|  |  |
| --- | --- |
| **ASP.NET Web App MVC** | là một mô hình phát triển ứng dụng web trong ASP.NET dựa trên kiến trúc Model-View-Controller (MVC). Đây là một cách tiếp cận phổ biến để xây dựng các ứng dụng web có cấu trúc rõ ràng, dễ bảo trì và mở rộng.  MVC là một mô hình thiết kế phần mềm chia ứng dụng thành 3 thành phần chính:   * Model: Chứa dữ liệu và logic xử lý dữ liệu (thường kết nối với cơ sở dữ liệu). * View: Chịu trách nhiệm hiển thị giao diện người dùng. * Controller: Xử lý yêu cầu từ người dùng, cập nhật Model và chọn View phù hợp để hiển thị.   Khi người dùng gửi một yêu cầu HTTP (ví dụ: truy cập một trang web), luồng xử lý trong ASP.NET MVC như sau:   * Người dùng gửi yêu cầu ➝ Trình duyệt gửi HTTP Request. * Bộ định tuyến (Routing) xác định Controller phù hợp dựa trên URL. * Controller xử lý logic và gọi Model để lấy dữ liệu (nếu cần). * Model truy xuất dữ liệu từ cơ sở dữ liệu và trả về Controller. * Controller chọn View phù hợp và truyền dữ liệu vào View. * View hiển thị dữ liệu và trả kết quả về trình duyệt. |
| **ADO.Net** |  |
| **Controller** | là một thành phần quan trọng trong mô hình **MVC (Model-View-Controller)** của ASP.NET. Nó chịu trách nhiệm xử lý các yêu cầu từ trình duyệt, tương tác với Model (dữ liệu), và chọn View (giao diện) để hiển thị nội dung cho người dùng.    🔹 Nhận yêu cầu từ trình duyệt (HTTP Request). 🔹 Xử lý logic nghiệp vụ, tương tác với Model để lấy hoặc cập nhật dữ liệu. 🔹 Chọn View phù hợp và trả kết quả về trình duyệt (HTTP Response).  Controller là **cầu nối giữa người dùng và hệ thống**, đảm bảo rằng yêu cầu được xử lý đúng cách. **Routing – Cách Controller xử lý yêu cầu** **Controller nhận request thông qua hệ thống Routing của ASP.NET MVC.** Mặc định, route có dạng:   **Action Method trong Controller** Một Controller có nhiều **Action Method**, mỗi phương thức tương ứng với một yêu cầu HTTP.      - Mỗi khi tạo controller thì lớp đó sẽ kế thừa controller của asp và mỗi phương thức sẽ là một hướng để |
| File Program.cs | trong ASP.NET Core MVC, Program.cs là điểm khởi động chính của ứng dụng. Nó chịu trách nhiệm cấu hình, khởi chạy web server, đăng ký các dịch vụ (Dependency Injection), và thiết lập pipeline xử lý HTTP requests. |
| **WebApplicationBuilder** | WebApplicationBuilder là một lớp trong ASP.NET Core được sử dụng để cấu hình và khởi tạo một ứng dụng web. Nó giúp bạn:  ✔ Cấu hình môi trường chạy ứng dụng (Development, Production, Staging). ✔ Đăng ký các dịch vụ (Dependency Injection - DI) như MVC, Entity Framework, Authentication. ✔ Thiết lập máy chủ web (Kestrel hoặc IIS). ✔ Đọc các cấu hình từ appsettings.json.  **Cách WebApplicationBuilder hoạt động**  📌 Khi chạy WebApplication.CreateBuilder(args), ASP.NET Core thực hiện các bước sau:  Tạo đối tượng WebApplicationBuilder.  Đọc cấu hình ứng dụng (appsettings.json, biến môi trường, args).  Khởi tạo hệ thống Dependency Injection (builder.Services).  Thiết lập máy chủ web (builder.WebHost).  Tạo ứng dụng (var app = builder.Build();).  **WebApplicationBuilder giúp cấu hình và khởi động ứng dụng ASP.NET Core.** ✔ Có 3 thành phần chính: builder.Configuration, builder.Services, builder.WebHost. ✔ Cho phép đăng ký dịch vụ, thiết lập máy chủ web, và quản lý cấu hình.  🚀 Nếu bạn làm việc với ASP.NET Core,  WebApplicationBuilder là phần quan trọng giúp ứng dụng hoạt động! 💡 |
