**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

~~~~~~~~~~~~~

**BÁO CÁO MÔN CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI**

**ĐỀ TÀI: TÌM HIỂU VÀ ỨNG DỤNG CÔNG NGHỆ ASP.NET ĐỂ XÂY DỰNG**



**Giảng viên hướng dẫn:** Phạm Thi Vương

**Nhóm:**

Danh sách sinh viên thực hiện:

Dương Minh Trí – 3122410420

Đinh Bá Phong – 3122410305

Vũ Đăng Khoa – 3122410188

Đoàn Phong Lưu – 3122410227

Thành phố Hồ Chí Minh, tháng 9/2025

**NHẬN XÉT ĐÁNH GIÁ CỦA GIẢNG VIÊN**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**LỜI CẢM ƠN.**

- Lời đầu tiên, nhóm em xin bày tỏ lòng biết ơn sâu sắc đến thầy Phạm Thi Vương. Thầy không chỉ là người hướng dẫn tận tâm, mà còn là nguồn động viên vô cùng lớn, luôn đồng hành và giúp đỡ chúng em vượt qua mọi khó khăn trong suốt quá trình thực hiện đề tài. Những lời khuyên quý báu và sự chỉ dẫn nhiệt tình của thầy đã trang bị cho chúng em nền tảng kiến thức vững chắc để hoàn thành bài báo cáo này một cách tốt nhất.

- Chúng em cũng xin gửi lời tri ân chân thành đến Ban Giám hiệu Trường Đại học Sài Gòn, đặc biệt là Ban Chủ nhiệm Khoa Công nghệ thông tin và ngành Đào tạo, đã tạo điều kiện thuận lợi về mọi mặt, giúp chúng em có một môi trường học tập và nghiên cứu đầy đủ để có thể hoàn thành đề tài môn “Các Công Nghệ Lập Trình Hiện Đại”.

- Mặc dù đã cố gắng hết mình, nhưng do kiến thức và kinh nghiệm còn hạn chế, đề tài của chúng em chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những góp ý quý báu từ thầy để đề tài được hoàn thiện hơn, và chúng em có thể học hỏi, làm tốt hơn nữa trong những dự án sau.

Nhóm em xin chân thành cảm ơn.

MỤC LỤC chỗ này

**PHẦN 1: TỔNG QUAN VỀ CÔNG NGHỆ ASP.NET**

**Chương 1: Giới thiệu chung**

1. **Lịch sử hình thành và phát triển:**

* Tổ chức tạo ra: Microsoft.
* Thời điểm ra đời: Lần đầu được công bố vào năm 2002, mục đích để thay thế ASP (Active Server Pages).
* Bối cảnh và lý do ra đời:
  + ASP cũ chạy trên nền script engine, hiệu năng thấp, khó mở rộng.
  + Microsoft muốn xây dựng một framework mạnh mẽ hơn, hướng đối tượng, tích hợp chặt chẽ với .NET Framework để phát triển ứng dụng web hiện đại.
  + ASP.NET ra đời nhằm hỗ trợ phát triển web động, bảo mật, dễ bảo trì, có khả năng mở rộng tốt.

1. **Sự thay đổi qua các phiên bản:**

* **ASP.NET 1.0 (2002)**: Ra mắt cùng .NET Framework 1.0, hỗ trợ Web Forms.
* **ASP.NET 2.0 (2005)**: Cải tiến Web Forms, thêm Membership, Master Pages.
* **ASP.NET 3.5 (2007)**: Hỗ trợ AJAX, LINQ.
* **ASP.NET MVC (2009)**: Giới thiệu mô hình MVC, tách biệt logic và giao diện.
* **ASP.NET Web API (2012)**: Tối ưu cho xây dựng RESTful API.
* **ASP.NET Core (2016, tái cấu trúc lớn)**:
  + Mã nguồn mở.
  + Đa nền tảng (Windows, Linux, macOS).
  + Hiệu năng cao, kiến trúc module gọn nhẹ.
* **ASP.NET Core 5/6/7/8 (2020–2023)**: Hợp nhất với .NET 5+ (không còn tách .NET Framework/Core), hỗ trợ Minimal API, cải thiện hiệu năng, tích hợp container và cloud-native.

1. Hệ sinh thái:

* **Thư viện, framework, công cụ hỗ trợ**:
  + Entity Framework Core (ORM).
  + Blazor (SPA framework chạy trên WebAssembly hoặc server).
  + SignalR (real-time communicastion).
  + Identity (xác thực & phân quyền).
  + Visual Studio, Visual Studio Code.
* **Cộng đồng phát triển**:
  + Rất lớn, chủ yếu tập trung ở Microsoft Docs, GitHub, Stack Overflow.
  + Hỗ trợ chính thức từ Microsoft, cộng đồng đóng góp mạnh mẽ (ASP.NET Core open-source).

**Chương 2: Cơ hội nghề nghiệp và thị trường**

1. **Các vị trí công việc:**

* **ASP.NET Developer / .NET Developer**: Phát triển web/app với ASP.NET Core, Web API, Blazor.
* **Fullstack .NET Developer**: Kết hợp backend ASP.NET với frontend (Angular, React, Blazor).
