.NET 11: non chiamatelo un semplice upgrade

Nicola Iarocci
nicolaiarocci.com · invoicetronic.com
18 novembre 2026





Lodestar
IT'S TIME TO INNOVATE

.NET Conference Italia 2026

CONTESTO

Perché parlarne adesso



STS = **Standard** Term Support: ora 24 mesi. LTS: 36 mesi.

LA TESI

Le novità nascono in basso e salgono



Framework	ASP.NET Core · EF Core · Blazor
Librerie	validazione asincrona · FullJoin · System.Text.Json
Linguaggio	union · gerarchie chiuse
Runtime	Runtime Async

Quando una novità attraversa quattro livelli, aggiornare non è più solo cambiare un numero nel csproj.

RUNTIME

Runtime Async: chi esegue il codice async

Fino a .NET 10

Il compilatore riscrive il metodo in una classe nascosta: un segnalibro che ricorda dove ripartire dopo ogni await.

Con .NET 11 anteprima

Il segnalibro lo tiene il runtime. Il metodo resta quello che avete scritto.

```
<Features>runtime-async=on</Features>
```

nel .csproj · le librerie di .NET 11 funzionano già così

DEMO 1

Lo stack, dal vivo

1. `dotnet run -c Release`
2. nel .csproj decommento la riga
   `<Features>$(Features);runtime-async=on</Features>`
3. `dotnet run -c Release`

COSA ABBIAMO VISTO, E COSA NO

17 → 5

righe sullo stack, quando ogni await trova il lavoro già pronto: il caso più comune

Quando il metodo aspetta davvero
l’elenco si accorcia in tutti e due i modi

Messaggi di errore nei log
identici nei due modi

RUNTIME ASYNC · 1/2

Cosa cambia davvero

1 Debugger e profiler più leggibili

mostrano le vostre chiamate, non gli ingranaggi

2 Meno memoria sprecata

il runtime riusa invece di creare oggetti a ogni await

3 Più veloce nei casi comuni

un await su un'operazione già conclusa prende la strada breve

4 Debug più semplice

i breakpoint funzionano, e lo step non entra nel codice generato

Bonus per tutti, anche senza attivarlo: in .NET 11 ogni await fa meno lavoro di contorno.

RUNTIME ASYNC · 2/2

Non fidatevi: l’ho misurato

≈ ½

il costo di ogni chiamata async

−44%

memoria, quando il metodo si ferma davvero

Cosa

La catena della demo, nei due modi

Attenzione

Sono nanosecondi: la vostra app non andrà il doppio

Dove si sente

Librerie e servizi molto carichi

Benchmark su Apple M2 Pro · è nel repo: rifatelo sulla vostra macchina

RUNTIME

Più veloce, senza fare niente

20×

Ricerche in dizionari e HashSet

confronti preparati in anticipo, senza reflection

1

Un solo runtime: le app MAUI passano a CoreCLR

come i server; su iOS in generale più veloce di Mono

0

Righe di codice da cambiare

basta ricompilare

20×: “fino a”, nei test di Microsoft.

Approfondimento: “Performance Improvements in .NET 11”, Stephen Toub · devblogs.microsoft.com

C# 15 · UNION 1/2

Uno, e uno solo, tra pochi tipi

```
public record CartaDiCredito(string Circuito, string UltimeCifre);
public record Bonifico(string Iban);
public record Contrassegno(decimal Supplemento);

public union MetodoDiPagamento(CartaDiCredito, Bonifico, Contrassegno);
```

Cosa

Un valore che è uno solo tra pochi tipi

Qui

Tre record senza niente in comune

Prima

Un’interfaccia vuota, oppure object

C# 15 · UNION 2/2

Il compilatore conosce i casi

```
string Descrivi(MetodoDiPagamento p) ⇒ p switch
{
    CartaDiCredito c ⇒ $"Carta {c.Circuito}",
    Bonifico b       ⇒ $"Bonifico su {b.Iban}",
    Contrassegno     ⇒ "Contrassegno",
};   // nessun default
```

