“Tôi không muốn thiết kế lại bánh xe, nhưng tôi muốn hiểu cách tạo ra nó.”

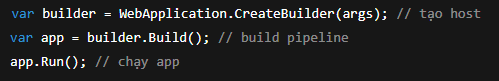
🧠 Tầng trừu tượng thấp → cao trong ASP.NET Core Web API:

**1. Hosting & Web Server (Tầng vật lý chạy app)**

* ASP.NET Core chạy trên một **host** (Kestrel hoặc IIS).
* Khi bạn chạy dotnet run, ASP.NET khởi tạo một **generic host** (IHost) hoặc **web host** (IWebHost).

**Cốt lõi:**  
Program.cs là nơi khởi động host, cấu hình services, middleware.

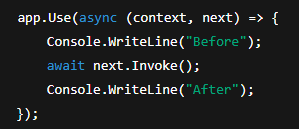
Ví dụ (ASP.NET Core 6+ Minimal Hosting):



👉 Đây là tầng gốc – gần giống *bare metal* – chỉ là nơi đưa app online, chưa có gì khác.

**⚙️ 2. Middleware Pipeline (Tầng request pipeline)**

* ASP.NET dùng **middleware** (dựa trên RequestDelegate) để xử lý HTTP Request theo **chuỗi pipeline**.
* Mỗi middleware có thể:
  + xử lý request,
  + chỉnh sửa response,
  + hoặc chuyển tiếp tới middleware tiếp theo (await next()).



Đây là chỗ để cấu hình:  
UseAuthentication, UseAuthorization, UseCors, UseRouting, UseEndpoints, v.v.

**🧩 3. Routing & Endpoint (Tầng điều hướng)**

* ASP.NET dùng **Endpoint Routing**:
  + Map các HTTP route tới các controller hoặc handlers cụ thể.



* Ở đây, nó tra route table để tìm Controller.Action tương ứng với URL được gọi.

**🧱 4. Controllers, Actions, Model Binding (Tầng xử lý logic)**

* Controllers nhận HTTP request → xử lý business logic → trả về HTTP response.
* ASP.NET tự động:
  + **Bind** dữ liệu từ URL, body, query → thành tham số ([FromBody], [FromQuery], [FromRoute]).
  + **Validation** bằng Data Annotations hoặc FluentValidation.
  + **Serialization** sang JSON khi return Ok(object).

**📦 5. Dependency Injection (Tầng kết nối logic)**

* ASP.NET Core có built-in **IoC Container**.
* Tất cả service (DbContext, Service, Repository, EmailSender,...) được inject qua constructor hoặc [FromServices].
* DI Container giải quyết việc quản lý vòng đời, phụ thuộc và testability.



**🗂 6. Business Layer & Repository (Tầng nghiệp vụ)**

* **Application Layer**: Dùng để viết logic nghiệp vụ (use case) → Giao tiếp qua interface và DTO.
* **Infrastructure Layer**: Là nơi gọi đến DB, lưu trữ, file, 3rd party.

💡 Clean Architecture giúp tách biệt các tầng này:

