Day 40/60 - Understanding ConfigureAwait in C#

When working with async/await, ConfigureAwait(bool) plays a key role in optimizing performance and avoiding deadlocks in certain scenarios.

What is ConfigureAwait(false)?
By default, await tries to resume execution on the original context (e.g., UI thread). ConfigureAwait(false) tells the runtime not to capture the context, improving performance in non-UI applications.

Example: Default Behavior (Captures Context)

```
await SomeAsyncMethod(); // Resumes on the original context (e.g., UI thread)
```

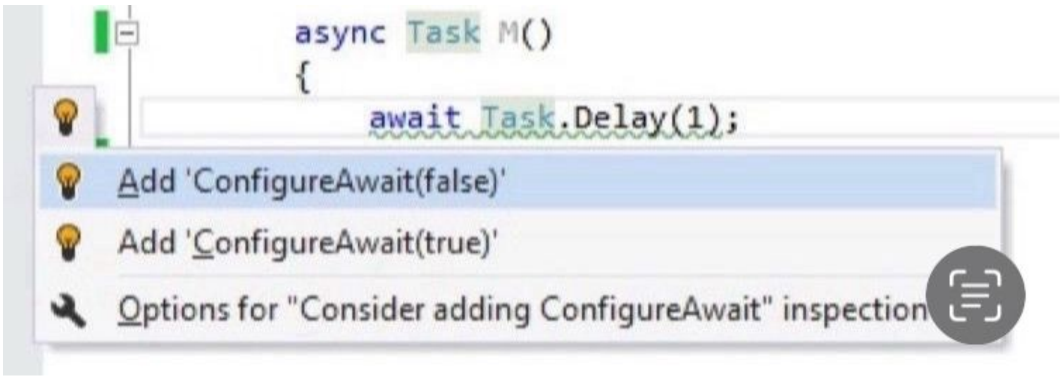
Using ConfigureAwait(false) (No Context Capture)

```
await SomeAsyncMethod().ConfigureAwait(false); // Resumes on a thread pool thread
```

When to Use It?
Use ConfigureAwait(false) in library code, APIs, and background tasks to improve performance.
Avoid it in UI applications (e.g., WPF, WinForms) where the UI thread must be updated after awaiting.

ConfigureAwait(false) reduces unnecessary thread switches, making async code more efficient and scalable in .NET applications.

#dotnet #csharp #async #ConfigureAwait #performance



```
async Task M()
{
    await Task.Delay(1);
```

Add 'ConfigureAwait(false)'
Add 'ConfigureAwait(true)'
Options for "Consider adding ConfigureAwait" inspection

56Abhinn Mishra and 55 others