Conversione

Ogni caso diventa una union da solo

Esaustività

Il compilatore verifica che gestiate tutti i casi

Niente default

Se manca un caso, ve lo dice lui

C# 15

Gerarchie chiuse: tutti i figli sono noti

```
public closed record StatoOrdine;
public record InAttesa : StatoOrdine;
public record Spedito(DateOnly Il, string Corriere) : StatoOrdine;
public record Annullato(string Motivo) : StatoOrdine;
```

union

casi indipendenti, anche tipi che non controllate voi

closed

casi con una base comune, derivabili solo nel proprio assembly

DEMO 2

Dal modello al contratto HTTP

```
dotnet build   # dopo aver aggiunto Consegnato
curl -s localhost:5080/ordini/1/pagamento | jq
curl -s localhost:5080/openapi/v1.json | jq '.components.schemas.MetodoDiPagamento'
```

UNION E JSON · 1/2

Rileggere una union

```
[JsonUnion(TypeClassifier =
    typeof(JsonUnionTypeStructuralClassifier))]
public union MetodoDiPagamento(…);
```

```
{"iban":"IT60X0542811101000000123456"}  → Bonifico
```

Il problema

Il JSON contiene solo i dati, non il tipo

Di default

Tre oggetti: non sa scegliere, errore

La soluzione

Riconoscere il caso dai nomi dei campi

UNION E JSON · 2/2

La gerarchia chiusa si spiega da sola

```
[JsonPolymorphic(
    InferClosedTypePolymorphism = true)]
public closed record class StatoOrdine;
```

```
{"$type":"Spedito","il":"2026-11-20","corriere":"BRT"}
```

Novità di .NET 11

Trova da solo tutti i tipi derivati

Nel JSON

Anche il nome del tipo, in $type

In lettura

Nessun dubbio, anche con campi uguali

Senza l’opzione: {} — dati persi, nessun errore

C# 15 · PICCOLE COMODITÀ 1/3

Parametri alle collezioni con with

```
string[] valori = ["uno", "due", "tre"];

List<string> nomi =
    [with(capacity: valori.Length * 2), .. valori];

HashSet<string> set =
    [with(StringComparer.OrdinalIgnoreCase),
     "Hello", "HELLO"];

// set contiene un solo elemento
```

Cosa

I parametri del costruttore, dentro le parentesi quadre

Esempi

Capacità iniziale, criterio di confronto

Prima

Si tornava al costruttore classico

C# 15 · PICCOLE COMODITÀ 2/3

Parentesi quadre con gli extension

```
public static class SequenzaExtensions
{
    extension(IEnumerable<int> sequenza)
    {
        public int this[int indice] ⇒
            sequenza.ElementAt(indice);
    }
}

var numeri = Enumerable.Range(1, 10);
int terzo = numeri[2];  // come un array
```

Da sempre

Extension method: Where e Select di LINQ

C# 14

Il blocco extension: anche le proprietà

C# 15

Anche l’accesso con le parentesi quadre

C# 15 · PICCOLE COMODITÀ 3/3

break e continue con etichetta

```
List<Riga> righeLibere = [];

outer: foreach (var riga in griglia)
{
    foreach (var cella in riga.Celle)
        if (cella.Bloccata) continue outer;

    righeLibere.Add(riga);
}
```

Cosa

Saltare a un ciclo esterno, per nome

Qui

Una cella bloccata scarta tutta la riga

Con un continue normale

Si passerebbe solo alla cella successiva

FILO 1 · VALIDAZIONE 1/2

Una regola che va sul database

Problema · questa email esiste già?

Fino a .NET 10 · solo regole sincrone

.NET 11 · async, scritta nel modello

```
public sealed class EmailUnicaAttribute : AsyncValidationAttribute
{
    protected override async Task<ValidationResult?> IsValidAsync(
        object? value, ValidationContext context, CancellationToken ct)
    {
        var utenti = context.GetRequiredService<IUtentiService>();
        return value is string email && await utenti.EsisteAsync(email, ct)
            ? new ValidationResult("Email già registrata.")
            : ValidationResult.Success;
    }
}
```

FILO 1 · VALIDAZIONE 2/2

La stessa regola sale fino ai framework

DataAnnotations

la regola, scritta una volta nel modello

→

Minimal API

AddValidation: controllo prima dell'endpoint, in parallelo quando può

→

Blazor

il form aspetta e può mostrare "sto verificando"

```
builder.Services.AddValidation();  // e la regola vale per tutte le richieste
```

Una regola nel modello: la ritrovate nelle API e nei form.

