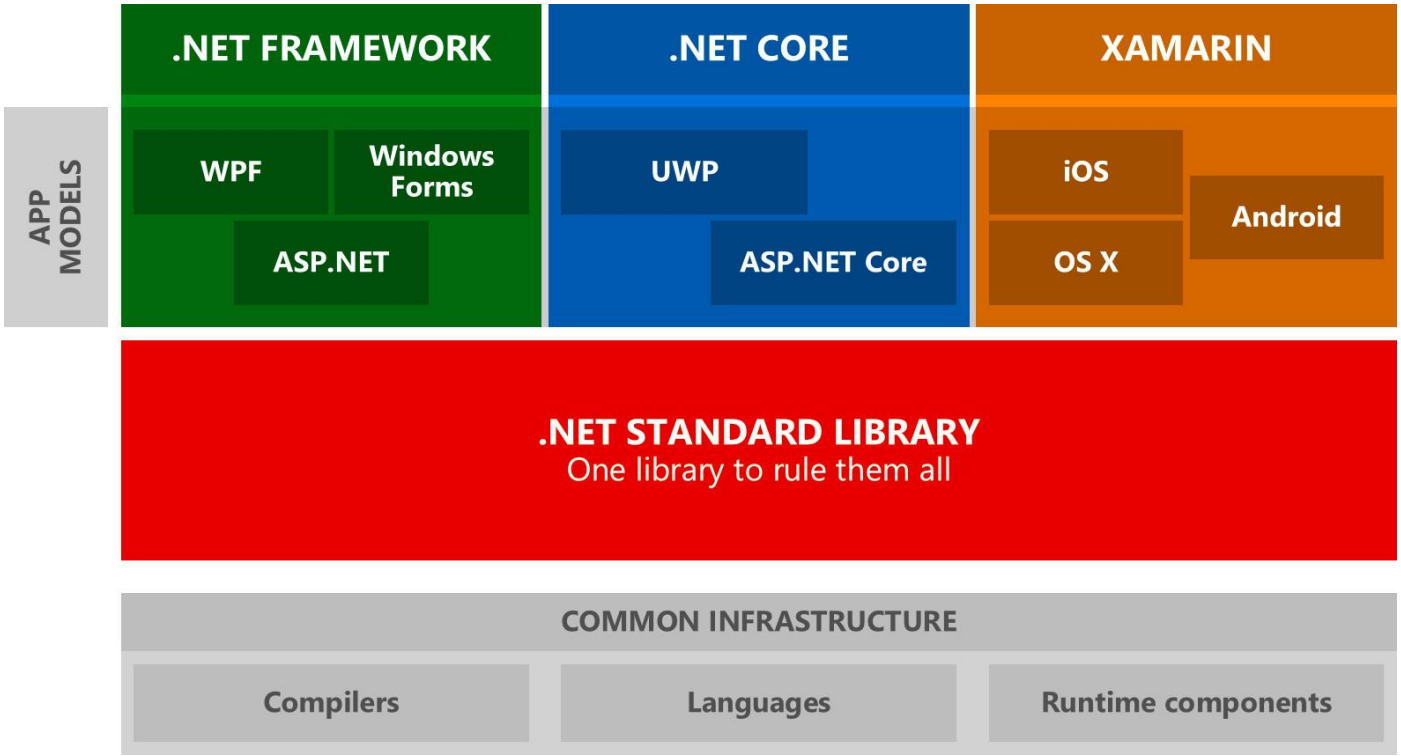


} Qui risiede la vostra applicazione: modello, data ecc…. quello che scrivete, insomma!

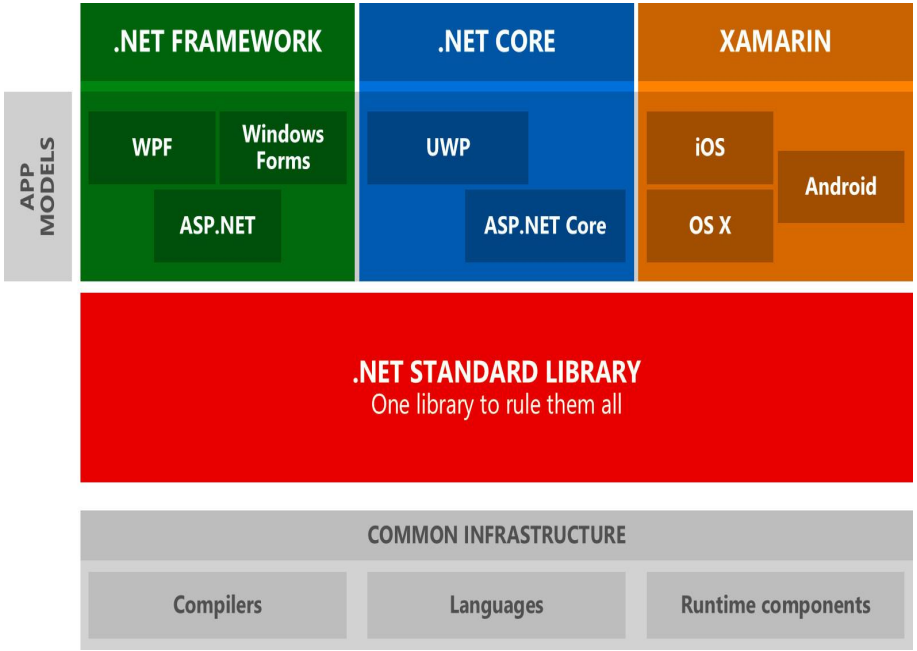
} Qui ci sono le librerie base (es. System.IO, System.Net ecc...). Notiamo come ci siano implementazioni UGUALI ma di proprietà diversa.

