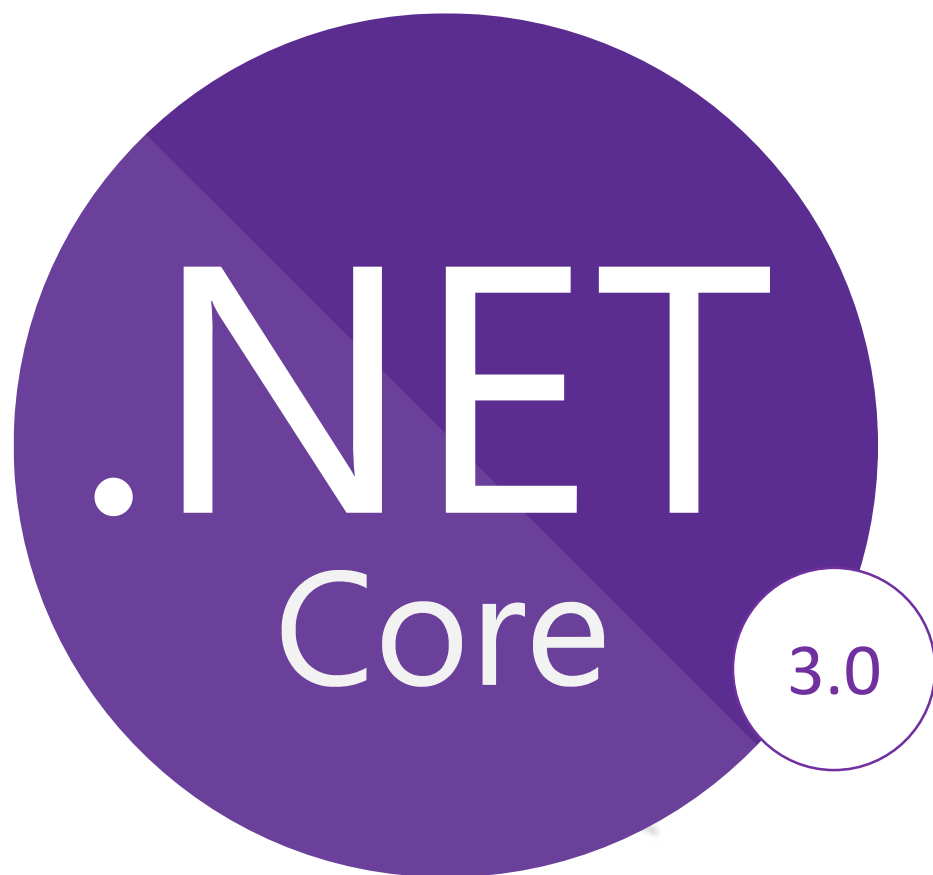


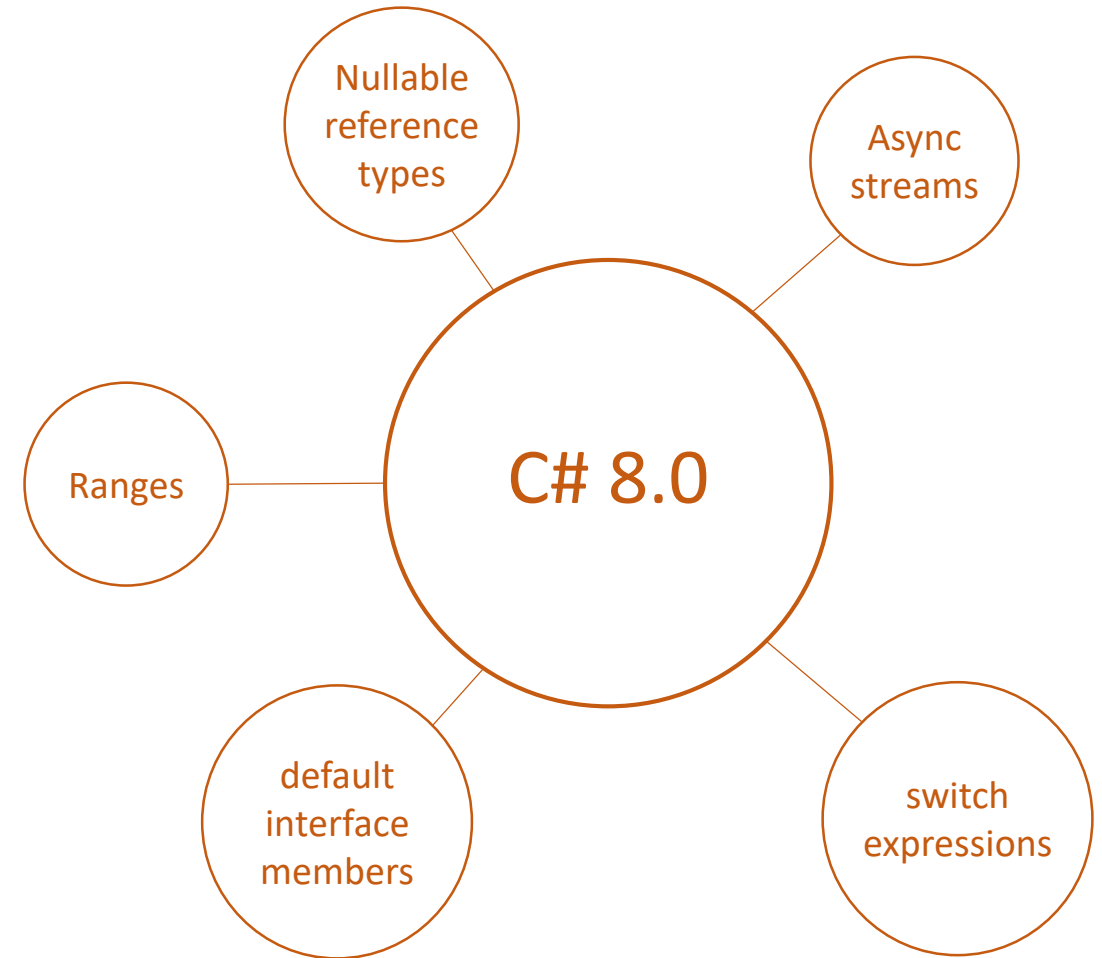
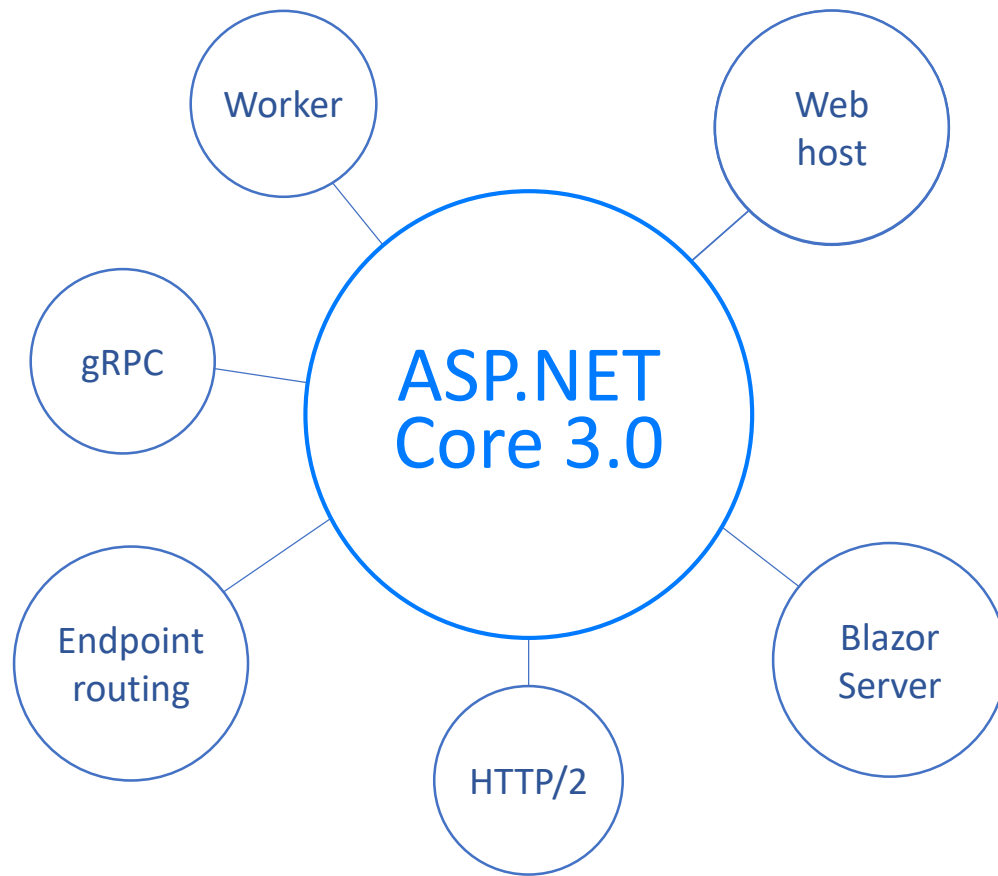
Sezione 15

Novità di .NET Core 3.0

Le novità di .NET Core 3.0



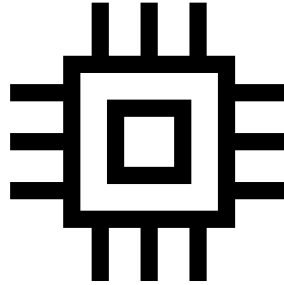
Le novità di .NET Core 3.0



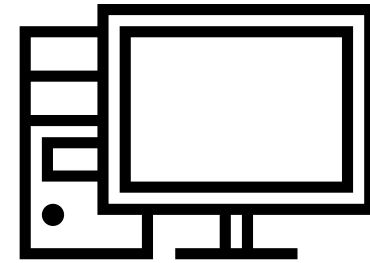
Obiettivo: aumentare la diffusione di .NET Core



Web



IoT



Desktop

Nuovi template

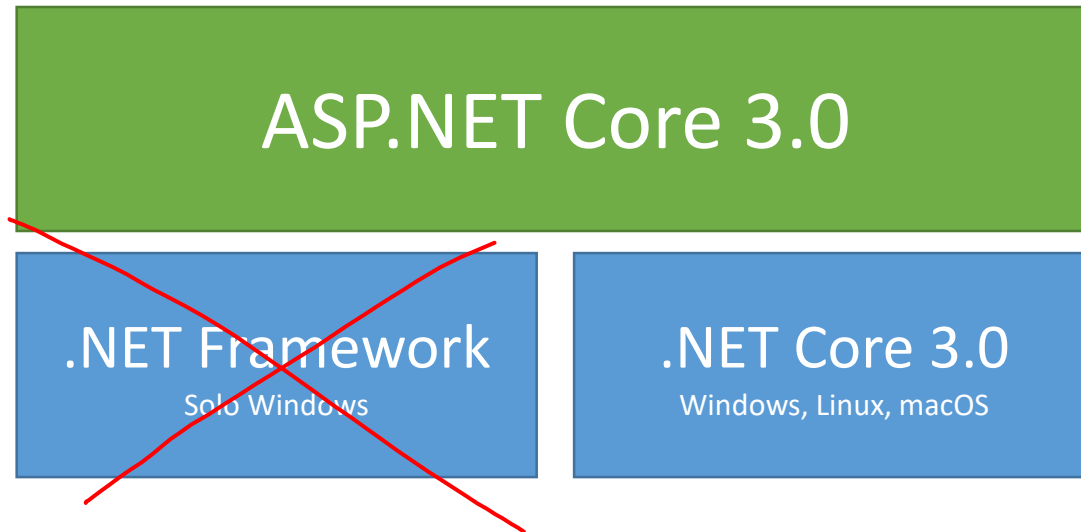
dotnet new

Templates	Short Name	Language
-----	-----	-----
WPF Application	wpf	[C#]
WPF Class library	wplib	[C#]
WPF Custom Control Library	wpfcustomcontrollib	[C#]
WPF User Control Library	wpfusercontrollib	[C#]
Windows Forms (WinForms) Application	winforms	[C#]
Windows Forms (WinForms) Class library	winformslib	[C#]
Worker Service	worker	[C#]
ASP.NET Core gRPC Service	grpc	[C#]
Blazor Server App	blazorserver	[C#]