FILO 2

FullJoin, da LINQ al database

```
var risultati = await context.Clienti
    .FullJoin(context.Ordini,
        c ⇒ c.Id, o ⇒ o.ClienteId,
        (c, o) ⇒ new { Cliente = c, Ordine = o })
    .ToListAsync();
```

EF Core 11 lo traduce

```
SELECT … FROM [Clienti] AS [c]
FULL JOIN [Ordini] AS [o] ON [c].[Id] = [o].[ClienteId]
```

Clienti senza ordini, ordini senza cliente: ci sono tutti.

EF CORE 11 per tutti

SQL più semplice, query più veloci

−29%

tempo della query, in uno scenario comune

Quando

Caricate un’entità insieme ai dati collegati

Cosa cambia

Spariscono join e ordinamenti inutili

Cosa fate voi

Niente

Benchmark del team EF Core su SQL Server: i risultati variano con schema e dati.

FILO 3

Osservabilità già inclusa

```
builder.Services.AddMemoryCache(o ⇒ o.TrackStatistics = true);

builder.Services.AddOpenTelemetry()
    .WithTracing(t ⇒ t.AddSource("Microsoft.AspNetCore"))
    .WithMetrics(m ⇒ m.AddMeter("Microsoft.Extensions.Caching.Memory.MemoryCache"));
```

Richieste HTTP

metodo, indirizzo, codice di risposta: le fornisce il framework

Non serve più:

`OpenTelemetry.Instrumentation.AspNetCore`

Cache in memoria

richieste, dati trovati e non trovati, elementi: basta attivarle

Non serve più:

il vostro codice che leggeva le statistiche e le pubblicava

OPENAPI 3.2 DI DEFAULT · 1/3

Il metodo HTTP QUERY

```
app.MapMethods("/ricerca", ["QUERY"],
    (FiltroRicerca f) ⇒ Cerca(f));
```

Nel documento generato

```
"/ricerca": {
  "query": { "requestBody": { … } }
}
```

Cos’è

Un metodo proposto per le ricerche, ancora in standardizzazione

Perché

I filtri vanno nel corpo, non nell’indirizzo

Con la 3.2

Compare accanto a get e post

OPENAPI 3.2 DI DEFAULT · 2/3

Obsolete diventa deprecated

```
app.MapGet("/prodotti/legacy/{id}", GetLegacy);

[Obsolete("Usa /prodotti/{id}")]
static ProdottoLegacy GetLegacy(int id) ⇒ …;
```

Nel documento generato

```
"/prodotti/legacy/{id}": {
  "get": { "deprecated": true }
}
```

Cosa

L’attributo Obsolete arriva nel contratto

Chi usa il client

Vede l’avviso, senza codice in più

Vale anche per

Tipi e singole proprietà

OPENAPI 3.2 DI DEFAULT · 3/3

Se un vostro strumento non legge ancora la 3.2

```
builder.Services.AddOpenApi(o ⇒
    o.OpenApiVersion = OpenApiSpecVersion.OpenApi3_1);
```

