💡 ASP.NET Core khuyến khích dùng async/await → **giải phóng thread** khi đang chờ I/O.

**🎯 Vấn đề: Thread là tài nguyên giới hạn**

* Trong ASP.NET Core, các HTTP request được xử lý bởi các **thread trong ThreadPool**.
* Mỗi request chiếm **1 thread**.
* Nếu thread bị **“đứng chờ I/O”** (chờ đọc DB, file, API...), thì thread đó **bị chiếm** mà không làm gì.

Khi có quá nhiều thread bị chờ I/O, **server bị nghẽn**, không còn thread để xử lý các request mới → gọi là **thread starvation**.

**🧠 Giải pháp: async/await**

* Khi bạn dùng await cho các thao tác I/O như:
  + await dbContext.SaveChangesAsync()
  + await httpClient.GetAsync()
  + await fileStream.ReadAsync()

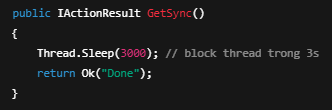
**.NET sẽ tạm *trả lại thread* về ThreadPool trong khi chờ I/O hoàn thành**.

➡ Điều này giúp server **không tốn thread trong lúc chờ**, tăng khả năng **xử lý song song hàng ngàn request**.

**🎮 Ví dụ minh họa:**

**Không dùng async – giữ thread bị block:**

**Mỗi request giữ 1 thread suốt 3 giây → nghẽn server nếu có nhiều request.**

****

**Dùng async/await – không giữ thread khi chờ:**

**Thread được trả lại ThreadPool trong lúc chờ, không bị block**