Migrare da ASP.NET Core 2.2 a 3.0

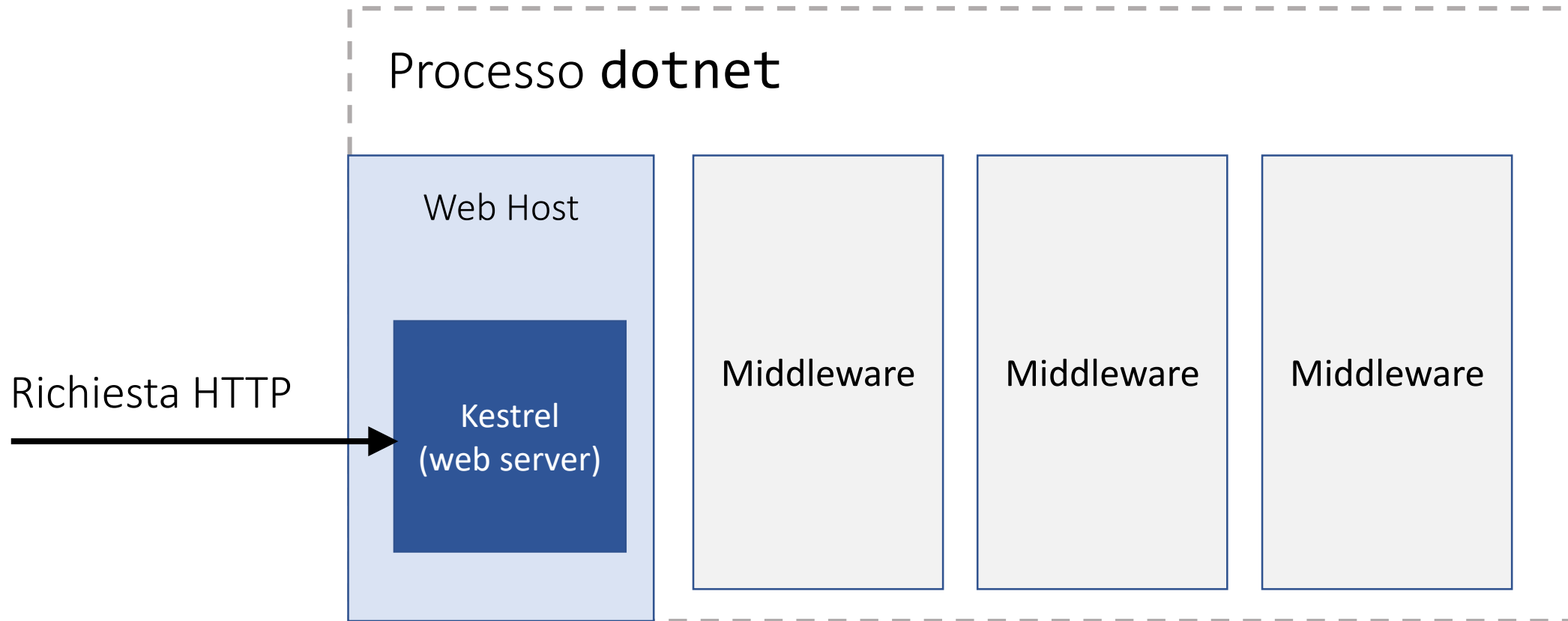
- Scarichiamo .NET Core SDK 3.0;
 - Se abbiamo un file `global.json`, aggiorniamo la versione della SDK;
 - Nel file `.csproj`, rimuovere i riferimenti ai pacchetti di ASP.NET Core;
 - Nel file `Program.cs`, usare il `generic host`;
 - Nel file `Startup.cs` usare l'endpoint routing;
 - Modifichiamo le interfacce obsolete come `IHostingEnvironment`;
 - Aggiorniamo il percorso all'assembly nel file `.vscode/launch.json`.
-
- Le istruzioni dettagliate sono su:
<https://docs.microsoft.com/it-it/aspnet/core/migration/22-to-30>

