**1. ASP (Active Server Pages)**

ASP یک تکنولوژی ابتدایی از مایکروسافت بود که امکان ایجاد صفحات وب پویا را فراهم می‌کرد. این صفحات روی سرور اجرا می‌شدند و خروجی HTML برای مرورگر ارسال می‌شد. ASP با زبان‌هایی مثل VBScript کار می‌کرد و نقطه‌ی شروع ورود مایکروسافت به حوزه توسعه وب بود. با اینکه اکنون منسوخ شده، پایه‌گذار فریم‌ورک‌های جدیدتر مثل ASP.NET و ASP.NET Core محسوب می‌شود..

**2. .NET**

.NET یک بستر توسعه نرم‌افزار چندسکویی و چندزبانه است که توسط مایکروسافت توسعه یافته. این پلتفرم به توسعه‌دهندگان اجازه می‌دهد برنامه‌هایی برای وب، دسکتاپ، موبایل، بازی و کلود بسازند. .NET شامل زبان‌های برنامه‌نویسی، کتابخانه‌ها، runtime، ابزارها و فریم‌ورک‌های مختلفی مثل ASP.NET، WPF، WinForms و Blazor است. با معرفی .NET Core و سپس نسخه‌های متحد .NET 5 به بعد مایکروسافت این پلتفرم را به سمت یکپارچگی و بازمتن بودن برده است.

**3. ASP.NET Core**

ASP.NET Core نسخه‌ی بازنویسی‌شده و پیشرفته ASP.NET است که با معماری ماژولار و سبک، عملکرد بسیار بهتری دارد. این فریم‌ورک از ابتدا برای کراس‌پلتفرم بودن طراحی شده و روی ویندوز، لینوکس و مک اجرا می‌شود. از ویژگی‌های کلیدی آن می‌توان به پشتیبانی از تزریق وابستگی، میدلورهای قابل تنظیم، روتینگ منعطف، Razor Pages، Blazor، و پشتیبانی کامل از RESTful API اشاره کرد. این فریم‌ورک همچنین قابلیت توسعه‌ی میکروسرویس‌ها و کار با کانتینرها مانند Docker را دارد.

**4. Middleware**

میدلورها اجزایی هستند که درخواست‌های HTTP را قبل از رسیدن به مقصد نهایی مانند کنترلر یا endpoint دریافت و پردازش می‌کنند. این اجزا در pipeline اپلیکیشن قرار دارند و می‌توانند وظایفی مانند احراز هویت، گزارش‌گیری، فشرده‌سازی محتوا، مدیریت خطا و موارد امنیتی را انجام دهند. یکی از مزیت‌های ASP.NET Core این است که توسعه‌دهنده می‌تواند ترتیب و محتوای این pipeline را به صورت دقیق مدیریت یا سفارشی‌سازی کند.

**5. Web Routing**

روتینگ مکانیزمی است که مشخص می‌کند هر URL دریافتی از مرورگر باید به کدام قطعه کد در برنامه متصل شود. در ASP.NET Core، سیستم روتینگ انعطاف‌پذیر و قابل تنظیم است. مسیرها می‌توانند به کنترلرها، اکشن‌ها یا Razor Pages اشاره کنند و از پارامترهای URL، کوئری‌استرینگ یا شرط‌های سفارشی استفاده نمایند. از نسخه‌های جدیدتر نیز روتینگ endpoint-based جایگزین روتینگ سنتی شده و امکان تعریف مسیرها را به صورت واضح‌تر فراهم می‌کند.

**6. Data Annotation**

Data Annotationها ویژگی‌هایی (attributes) هستند که روی کلاس‌ها و اعضای آن‌ها اعمال می‌شوند تا اطلاعات اضافی درباره‌ی ساختار یا اعتبار داده‌ها را فراهم کنند. این ویژگی‌ها بیشتر برای اعتبارسنجی فرم‌ها یا تعیین نحوه نمایش داده‌ها در واسط کاربری به کار می‌روند. برای مثال، می‌توان مشخص کرد که یک فیلد اجباری است، یا حداکثر تعداد کاراکتر دارد. این روش اعتبارسنجی ساده و قابل ترکیب با سایر روش‌های اعتبارسنجی سمت سرور است.

**7. DbContext**

DbContext هسته مرکزی کار با پایگاه‌داده در Entity Framework Core است. این کلاس نقش پل ارتباطی بین داده‌های اپلیکیشن و دیتابیس را ایفا می‌کند. توسعه‌دهنده با استفاده از این کلاس می‌تواند داده‌ها را خوانده، درج کرده، به‌روزرسانی کند یا حذف نماید. همچنین از طریق آن، پیکربندی مدل داده‌ها، روابط بین جداول و تنظیمات migration انجام می‌شود. هر DbContext معمولاً شامل مجموعه‌ای از ویژگی‌های DbSet است که نماینده جدول‌ها در دیتابیس هستند.