* **DevOps Engineer (chuyên .NET)**: Triển khai, CI/CD cho ứng dụng ASP.NET Core trên Azure, AWS.
* **Software Engineer (C#/.NET)**: Phát triển sản phẩm tổng hợp (web, desktop, cloud).

1. **Nhu cầu tuyển dụng:**

**Hiện tại (2025):**

* Ở Việt Nam, ASP.NET Core nằm trong top 5 công nghệ backend được tuyển dụng nhiều (theo TopCV, ITviec).
* Thường xuất hiện trong job description yêu cầu kết hợp SQL Server, Azure, và frontend framework.

**Tương lai gần:**

* Nhu cầu ổn định, đặc biệt ở các doanh nghiệp lớn, ngân hàng, công ty outsource cho thị trường Âu-Mỹ.
* ASP.NET Core sẽ duy trì do Microsoft tiếp tục đầu tư mạnh mẽ.

1. **Mức lương tham khảo:**

**Ở Việt Nam, năm 2025:**

* Junior .NET Developer: 10–15 triệu VNĐ/tháng.
* Middle .NET Developer: 20–30 triệu VNĐ/tháng.
* Senior .NET Developer: 35–50 triệu VNĐ/tháng, có thể cao hơn tại các công ty quốc tế hoặc Fintech.

1. **Các công ty/sản phẩm tiêu biểu đang sử dụng:**

* **Toàn cầu**: Stack Overflow, GoDaddy, Dell, Visual Studio Team Services.
* **Việt Nam**: Nhiều ngân hàng (Techcombank, Vietcombank), tập đoàn lớn (FPT, Viettel), công ty outsource (TMA Solutions, NashTech).

**PHẦN 2:  NỘI DUNG CỐT LÕI VÀ THỰC HÀNH**

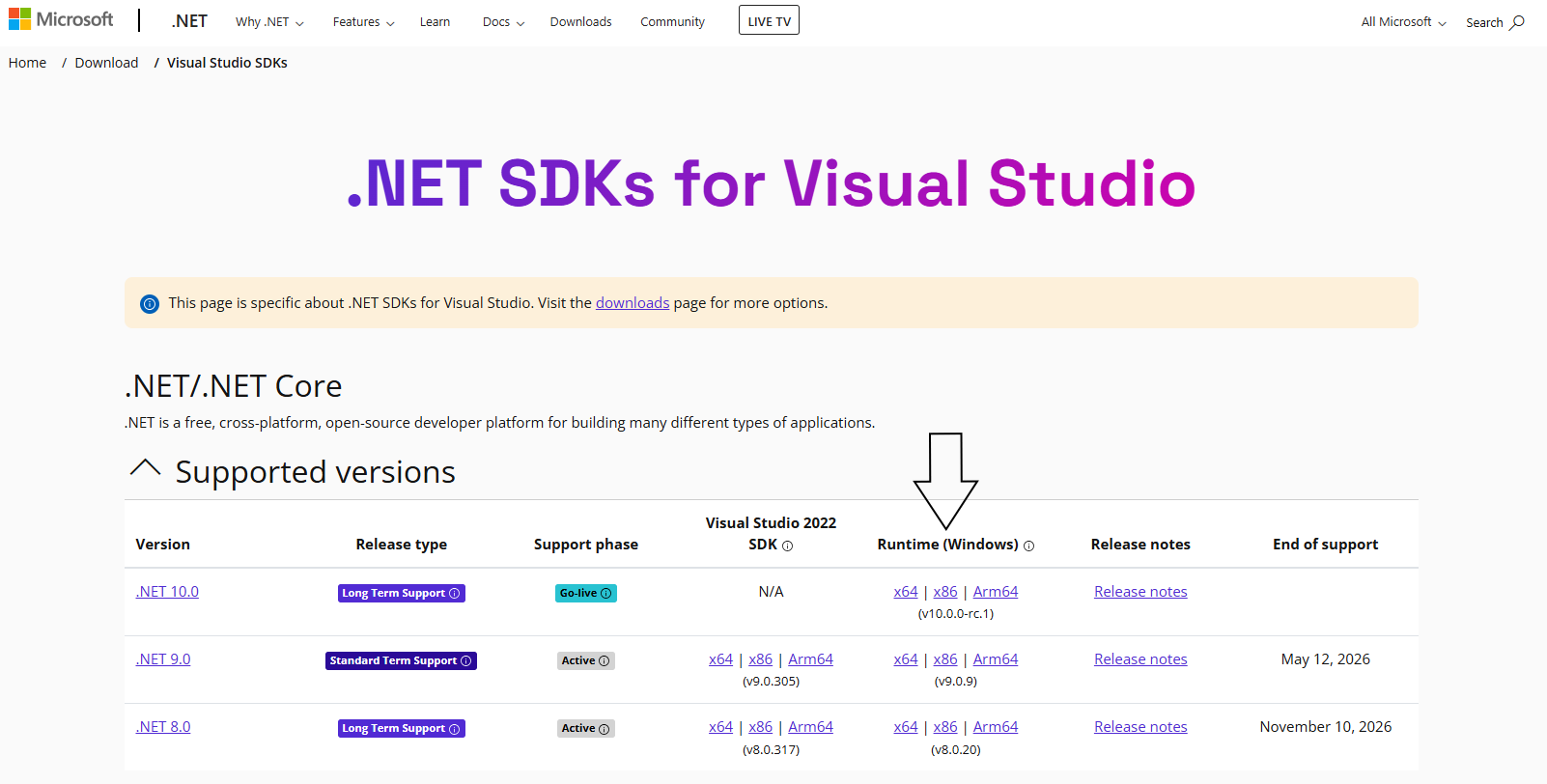
**Chương 3: Cài đặt môi trường và “Hello World”**

1. **Lý thuyết:**

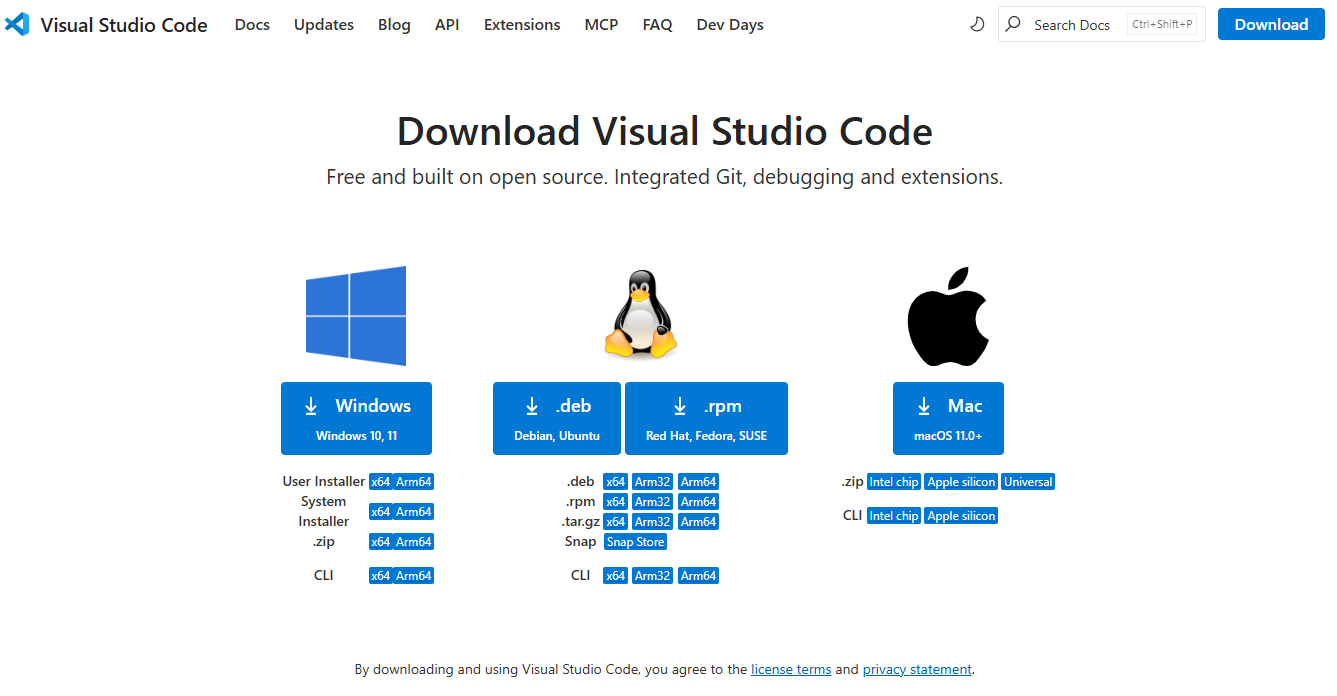
* Các công cụ cần thiết để bắt đầu:
  + .NET SDK
  + Visual Studio Code.
    1. Sau khi cài xong Visual Studio Code, ta sẽ tải các extension cần thiết là: C# Extension, C# Dev Kit và .NET Runtime Install Tool.