ASP.NET Core 3.0 richiede .NET Core 3.0

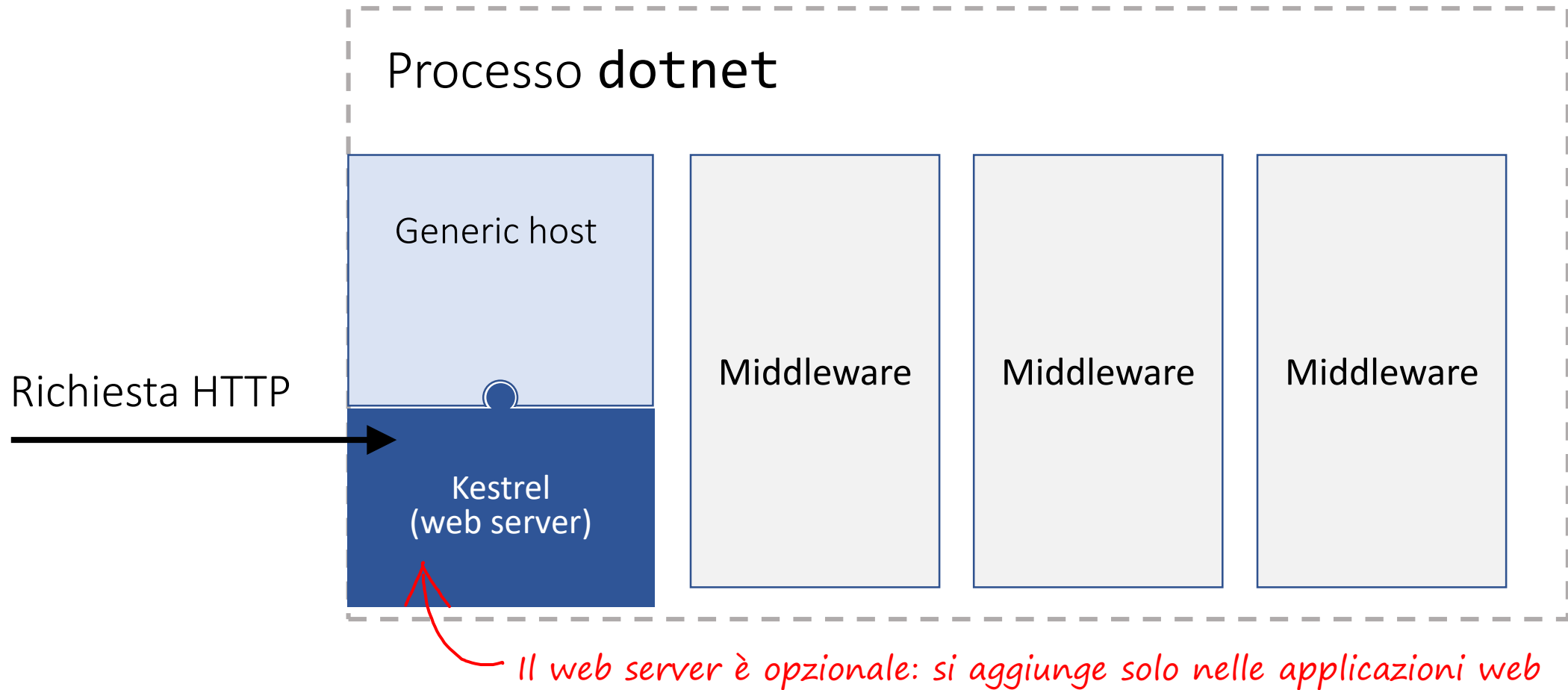


Per usare .NET Framework si dovrà restare su ASP.NET Core 2.1.

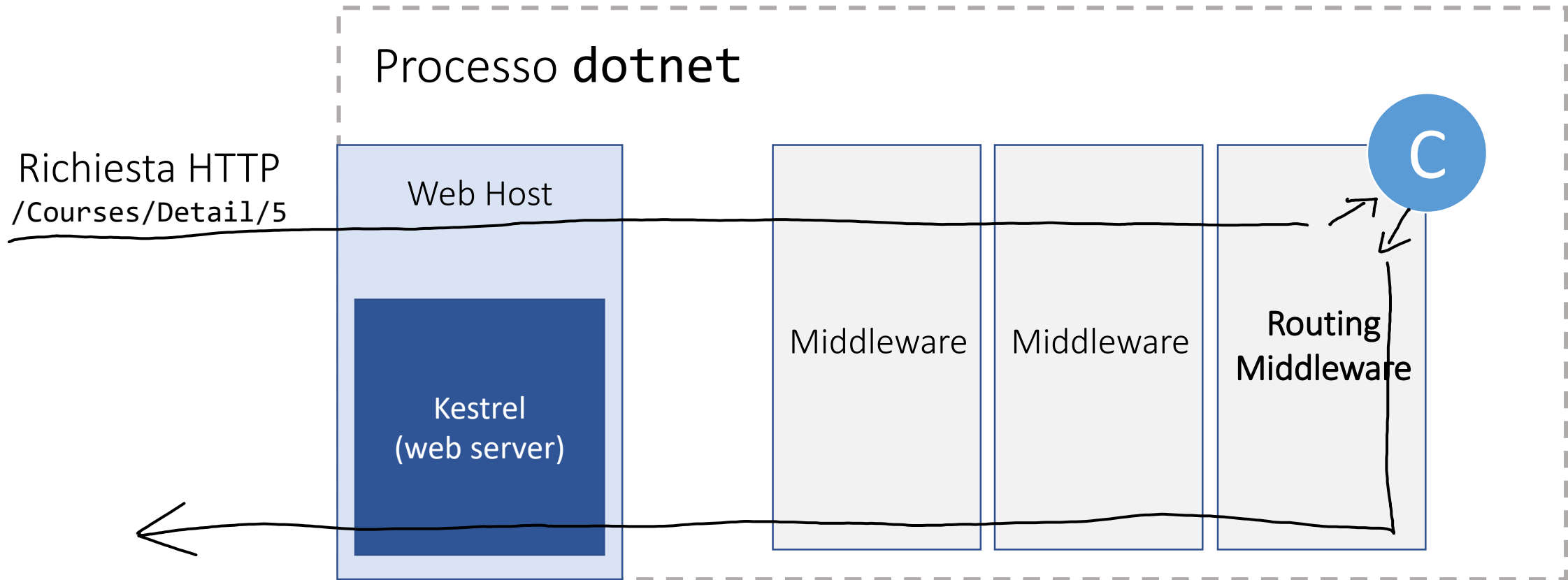
Web Host (tradizionale)



