Avoiding Common Pitfalls

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Wrapping Legacy Code

Blocking Async Code

Modifying Shared State

Offloading Legacy Code to a Background Thread

Legacy code, like long-running algorithms, is computational bound code

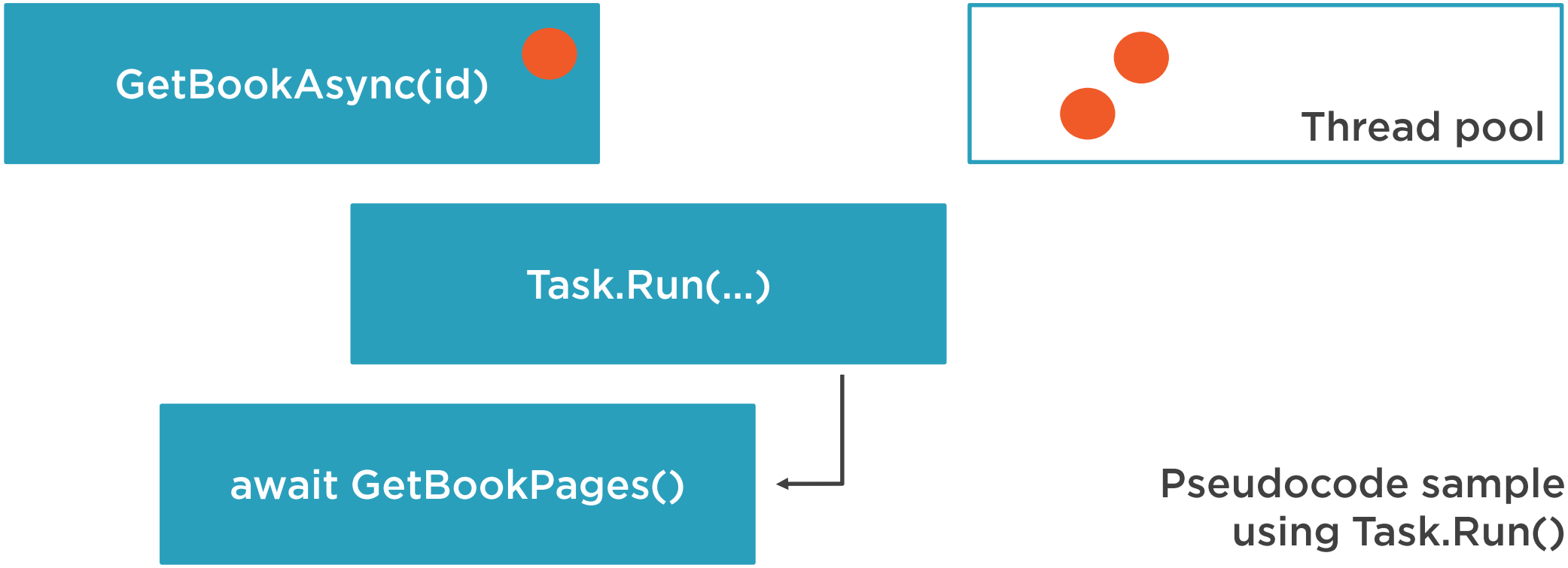
These can be offloaded to a background thread using async/await