| **CreateBuilder** | là một phương thức **tĩnh (static)** của lớp WebApplication trong **ASP.NET Core**, dùng để **tạo một đối tượng** WebApplicationBuilder.   CreateBuilder **làm những gì?** Khi gọi WebApplication.CreateBuilder(args);, ASP.NET Core thực hiện **các bước sau**: **1. Thiết lập cấu hình ứng dụng (**Configuration**)** ✔ Đọc cấu hình từ:   * appsettings.json * appsettings.{Environment}.json * Biến môi trường (Environment Variables) * Tham số dòng lệnh (Command Line Arguments)    **Đăng ký hệ thống Dependency Injection (**Services**)** ✔ Tạo một **DI Container** để quản lý các **dịch vụ (services)** như:   * **MVC (**AddControllersWithViews()**)** * **Entity Framework Core (**AddDbContext()**)** * **Authentication, Logging, Caching**    **3. Cấu hình máy chủ Web (**WebHost**)** ✔ CreateBuilder tự động thiết lập máy chủ web **Kestrel** hoặc **IIS**. ✔ Hỗ trợ **cấu hình Web Server** tùy chỉnh.   **2️⃣ Luồng hoạt động của** CreateBuilder 📌 **Khi gọi** WebApplication.CreateBuilder(args);**, ASP.NET Core thực hiện các bước sau**:   1. **Tạo đối tượng** WebApplicationBuilder. 2. **Đọc cấu hình ứng dụng (**appsettings.json**, biến môi trường, args)**. 3. **Khởi tạo hệ thống Dependency Injection (**builder.Services**)**. 4. **Thiết lập máy chủ web (**builder.WebHost**)**. 5. **Tạo ứng dụng (**var app = builder.Build();**)**. |
| builder.Services | là một **thuộc tính của** WebApplicationBuilder, cung cấp một **Dependency Injection (DI) container** trong ASP.NET Core. Nó được dùng để **đăng ký và quản lý các dịch vụ** trong ứng dụng.   **1️⃣** builder.Services **hoạt động như thế nào?** ASP.NET Core sử dụng **Dependency Injection (DI)** để quản lý các dịch vụ.   * **Dịch vụ** là các đối tượng được tạo một lần và có thể tái sử dụng. * Khi bạn **đăng ký một dịch vụ trong** builder.Services, ASP.NET Core sẽ tự động **cung cấp nó** cho các thành phần cần dùng.    **2️⃣ Các phương thức đăng ký dịch vụ** Có 3 cách chính để đăng ký dịch vụ trong builder.Services:   1. AddSingleton<T>() – Dịch vụ tồn tại **xuyên suốt vòng đời ứng dụng**. 2. AddScoped<T>() – Dịch vụ tồn tại **trong mỗi request HTTP**. 3. AddTransient<T>() – Dịch vụ được **tạo mới mỗi lần gọi**.     builder.Services **là nơi đăng ký dịch vụ (Dependency Injection) trong ASP.NET Core.** ✔ **Giúp quản lý Controller, Database, Authentication, Logging,...** ✔ **Có 3 loại dịch vụ chính:** Singleton**,** Scoped**,** Transient**.** ✔ **Là phần quan trọng giúp ứng dụng ASP.NET Core mở rộng linh hoạt và dễ bảo trì.**  🚀 **Tóm lại, nếu bạn làm việc với ASP.NET Core,** builder.Services **là nền tảng giúp ứng dụng chạy mượt mà!** 💡  Vì sao phải đăng ký Repository trong ASP.NET Core?  **Trong Java Servlet**, bạn có thể **tạo trực tiếp đối tượng Repository** bằng new. 🔹 **Trong ASP.NET Core**, bạn cần **đăng ký Repository vào** builder.Services **trước khi sử dụng**. |
| AddControllersWithViews | là một phương thức trong **ASP.NET Core**, dùng để **đăng ký MVC (Model-View-Controller)** vào hệ thống **Dependency Injection (DI)**.  🔹 **Chức năng chính:** ✔ **Hỗ trợ Controller để xử lý request** ✔ **Hỗ trợ View (**.cshtml**) để hiển thị giao diện** ✔ **Cho phép Routing để điều hướng URL** |
| Build() |  |
|  |  |