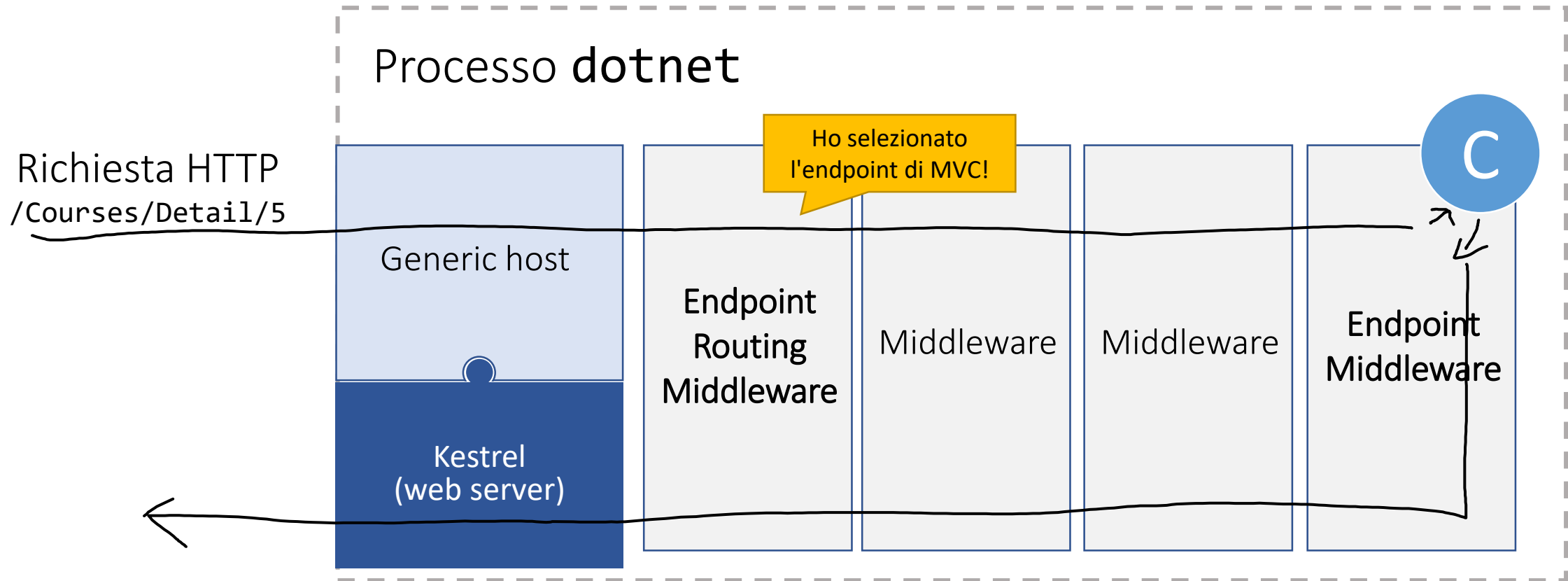
Generic Host



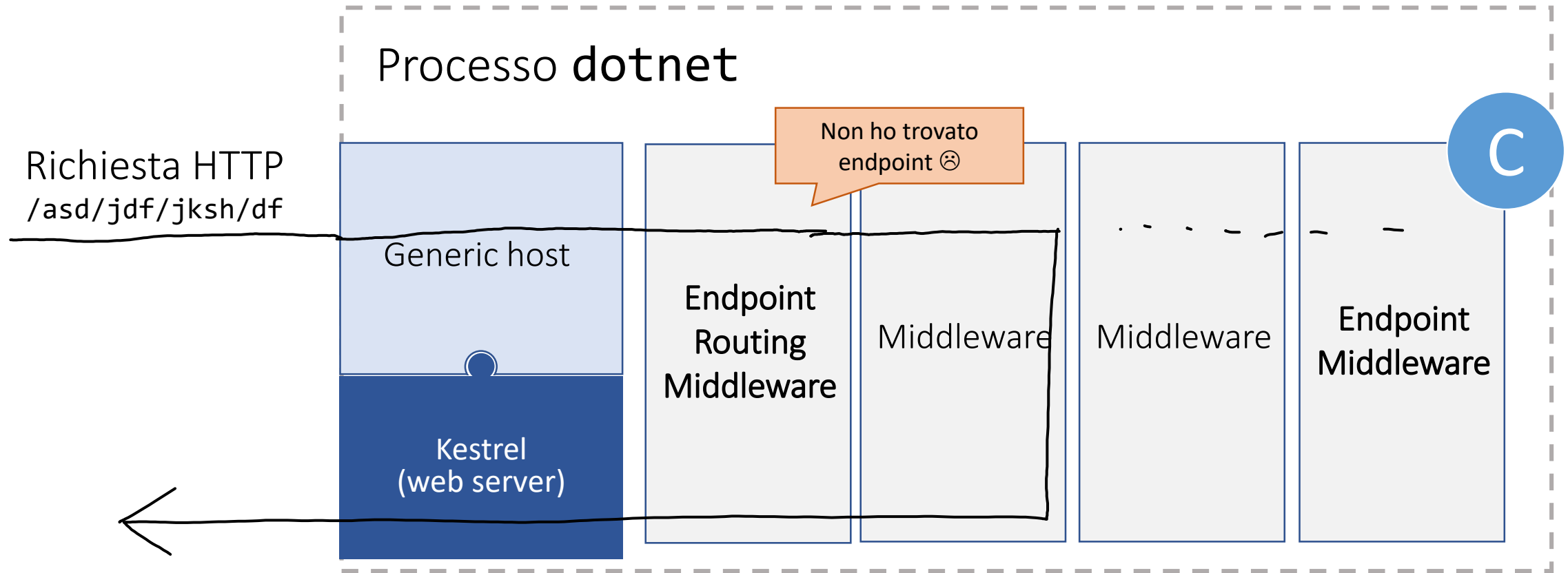
Routing tradizionale



Endpoint Routing



Endpoint Routing



SQL Ser... X

SQL Serve

(locald

(locald





Projects -

NuGet: WebApplication14 X

Browse Installed **Updates 16**

Search (Ctrl+L) Include prerelease Package source: nuget.org

Select all packages Update

<input type="checkbox"/>	 Microsoft.CodeDom.Providers.DotNetC v2.0.0 Replacement CodeDOM providers that use the n... v2.0.1
<input type="checkbox"/>	 Microsoft.jQuery.Unobtrusive.Validatio v3.2.4 The jQuery Unobtrusive Validation library comple... v3.2.11
<input type="checkbox"/>	 Newtonsoft.Json by James Newton-King v11.0.1 Json.NET is a popular high-performance JSON f... v12.0.2
<input type="checkbox"/>	 System.Diagnostics.DiagnosticSource v4.4.1 Provides Classes that allow you to decouple cod... v4.6.0

Output

Show output from:

Data Tools Operations Package Manager Console Error List Output

Solution Explorer

Search Solution Explorer (Ctrl+è)