- Can run concurrently

Demo

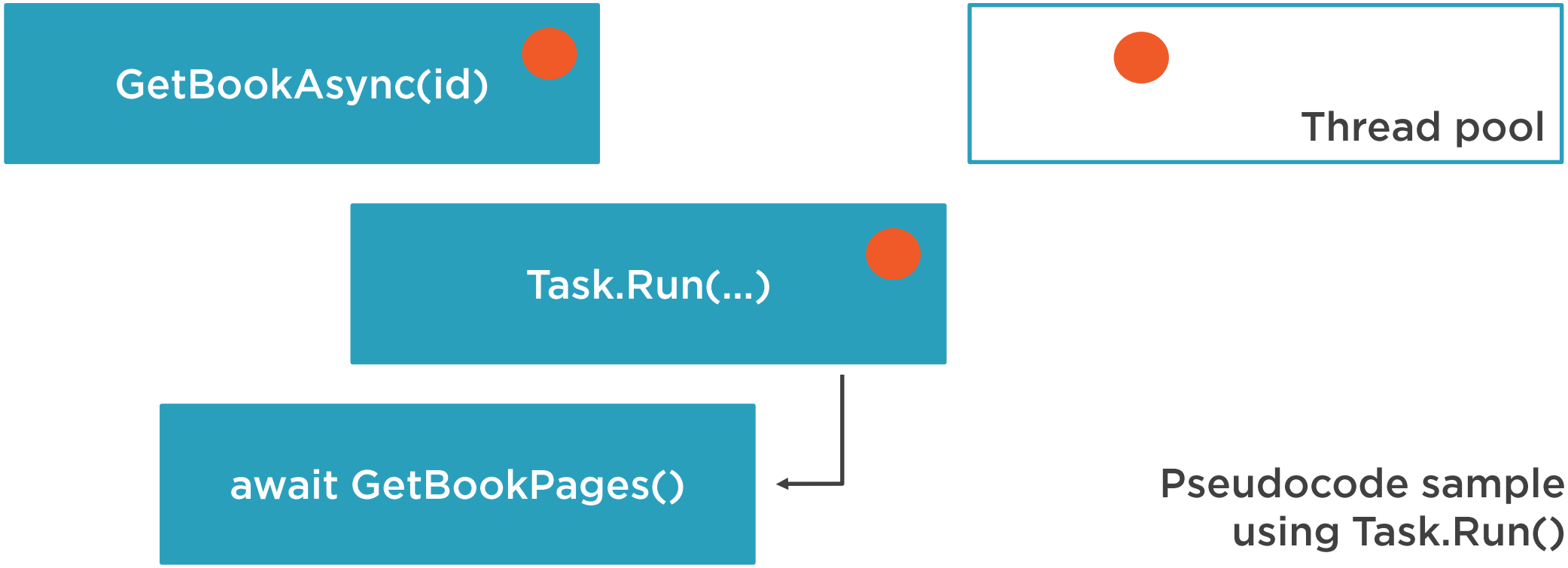
Wrapping Synchronous Code with Task.Run()

Pitfall #1: Using Task.Run() on the Server



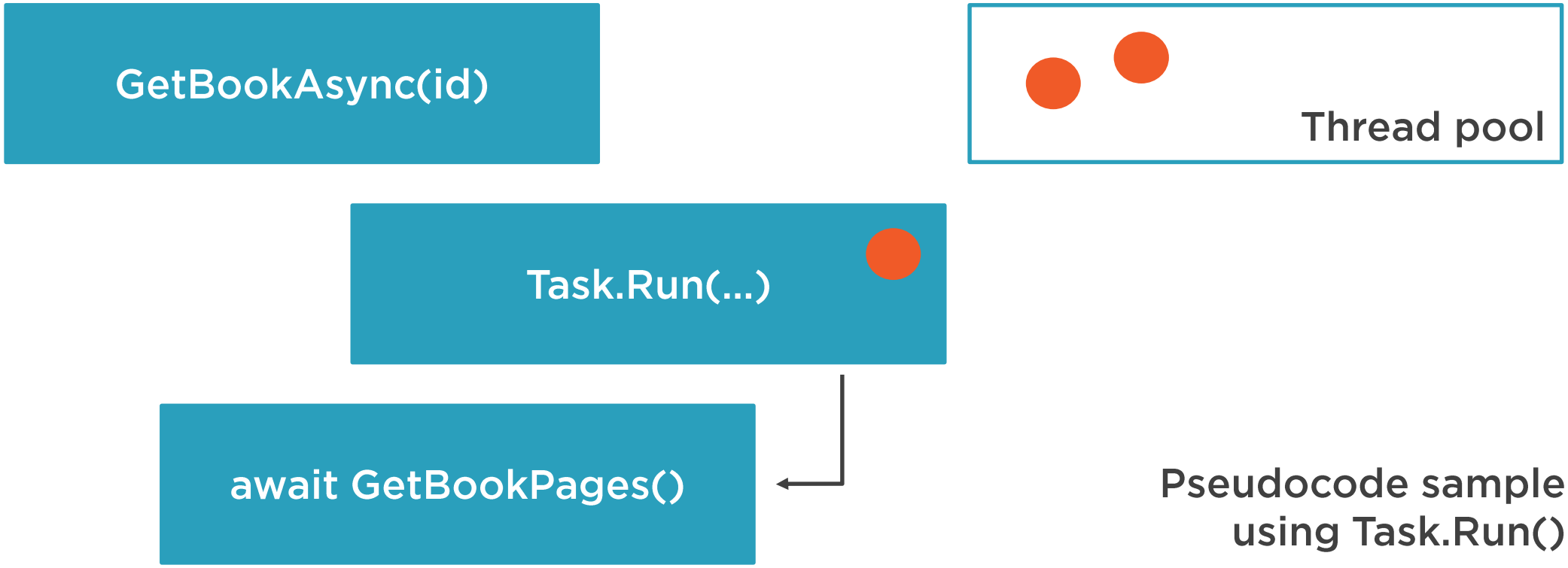
Pseudocode sample
using Task.Run()

