**Tiến trình (Process) và Luồng (Thread)** – hiểu rõ giúp bạn tối ưu hoá hiệu năng khi xây dựng **ứng dụng web**, **microservices**, hoặc **xử lý song song**.

**🧠 I. Khái niệm chung: Process vs Thread**

| **Khái niệm** | **Tiến trình (Process)** | **Luồng (Thread)** |
| --- | --- | --- |
| Là gì? | Đơn vị thực thi độc lập trong OS | Đơn vị nhỏ hơn, sống trong tiến trình |
| Bộ nhớ | Mỗi process có vùng bộ nhớ riêng | Các thread **chia sẻ** bộ nhớ trong process |
| Độ nặng | Tạo process tốn nhiều tài nguyên hơn | Thread nhẹ, tạo nhanh hơn |
| Tách biệt | Process cách ly hoàn toàn | Thread có thể ảnh hưởng lẫn nhau nếu lỗi |
| Ví dụ | Chrome mở 3 tab = 3 process | 1 tab có thể có nhiều thread để render, tải JS |

**🔧 II. Ví dụ thực tế (máy tính nói chung)**

**Desktop App (VD: MS Word)**

* Khi mở MS Word → tạo 1 **Process**
* Trong đó:
  + 1 Thread hiển thị giao diện
  + 1 Thread xử lý gõ bàn phím
  + 1 Thread tự động lưu (AutoSave)

MS Word = **1 Process** với **nhiều Thread**

**🌐 III. Trong dịch vụ Web (ASP.NET Core)**

**Khi bạn chạy một web API :** *dotnet run*

* OS tạo ra **1 tiến trình** (dotnet.exe)
* Bên trong tiến trình đó:
* ASP.NET Core dùng **ThreadPool** (luồng nền) để xử lý **mỗi HTTP request**

**Dòng chảy request đơn giản:**

Client gửi HTTP request →

Kestrel nhận →

Thread từ ThreadPool được mượn →

Middleware/Controller xử lý →

Trả về response →

ThreadPool trả lại thread

**🧵 3. Sự khác biệt trong Web App**

| **Tác vụ** | **Tiến trình** | **Luồng xử lý** |
| --- | --- | --- |
| Khởi chạy ứng dụng ASP.NET | 1 process | nhiều thread xử lý song song |
| Nhận nhiều request cùng lúc | 1 process | mỗi request dùng 1 thread riêng |
| Gọi DB, gửi mail | Task async | không chiếm thread trong khi chờ |