Solution 'WebApplication14' (1 project)

WebApplication14

Connected Services

Properties

References

Add Reference...

Add Service Reference...

Add Connected Service

Add Analyzer...

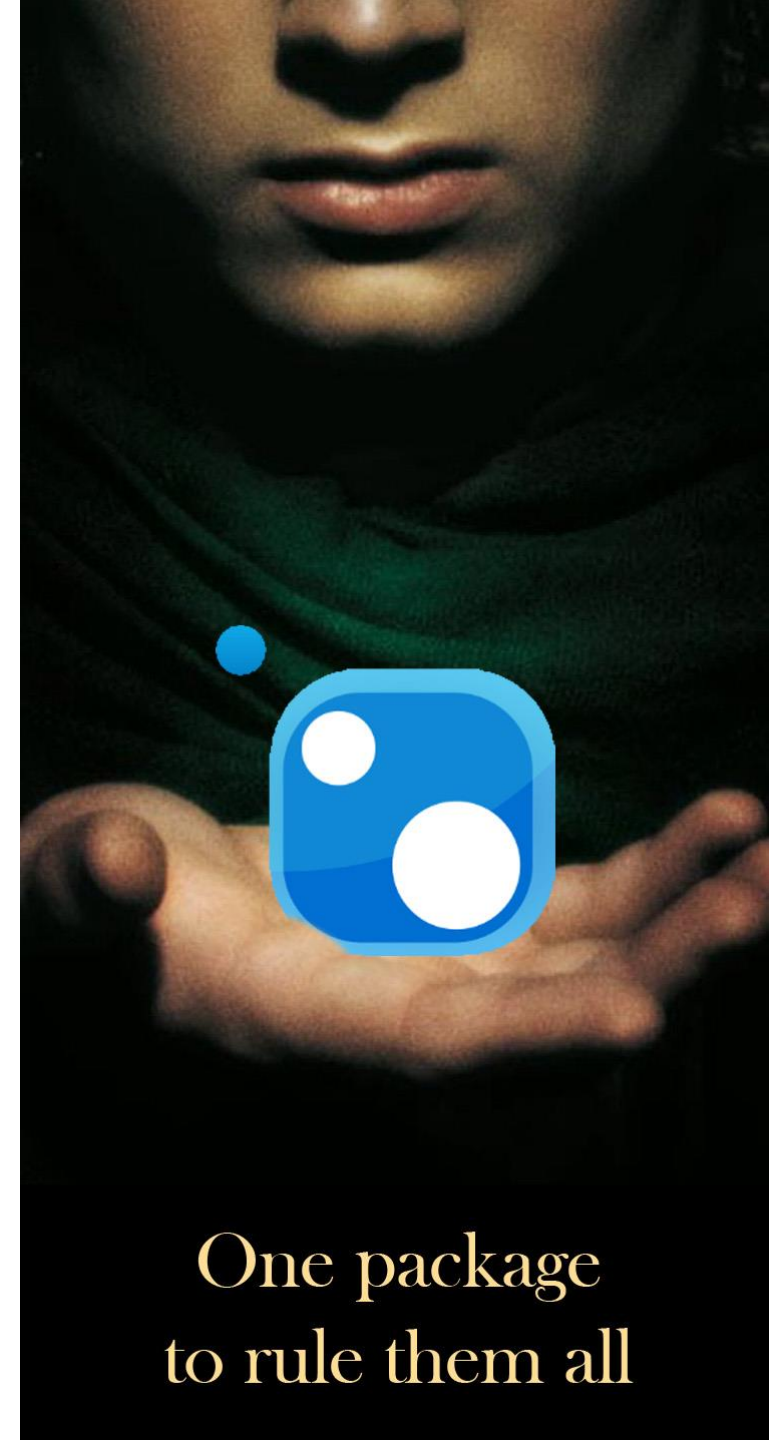
Manage NuGet Packages...

Scope to This

New Solution Explorer View

dotnet-outdated

- È un pacchetto NuGet che ci permette di verificare se ci sono aggiornamenti per gli altri pacchetti referenziati dall'applicazione;
- Si installa con il comando:
`dotnet tool install --global dotnet-outdated`
- Poi, per eseguire l'analisi:
`dotnet outdated`
- Per aggiornarli:
`dotnet outdated -u`



One package
to rule them all

Compilazione dinamica delle view Razor

- Installiamo il pacchetto NuGet con il seguente comando:

```
dotnet add package Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation
```

- Integriamo così la chiamata ad AddMvc nella classe Startup:

```
services.AddMvc(options =>
{
    //...
})
.SetCompatibilityVersion(CompatibilityVersion.Version_3_0)
.AddRazorRuntimeCompilation();
```