} Questo è il runtime (CLR)

.NET oggi: l'architettura



.NET oggi: dotnetcore



} Questa parte qui è rimasta invariata. E' sempre il vostro codice!

} Unica implementazione standard: minimale, regole ben precise.

} MS mette a disposizione un nuovo CLR compatibili con varie piattaforme

.NET oggi: dotnetcore

Disponibili per tutto

- CoreCLR è il nuovo CLR:
  - Nuovo JIT
  - Interop
  - GC
  - Servizi di base (Thread, syscall cec…)

- CoreFX è il nuovo blocco della BCL:
  - CoreFX sta per .NET Core Foundation Libraries.
  - Compatibile con tutti i CLR disponibili in giro

Specifico Microsoft Windows

Su windows le UWP possono usare un CLR chiamato CoreRT che fa uso di **.NET Native** che genera ***codice nativo, senza dipendenze.***

CoreRT e .NET Native è disponibile anche per altre piattaforme.

.NET Core

Open Source

CoreCLR
https://github.com/dotnet/coreclr

CoreRT
https://github.com/dotnet/corert

CoreFX
https://github.com/dotnet/corefx

...e molto altro ancora!

Trovate tutto qui
https://github.com/dotnet

Perchè .NET Core e non .NET?

- .NET attuale è progettato su vecchi standard.
- Troppe dipendenze e targeting!
- Server-side
- Incoraggiare l’uso di sistemi non proprietari (es. Windows) fornendo una solida base di sviluppo.
- Sviluppo moderno: l'architettura attuale di .NET Core è pensata per:
  - Scalabilità
  - Load Balancing
  - Isolamento
    - Containers!

Introduzione a .NET Core

- **linguaggio principe: C# (si pronuncia “C Sharp” e non “C hashtag”)**
  - Sintassi Java-like (ma più coerente)
  - Fortemente tipizzato
  - BCL ampia e parco librerie enorme
  - async/await pattern
  - yield
  - GC
  - Supporto ai Generics fantastico

Introduzione a .NET Core

CLI

Command Line Interface che fa da padrone su tutto lo sviluppo .NET.

Gli IDE, gli editor e qualsiasi altra cosa fa uso di questo tool.

Nota: su .NET Core 2.x sono disponibili nuovi comandi per la CLI.

Nuget e BCL

Nuget è il gestore delle dipendenze principe.

La CLI recupera i pacchetti da Nuget.

Una dll per ciascun namespace:

- o sono specificate come referenza e poi prelevate da Nuget.
- o sono parte della solution stessa.

Introduzione a .NET Core

- Installazione: https://www.microsoft.com/net/core
  - Windows, Linux, Mac, Docker

- Da console:
  - Creare un nuovo progetto (crea una nuova cartella con il nome del progetto)
    - `dotnet new console -o myapp`
  - Aggiornare/scaricare le dipendenze da nuget
    - `dotnet restore`
  - Compilare e avviare l'applicazione
    - `dotnet run`

Per gli ultimi due comandi è necessario trovarsi nella cartella del progetto quindi prima di darli spostiamoci in "myapp" con `cd myapp`*.*

.NET Core - Esempio stupido

```
using System.IO;

namespace myapp
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            // Enumero tutti i file nella cartella attuale
            foreach(var f in System.IO.Directory.EnumerateFiles("./"))
                Console.WriteLine(f);
            Console.WriteLine("Fine.");
        }
    }
}
```

Codice banale:

“Cerca tutti i file nel percorso attuale e scrivili in output.”

Prima di terminare, scrivi “Fine.”

Dove gira?
Su tutto, persino mobile!

Introduzione a .NET Core

Progetti che possono essere costruiti tramite CLI con new:

- **console**
- **classlib**
- **web**
- **webapi**

Ce ne sono anche altri di più alto livello come **angular** e **razor**.

Fonte:
https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-new?tabs=netcore2x

Introduzione a .NET Core

ASP.NET Core

- Framework cross-platform
- Performance!
- Open Source

- Non è un web server!
- … ma ne usa uno!

Fonte:
https://docs.microsoft.com/en-us/aspnet/core/fundamentals/index?tabs=aspnetcore2x

Introduzione a .NET Core

ASP.NET Core e Kestrel

- Web server cross platform
- Basato su libuv (https://github.com/libuv/libuv)
  - async I/O
- Usato in coppia con un reverse proxy per prestazioni migliori
  - ...ma funziona anche senza ;)

Fonte:
https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?tabs=aspnetcore2x

Introduzione a .NET Core

```
dotnet new web -o mywebapp
```

.NET Core - Esempio web

```
namespace myweb
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            BuildWebHost(args).Run();
        }

        1 reference
        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .Build();
    }
}
```

Startup.cs ● Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;

namespace myweb
{
    1 reference
    public class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/
        0 references
        public void ConfigureServices(IServiceCollection services)
        {
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.Run(async (context) =>
            {
                await context.Response.WriteAsync("Hello World!");
            });
        }
    }
}
```

Domande? :)

Grazie dell’attenzione!