Pitfall #1: Using Task.Run() on the Server



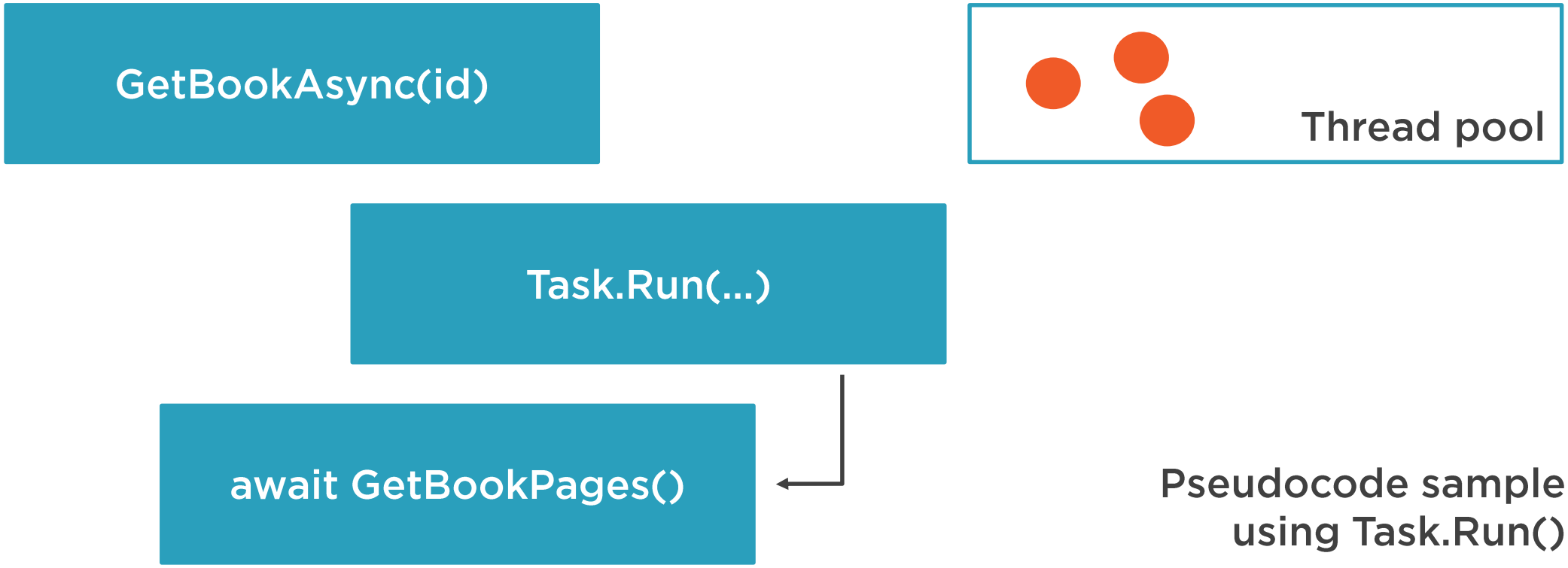
Pseudocode sample
using Task.Run()