1. **Hướng dẫn:**

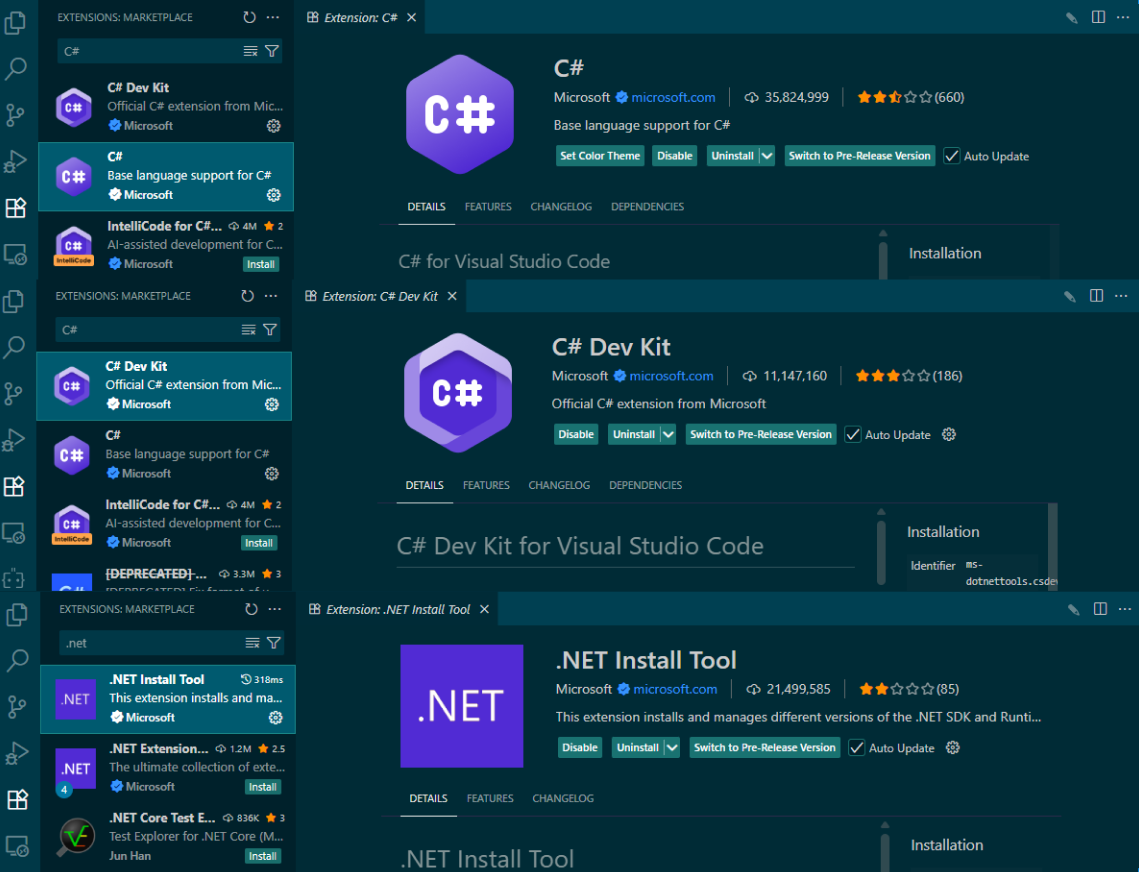
* Bước 1: Tải và cài đặt .NET SDK ở trang chủ chính thức của Microsoft: <https://dotnet.microsoft.com/en-us/download/visual-studio-sdks>, tải bộ cài đúng theo phiên bản hệ điều hành của mình.



* Bước 2: Tải và cài đặt Visual Studio Code ở trang chủ chính thức của Microsoft: <https://code.visualstudio.com/Download>, chọn đúng phiên bản theo hệ điều hành của mình.



* Bước 3: Tải các extensions C# Extension, C# Dev Kit và .NET Runtime Install Tool.



1. **Thực hành:**

Các bước chạy dự án Hello World ASP.NET đầu tiên như sau:

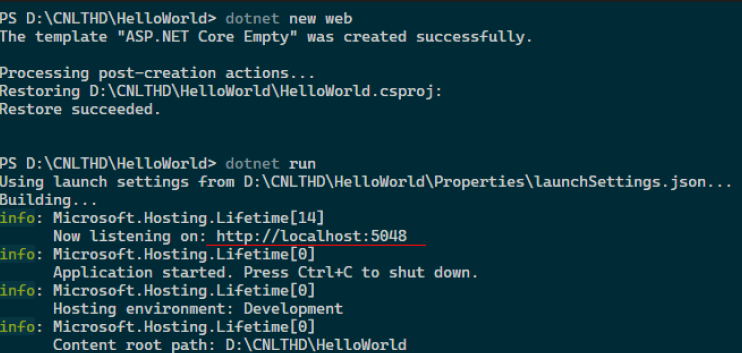
* Chạy các mã lệnh sau ở Terminal:

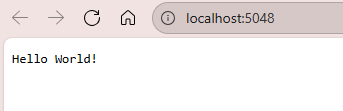
mkdir HelloAspNet && cd HelloAspNet

dotnet new web

dotnet run

* Nếu làm đúng các bước, Visual Studio Code sẽ thông báo building thành công và đưa lại đường dẫn mà khi mở ta có thể thấy được “Hello World!”

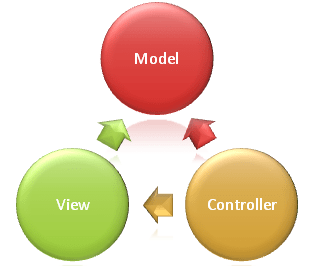




* Phân tích cấu trúc thư mục dự án vừa tạo:
  + bin/ và obj/ là nơi chứa các file chạy sau khi build và nơi chứa file tạm và metadata trong quá trình build.
  + Program.cs: Entry point, định nghĩa route, middleware, app.Run().
  + .csproj: Khai báo framework (.NET 9.0), package, tùy chọn build.
  + appsettings.json: Cấu hình mặc định (logging, connection string, host).
  + appsettings.Development.json: Ghi đè khi môi trường Development.
  + Properties/launchSettings.json: Cấu hình debug (port, env, auto mở browser).