Referenziare un pacchetto in maniera condizionale

- Nel file .csproj aggiungiamo l'attributo Condition

```
<PackageReference Include="Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation"  
                  Version="3.0.0" Condition="'$(Configuration)' == 'Debug'" />
```

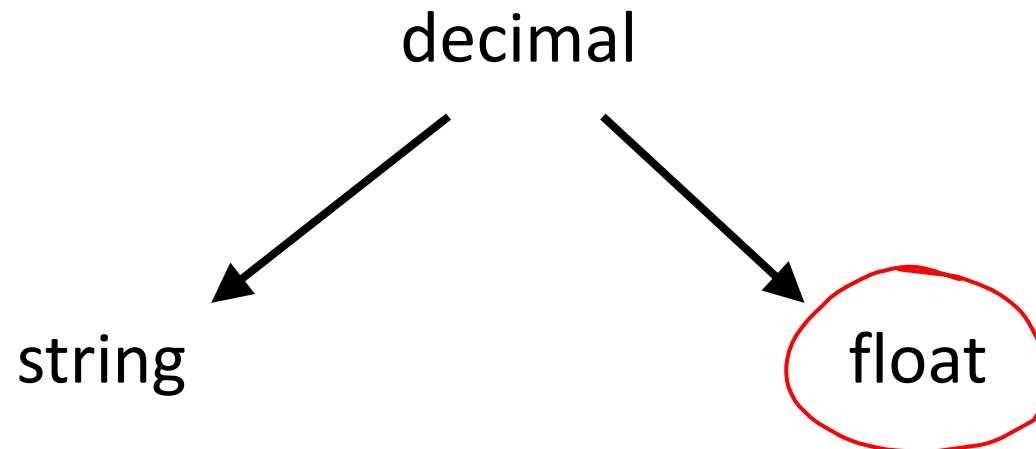
- Integriamo così la chiamata ad AddMvc nella classe Startup:

```
services.AddMvc(options =>  
{  
    //...  
})  
.SetCompatibilityVersion(CompatibilityVersion.Version_3_0)  
#if DEBUG  
.AddRazorRuntimeCompilation() <  
#endif  
;
```


EFCore 3.0 ci avvisa di possibili anomalie

- SQLite non può rappresentare il tipo decimal;
- Dobbiamo scegliere se trattarlo come string o come float;

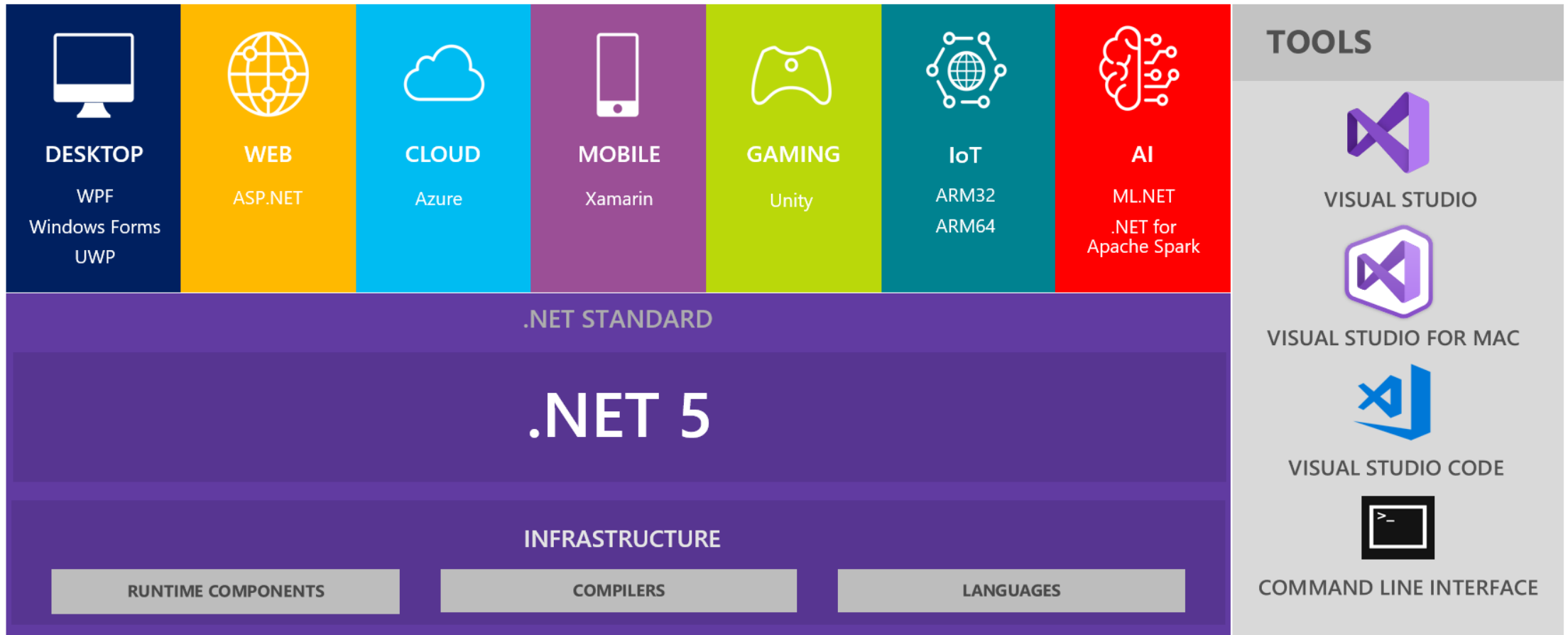
```
builder.Property(money => money.Amount).HasConversion<float>();
```