**8. Connection String**

رشته اتصال یا Connection String اطلاعات مورد نیاز برای برقراری ارتباط با یک منبع داده را در قالب یک رشته متنی نگه‌می‌دارد. این اطلاعات شامل آدرس سرور، نام پایگاه داده، نوع احراز هویت و سایر پارامترهای لازم برای اتصال است. در ASP.NET Core، این رشته معمولاً در فایل پیکربندی پروژه مانند appsettings.json قرار دارد و هنگام راه‌اندازی سرویس‌ها یا دیتابیس خوانده می‌شود. مدیریت درست Connection String از نظر امنیتی و کارایی اهمیت زیادی دارد.

**9. Dependency Injection (DI)**

DI یک الگوی طراحی است که وابستگی‌های یک کلاس را از بیرون تأمین می‌کند، به جای اینکه کلاس خودش آن‌ها را بسازد. ASP.NET Core از ابتدا با سیستم DI داخلی طراحی شده است. این قابلیت به توسعه‌دهندگان کمک می‌کند کدهایی بنویسند که قابل تست، ماژولار و کم‌وابستگی باشند. از DI برای تزریق سرویس‌ها، ریپوزیتوری‌ها، کانفیگ‌ها و کلاس‌های کمکی به کنترلرها یا سایر اجزای اپلیکیشن استفاده می‌شود.

**10. Controller**

کنترلرها بخشی از معماری MVC هستند و مسئول دریافت درخواست‌ها از سمت کاربر و پاسخ به آن‌ها می‌باشند. در ASP.NET Core، هر کنترلر یک کلاس است که شامل متدهایی به نام اکشن است. کنترلر منطق پردازش درخواست را اجرا کرده و نتیجه‌ای مثل یک View، JSON، یا Redirect بازمی‌گرداند. کنترلرها نقش اصلی در هماهنگی بین لایه‌های مختلف برنامه مثل مدل، سرویس‌ها و ویو دارند.

**11. ORM (Object-Relational Mapping)**

ORM یک رویکرد برای تعامل با پایگاه‌داده است که اجازه می‌دهد توسعه‌دهنده به جای نوشتن کد SQL، با اشیاء برنامه‌نویسی کار کند. در این مدل، جداول دیتابیس به کلاس، و سطرها به نمونه‌ای از کلاس تبدیل می‌شوند. ORM عملیات‌هایی مانند درج، حذف، به‌روزرسانی و بازیابی داده‌ها را به صورت شی‌گرا انجام می‌دهد. این رویکرد توسعه را ساده‌تر، سریع‌تر و امن‌تر می‌کند، اما در برخی مواقع نیاز به بهینه‌سازی دستی دارد.

**12. Entity Framework**

Entity Framework (EF) یک ORM توسعه‌یافته توسط مایکروسافت است که امکان تعامل با پایگاه‌داده را بدون نوشتن SQL مستقیم فراهم می‌کند. EF Core نسخه‌ی بازمتن و سبک‌تر آن است که مناسب توسعه در ASP.NET Core می‌باشد. این ابزار از migration برای مدیریت تغییرات ساختار پایگاه‌داده، queryهای LINQ برای بازیابی داده و مکانیزم ردیابی تغییرات برای ذخیره‌سازی استفاده می‌کند.

**13. Razor**

Razor یک موتور ویوی سبک و کارآمد است که به توسعه‌دهندگان اجازه می‌دهد کدهای C# را در درون HTML بنویسند. این تکنولوژی در فایل‌هایی با پسوند cshtmlاستفاده می‌شود و خوانایی کد را نسبت به روش‌های قدیمی‌تر افزایش داده است. Razor نقش مهمی در طراحی UI در ASP.NET Core دارد، به‌ویژه در ساخت صفحات داینامیک و پاسخ‌گو.

**14. Validation**

اعتبارسنجی فرآیندی است برای بررسی اینکه داده‌های ورودی کاربران مطابق با قواعد مورد انتظار باشند. در ASP.NET Core، اعتبارسنجی می‌تواند به‌صورت سمت سرور (از طریق Data Annotations یا Fluent Validation) یا سمت کلاینت (از طریق JavaScript) انجام شود. هدف از این کار جلوگیری از ورود داده‌های مخرب یا نامعتبر و حفظ امنیت و کیفیت اطلاعات در سیستم است.