**Chương 4: Các khái niệm cơ bản**

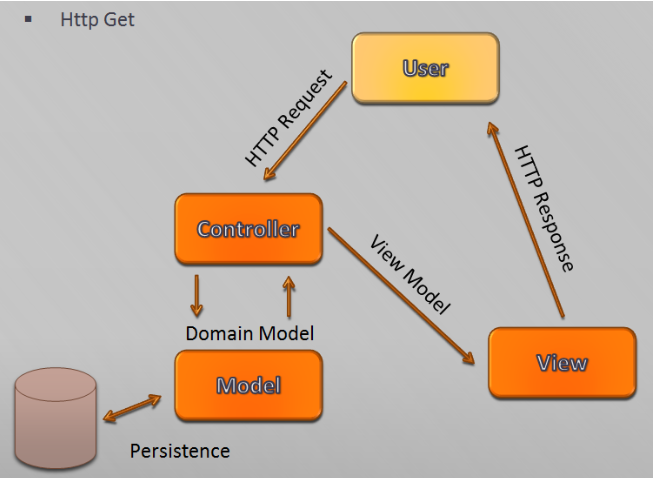
1. **Kiến trúc và mô hình lập trình.**
2. **Cấu trúc và ứng dụng ASP.NET**
3. **Routing cơ bản.**
4. **MVC Pattern.**
5. **Khái niệm cơ bản về MVC Pattern trong ASP.NET**

****

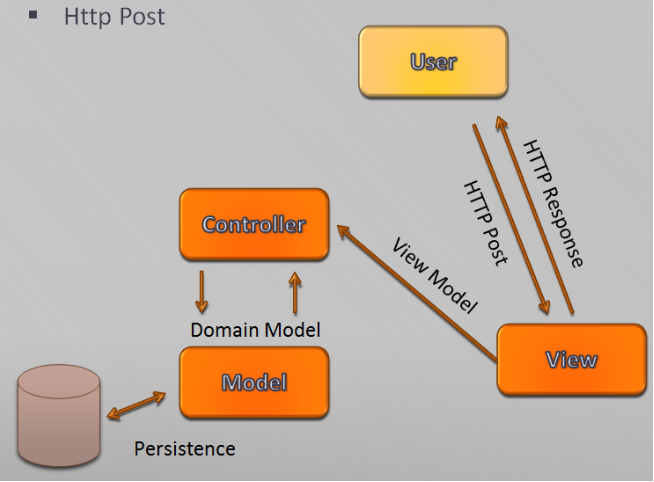
* **MVC (Model – View – Controller)** là một kiến trúc phần mềm được ASP.NET hỗ trợ để phát triển các ứng dụng web.
* **Mục tiêu**: tách biệt logic xử lý, giao diện, và luồng điều khiển, giúp code dễ bảo trì, mở rộng, và tái sử dụng.
* Ba thành phần chính:

1. **Model:** quản lý dữ liệu, logic nghiệp vụ, giao tiếp database.
2. **View:** hiển thị giao diện, render dữ liệu ra HTML.
3. **Controller:** trung gian điều khiển, nhận request từ client, gọi Model, đẩy dữ liệu ra View.

* Trong ASP.NET, Controller nhận HTTP Request từ User. Tùy theo loại request:
* HTTP GET: lấy dữ liệu và hiển thị



* HTTP POST: gửi dữ liệu từ client để lưu/ xử lý trên server.



1. **Cách dữ liệu di chuyển** (minh họa)

**Luồng tổng quát của MVC:**

User Request → Controller → Model → Controller → View → User UI

**Luồng khi người dùng chỉ lấy dữ liệu:**

User (HTTP GET) → Controller → Model → Controller → View →

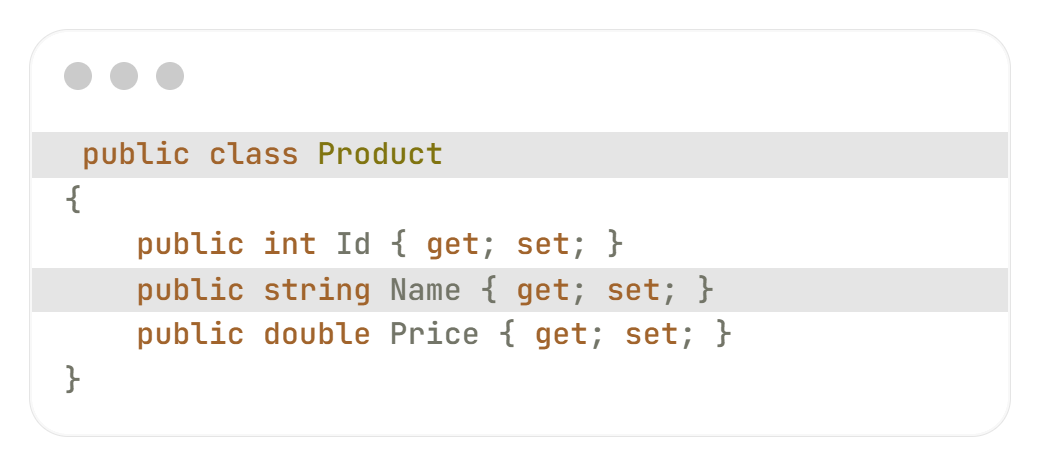
HTTP Response → User

**Luồng khi người dùng gửi dữ liệu để lưu vào database:**

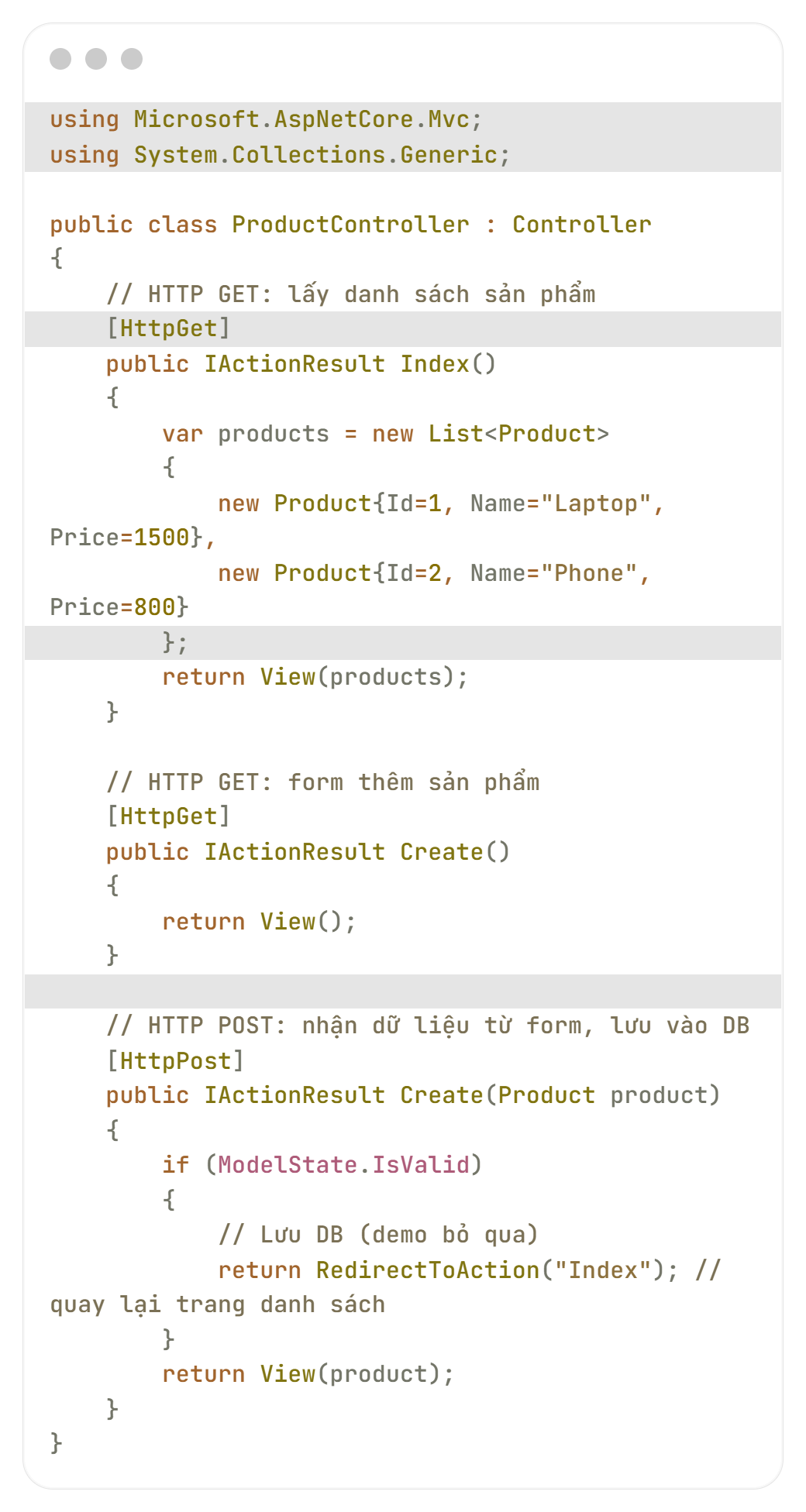
User (HTTP POST form data) → Controller → Model (lưu DB) → Controller → View/Redirect → Response → User

1. **Minh họa mã nguồn áp dụng mô hình MVC**

**Model** (Product.cs)



**Controller** (ProductController.cs)

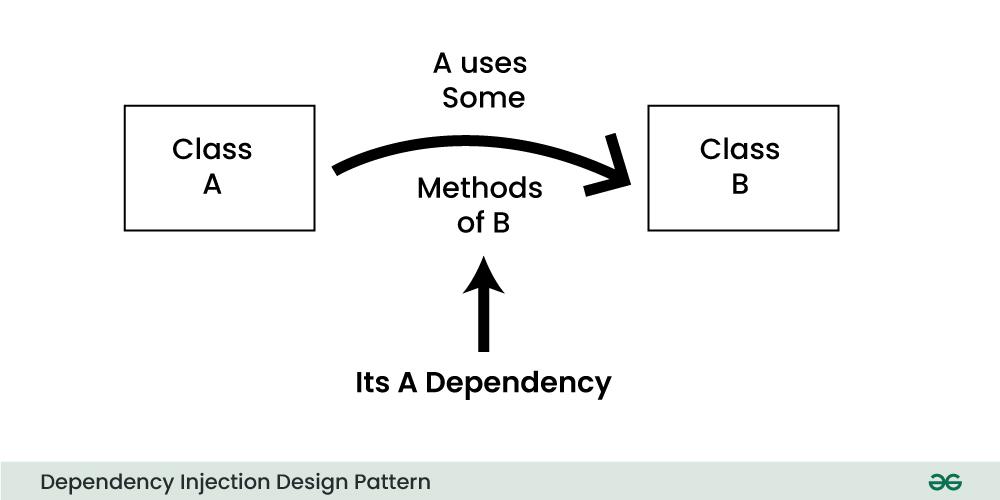


**View** (Create.cshtml)



1. **Razor và View Engine.**
2. **Dependency Injection cơ bản trong ASP.NET**
   1. **Dependency Injection là gì?**

Trong lập trình, Dependency Injection, thường được dịch thô là “tiêm phụ thuộc/chèn phụ thuộc”, là một kỹ thuật theo đó một đối tượng (hoặc static method) cung cấp các phụ thuộc của đối tượng khác. Một phụ thuộc là một đối tượng có thể được sử dụng (service).