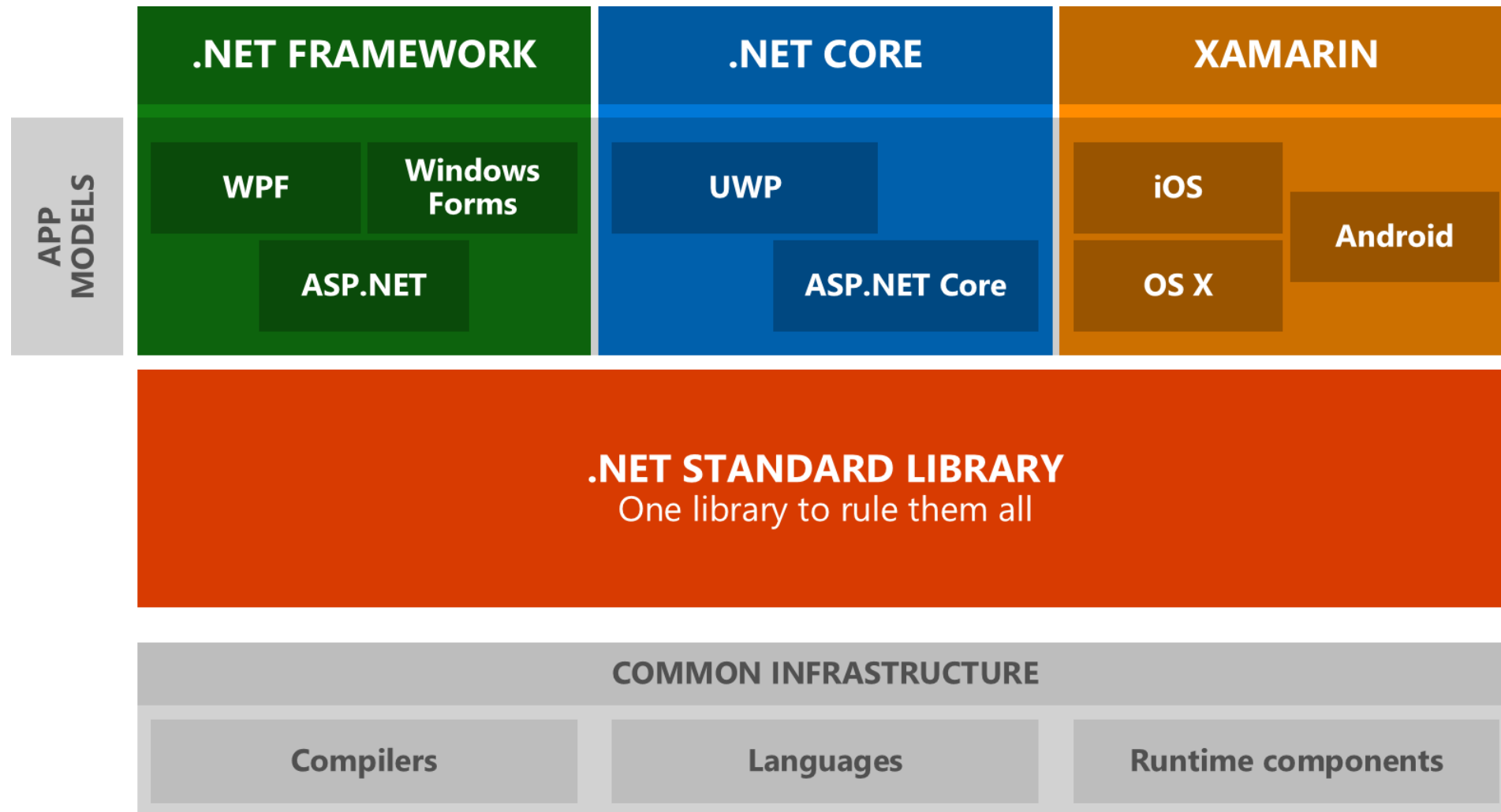
Il futuro di .NET



Il futuro di .NET



Mentre, attualmente...



Attenzione

- Non tutte le tecnologie 'classiche' saranno supportate da .NET 5.
- ~~WebForms~~ → MVC, Razor Pages, Blazor, Angular, Vue.js, ...
- ~~WCF~~ → WebAPI o gRPC.

Espressioni

```
int age = 18;
```

```
double circumference = radius * Math.PI * 2;
```

```
string result = age >= 18 ? "Can drive" : "Cannot drive";
```

switch expressions

```
string column = "title";

string orderby = column switch
{
    "title" => "ORDER BY Courses.Title",
    "price" => "ORDER BY Courses.CurrentPrice_Amount",
    "rating" => "ORDER BY Courses.Rating",
    _ => ""
};
```

switch expressions (tuple di valori)

```
string column = "title";  
bool ascending = true;  
  
string orderby = (column, ascending) switch  
{  
    ("title", true) => "ORDER BY Courses.Title ASC",  
    ("title", false) => "ORDER BY Courses.Title DESC",  
    ("price", true) => "ORDER BY Courses.CurrentPrice_Amount ASC",  
    ("price", false) => "ORDER BY Courses.CurrentPrice_Amount DESC",  
    ("rating", true) => "ORDER BY Courses.Rating ASC",  
    ("rating", false) => "ORDER BY Courses.Rating DESC",  
    _ => ""  
};
```


switch expressions (tipi complessi)

```
Apple apple = new Apple { Color = "Red", Weight = 280 };
```

```
string name = apple switch  
{  
    { Color: "Green" } => "Granny Smith",  
    { Color: "Yellow" } => "Golden delicious",  
    { Color: "Red" } when apple.Weight >= 200 => "Stark delicious",  
    { Color: "Red" } when apple.Weight < 200 => "Red delicious",  
    _ => "Unknown"  
};
```

Async streams

VANTAGGI

- Eliminiamo il passaggio nel DataSet;
- Meno duplicazione di dati in memoria;
- Più leggibile (?).

SVANTAGGI

- Non possiamo inviare più query in un solo round-trip al database;
 - Cioè con un solo SqlCommand.