Attenzione

È un cambio di default, senza preavviso nel codice

Soluzione

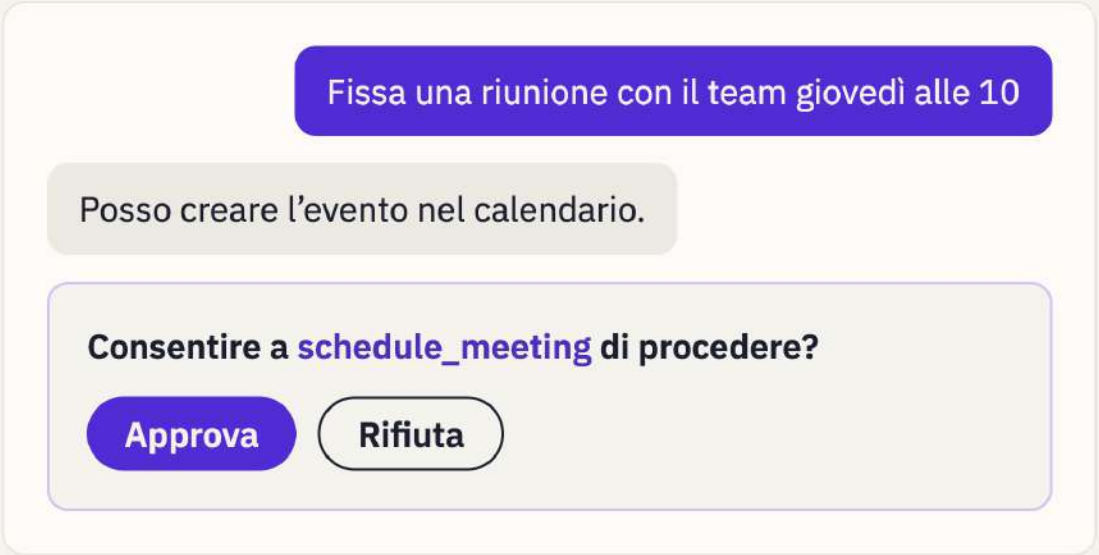
Si torna alla 3.1 con un’impostazione

Se non compila

Manca using Microsoft.OpenApi

E POI C’È QUESTO sperimentale

Componenti Blazor per gli agenti AI



Risposte che arrivano mentre vengono scritte

Le azioni dell’agente mostrate con i vostri componenti

Il permesso dell’utente prima di un’azione irreversibile

Dati condivisi tra app e agente

Microsoft.AspNetCore.Components.AI · funziona con IChatClient

IL REQUISITO CHE NESSUNO LEGGE

I processori di oltre 13 anni fa non sono più supportati

```
$ dotnet MiaApp.dll
The current CPU is missing one or more of the baseline instruction sets.
```

Controllate: VM con CPU “generica” · server datati · macchine di build

In cambio, il codice precompilato usa le istruzioni dei processori moderni.

Dettaglio tecnico: minimo x86-64-v2 · Windows su Arm richiede LSE

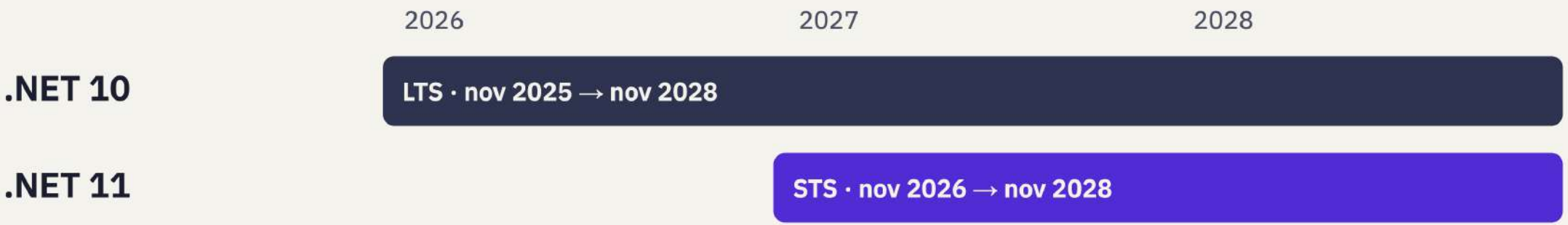
CHIUSURA

Prima di aggiornare

- [ ] **Processori:** niente CPU di oltre 13 anni fa (VM e macchine di build)
- [ ] **OpenAPI 3.2 di default:** i vostri strumenti la leggono?
- [ ] **EF Core su SQL Server:** di default presume SQL Server 2022
- [ ] **EF Core su Cosmos DB:** System.Text.Json al posto di Newtonsoft
- [ ] **Blazor QuickGrid:** pagina e ordinamento nell'indirizzo, link al posto dei bottoni

CHIUSURA

.NET 11 o .NET 10?



Stessa fine supporto. Un anno di maturità di differenza.

.NET 10

sistemi critici ancora su .NET 8 o 9, da spostare adesso: la scelta più matura

.NET 11

progetti nuovi

Grazie. Domande?

Nicola Iarocci
nicolaiarocci.com · invoicetronic.com

Demo, benchmark e materiale su
`github.com/nicolaiarocci/dotnet11`



aspitalia.com

Lodestar
IT'S TIME TO INNOVATE

.NET Conference
Italia 2026