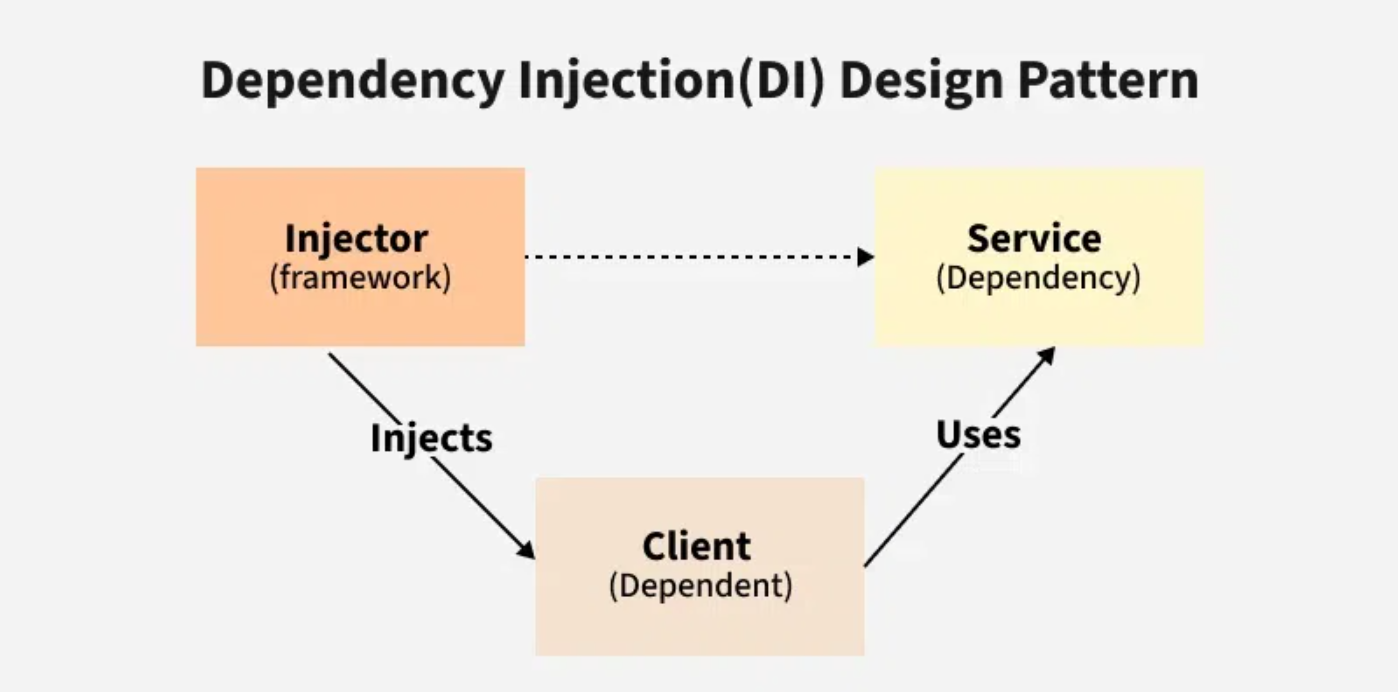
Ví dụ tạo trực tiếp:



Ví dụ tạo với DI (Dependency Injection)



* 1. **Vai trò của Dependency Injection**
  + Giảm coupling: Controller sẽ chỉ phụ thuộc vào interface, không phụ thuộc class cụ thể.
  + Dễ bảo trì: có thể thay thế TaskService bằng MockTaskService khi test.
  + Quản lý vòng đời: ASP.NET Core quản lý tạo/huỷ đối tượng, tối ưu bộ nhớ.
  + Test dễ hơn: cho phép viết unit test độc lập mà không cần kết nối DB thật.
  1. 4 vai trò trong Dependency Injection
  + **Client:**  
    Là thành phần hoặc lớp phụ thuộc vào các dịch vụ do lớp/ module khác cung cấp.  
    Client không tự tạo ra dependency mà chỉ nhận chúng từ **Injector**. (Injector chịu trách nhiệm cung cấp dependency, không phải Client).
  + **Service:**  
    Là thành phần hoặc lớp cung cấp một chức năng hoặc dịch vụ cụ thể mà Client cần.  
    Tập trung vào việc cung cấp chức năng riêng, được thiết kế độc lập với Client.
  + **Injector:**  
    Chịu trách nhiệm tạo các instance của service và “tiêm” (inject) chúng vào Client.  
    Nó biết Client cần dependency gì và cung cấp chúng ở runtime.
  + **Interface:**  
    Định nghĩa contract hoặc tập các phương thức mà service phải triển khai.  
    Client phụ thuộc vào interface thay vì phụ thuộc trực tiếp vào implementation, giúp linh hoạt và dễ thay thế implementation khi cần.



* 1. Các service trong container.

Thường có 3 scope phổ biến:

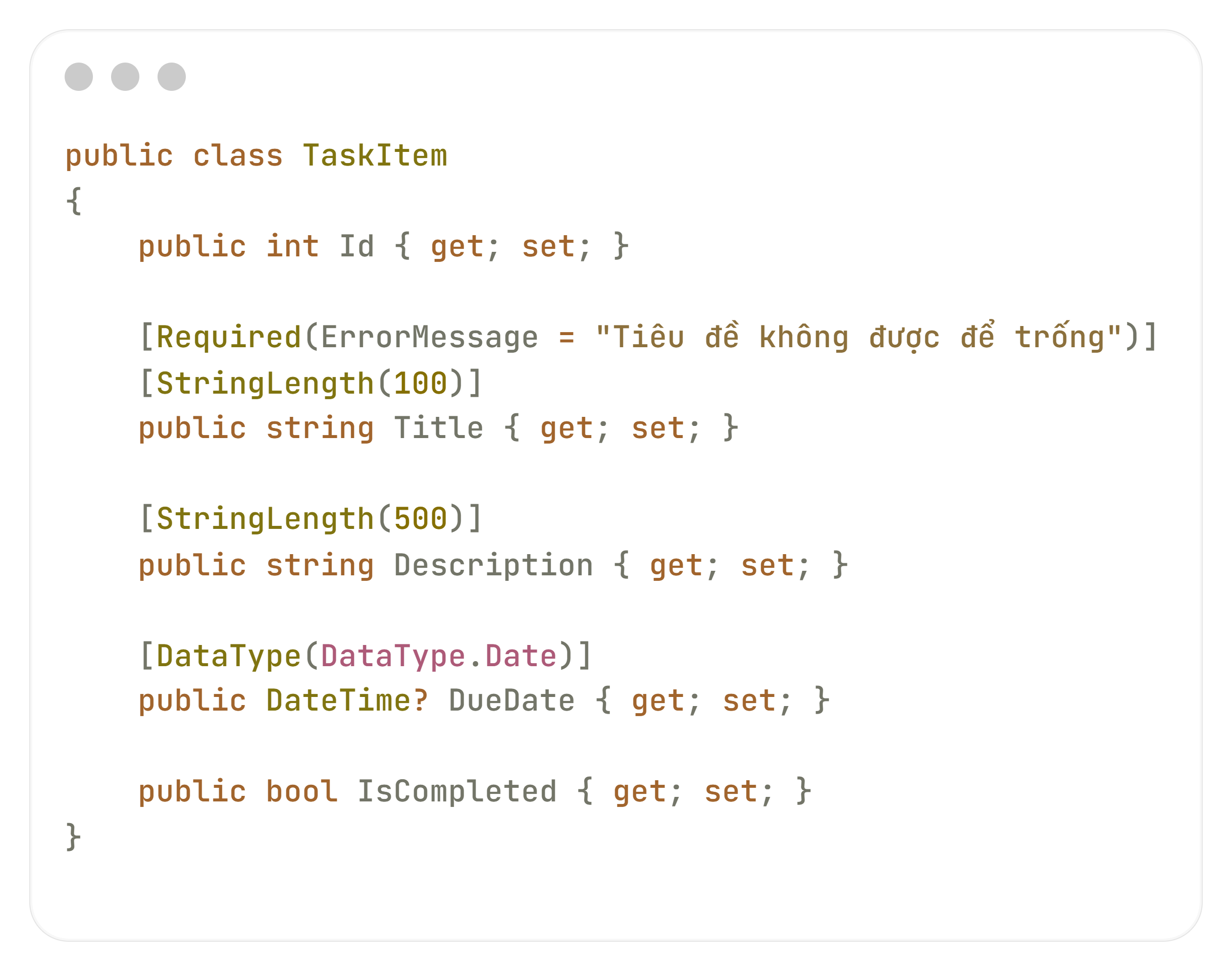
* + AddSingleton: Một instance duy nhất cho toàn ứng dụng, tái sử dụng cho mọi request.
  + AddScoped: Một instance cho mỗi request HTTP (thường dùng cho repository/service)
  + AddTransient: Tạo mới instance mỗi lần được yêu cầu (thích hợp cho service không giữ state).

Chọn đúng scope cho service sẽ giúp tối ưu bộ nhớ và tránh lỗi liên quan đến chia sẻ dữ liệu cho nhau.

1. **Làm việc với model trong ASP.NET Core**
   1. **Model là gì?**

Model thường là class đơn giản (POCO – Plain Old CLR Object), ánh xạ xuống bảng trong database.

Ví dụ:



* + **[Required]**: bắt buộc nhập.
  + **[StringLength]**: giới hạn độ dài.
  + **[Display]**: gán tên hiển thị trong view.
  + **[DataType]**: chỉ định kiểu hiển thị (date, email, password...).

Điểm cần lưu ý:

* + Các thuộc tính (property) là các cột trong bảng.
  + Sử dụng Data Annotation để ràng buộc dữ liệu (validation).
  + Model nên giữ logic nghiệp vụ cơ bản (ví dụ tính toán trạng thái task quá hạn).