Pitfall #1: Using Task.Run() on the Server



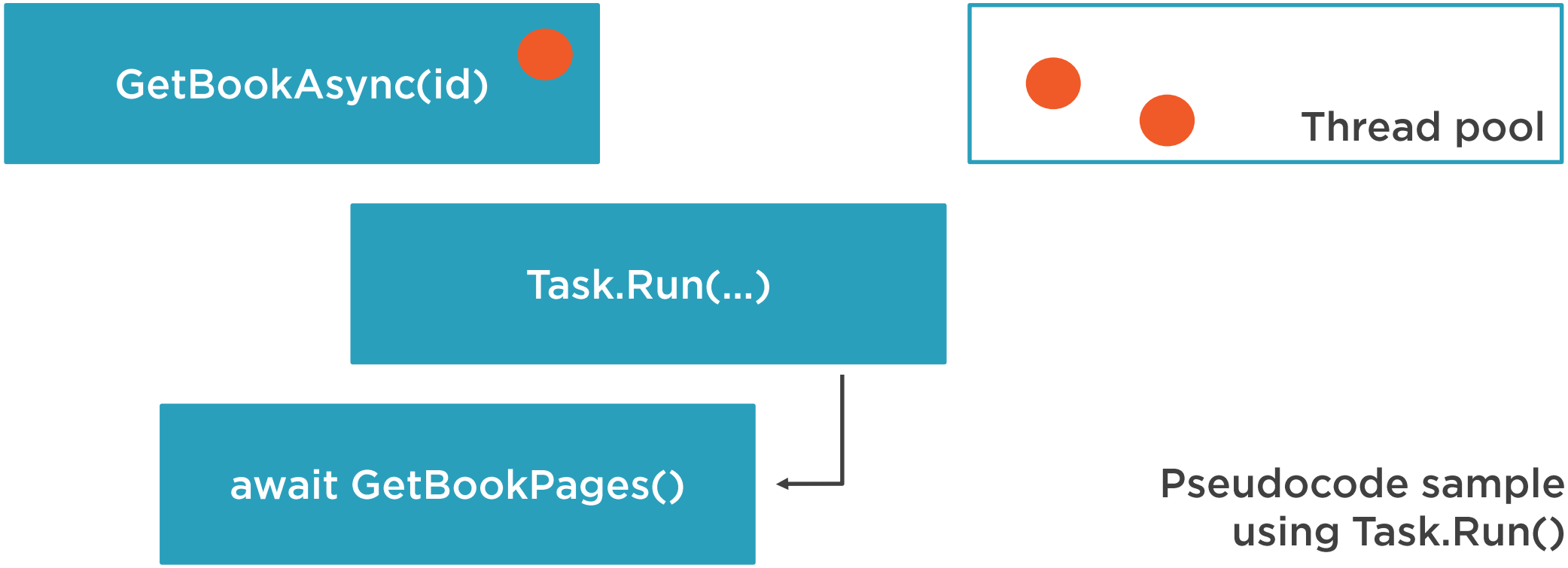
Pseudocode sample
using Task.Run()

Pitfall #1: Using Task.Run() on the Server



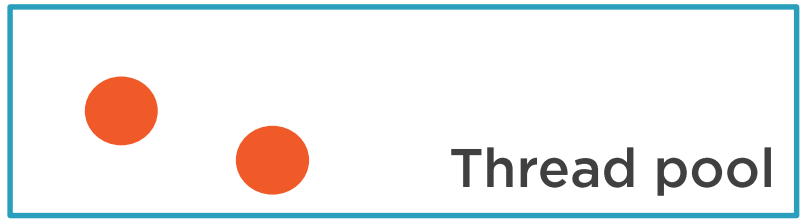
Pseudocode sample
using Task.Run()

Pitfall #1: Using Task.Run() on the Server



Pseudocode sample
using Task.Run()

Pitfall #1: Using Task.Run() on the Server



Pseudocode sample
without Task.Run()

Pitfall #1: Using Task.Run() on the Server

ASP.NET Core is not optimized for Task.Run()

- Creates an unoptimized thread
- Causes overhead

Pitfall #1: Using Task.Run() on the Server

Task.Run() on the server decreases scalability

It's intended for use on the client (eg: to keep the UI responsive)

Demo

Blocking Async Code

Pitfall #2: Blocking Async Code

Task.Wait() and Task.Result() block the calling thread

- Thread isn't returned to the thread pool

Blocking async code hurts scalability

Pitfall #2: Blocking Async Code

ASP.NET Core doesn't have a synchronization context (the old ASP.NET does)

- Improves performance
- Makes it easier to write async code

Demo

Modifying Shared State

Pitfall #3: Modifying Shared State

Different threads might manipulate the same state at the same time

- Correctness cannot be guaranteed

Summary

Don’t use Task.Run() on the server

- Hurts scalability

Don’t block async code

- Hurts scalability

Don’t modify shared state

- State can’t be guaranteed


you are ready
to be Awesome !!!
@kevinDockx