مؤسسه فرهنگی هنری
دیباگران تهران

آموزش
ASP.NET Core

روزبه امیرعصامی

به نام خدا

آموزش

ASP.NET Core

مؤلف:

روزبه امیر عصامی

مؤسسه فرهنگی هنری
دیباگران تهران

عنوان کتاب: آموزش ASP.NET Core

مولف : روزبه امیر عصامی

ناشر: موسسه فرهنگی هنری دیباگران تهران

ویراستار: مهدیه مخبری

صفحه آرایی: نازنین نصیری

طراح جلد:داریوش فرسایی

نوبت چاپ: اول

تاریخ نشر: ۱۴۰۳

چاپ و صحافی:درج عقیق

تیراژ:۱۰۰ جلد

شابک: ۰-۸۶۱-۲۱۸-۶۲۲-۹۷۸

نشانی واحد فروش:تهران، خیابان انقلاب،خیابان دانشگاه -تقاطع شهدای ژاندارمری-پلاک ۱۵۸ساختمان دانشگاه- طبقه دوم-واحد۴ تلفن ها: ۶۶۹۶۵۷۴۹-۲۲۰۸۵۱۱۱

فروشگاههای اینترنتی دیباگران تهران :

WWW.MFTBOOK.IR

www.dibagarantehran.com

سرشناسه:امیرعصامی،روزبه،۱۳۶۱-

عنوان و نام پدیدآور:آموزش ASP.NET Core /مولف: روزبه امیرعصامی؛

ویراستار:مهدیه مخبری.

مشخصات نشر: تهران : دیباگران تهران :۱۴۰۳

مشخصات ظاهری:۱۹۶ ص:مصور،

شابک: ۰-۸۶۱-۲۱۸-۶۲۲-۹۷۸

وضعیت فهرست نویسی: فیپا

موضوع:مایکروسافت دات نت فریم ورک

موضوع:Microsoft .NET Framework

موضوع:وبگاه ها-برنامه های تالیفی

موضوع: Web sites-authoring programs

موضوع:علوم کامپیوتر computer science

موضوع:نرم افزار-مهندسی software engineering

رده بندی کنگره:۷۶/۷۶ QA

رده بندی دیویی:۰۰۵/۲۷۶

شماره کتابشناسی ملی:۹۶۹۷۴۸۳

نشانی تلگرام:@mftbook **نشانی اینستاگرام دیبا dibagaran_publishing**

فهرست مطالب

فصل سوم

فصل چهارم

فصل پنجم

فصل ششم

فصل هفتم

فصل هشتم

فصل نهم

مقدمه ناشر

خط مشی انتشارات مؤسسه فرهنگی هنری دیباگران تهران در عرصه کتاب هایی با کیفیت عالی است که بتواند خواسته های به روز جامعه فرهنگی و علمی کشور را تا حد امکان پوشش دهد.

هر کتاب دیباگران تهران، یک فرصت جدید شغلی و علمی

حمد و سپاس ایزد منان را که با الطاف بیکران خود این توفیق را به ما ارزانی داشت تا بتوانیم در راه ارتقای دانش عمومی و فرهنگی این مرز و بوم در زمینه چاپ و نشر کتب علمی و آموزشی گام‌هایی هرچند کوچک برداشته و در انجام رسالتی که بر عهده داریم، مؤثر واقع شویم.

گستردگی علوم وسرعت توسعه روزافزون آن، شرایطی را به وجود آورده که هر روز شاهد تحولات اساسی چشمگیری در سطح جهان هستیم. این گسترش و توسعه، نیاز به منابع مختلف از جمله کتاب را به عنوان قدیمی‌ترین و راحت‌ترین راه دستیابی به اطلاعات و اطلاع‌رسانی، بیش از پیش برجسته نموده است.

در این راستا، واحد انتشارات مؤسسه فرهنگی هنری دیباگران تهران با همکاری اساتید، مؤلفان، مترجمان، متخصصان، پژوهشگران و محققان در زمینه های گوناگون و مورد نیاز جامعه تلاش نموده برای رفع کمبودها و نیازهای موجود، منابعی پُربار، معتبر و با کیفیت مناسب در اختیار علاقمندان قرار دهد.

کتابی که در دست دارید تألیف **"جناب آقای روزبه امیرعصامی"** است که با تلاش همکاران ما در نشر دیباگران تهران منتشرگشته و شایسته است از یکایک این گرامیان تشکر و قدردانی کنیم.

با نظرات خود مشوق و راهنمای ما باشید

با ارائه نظرات و پیشنهادات وخواسته های خود،به ما کمک کنید تا بهتر و دقیق تر در جهت رفع نیازهای علمی و آموزشی کشورمان قدم برداریم.برای رساندن پیام هایتان به ما از رسانه های دیباگران تهران شامل سایتهای فروشگاهی و صفحه اینستاگرام و شماره های تماس که در صفحه شناسنامه کتاب آمده استفاده نمایید.

مدیر انتشارات

مؤسسه فرهنگی هنری دیباگران تهران

dibagaran@mftplus.com

پیش‌نیازها

برای مطالعه این کتاب شما باید پیش‌نیازهای زیر را داشته باشید:

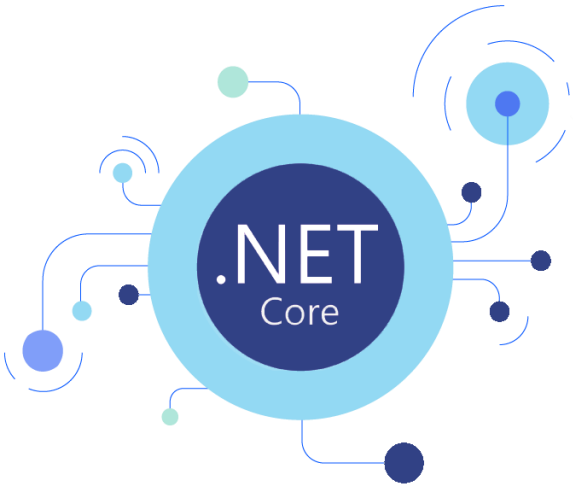
۱) دانستن اینکه وب اپلیکیشن چیست؟
۲) داشتن یک تجربه اولیه از ایجاد یک وب اپلیکیشن با سی شارپ.

ابزارهای مورد نیاز

تنها ابزاری که برای یادگیری به آن نیازی دارید Visual Studio 2019 است.

نکته

- هنگام نصب Visual Studio، اطمینان حاصل کنید که کامپوننت‌ها، ASP.NET و .NET Core را انتخاب کرده‌اید.



فصل اول

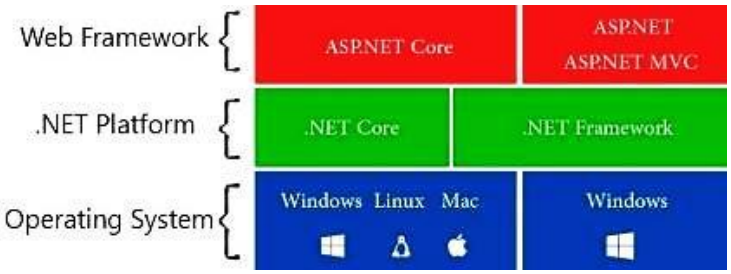
چرا ASP.NET Core؟

- ASP.NET Core چیست؟
- چرا ASP.NET Core را انتخاب کنیم؟
- ایجاد اولین اپلیکیشن ASP.NET Core
- اجرای وب اپلیکیشن
- درک ساختار پروژه

ASP.NET Core چیست؟

ASP.NET Core با اصول طراحی نرم‌افزار مدرن، بر روی پلتفرم جدید .NET Core ایجاد شده است که در آن فقط کامپوننت‌هایی که نیاز دارید را استفاده می‌کنید و می‌توانید برنامه خود را تا جایی که امکان دارد جمع‌وجور و با عملکرد بالا بسازید.

ASP.NET Core این امکان را به شما می‌دهد که وب اپلیکیشن‌های خود را در ویندوز لینوکس و Mac ایجاد و اجرا نمایید.



چرا ASP.NET Core را انتخاب کنیم؟

ASP.NET Core یک وب فریم‌ورک قدرتمند است که از آن می‌توان برای ایجاد سریع‌تر، راحت‌تر و امن‌تر انواع اپلیکیشن‌ها استفاده نمود. این وب فریم‌ورک پر از ویژگی‌های جالب است که در اینجا چند نمونه از مهم‌ترین آنها بیان می‌شود:

- ASP.NET Core یک فریم‌ورک مدرن، Scalable، Open Source و با Performance بالا است.
- ASP.NET Core یک معماری ماژولار برای Maintenance راحت‌تر دارد.
- ASP.NET Core مدیریت Cross-Site Request Forgery (CSRF) را برعهده می‌گیرد و می‌تواند روی هر پلتفرمی اجرا شود.

اما این فریم‌ورک برخی پیشرفت‌های زیربنایی هم داشته که در ادامه چند نمونه ذکر شده است:

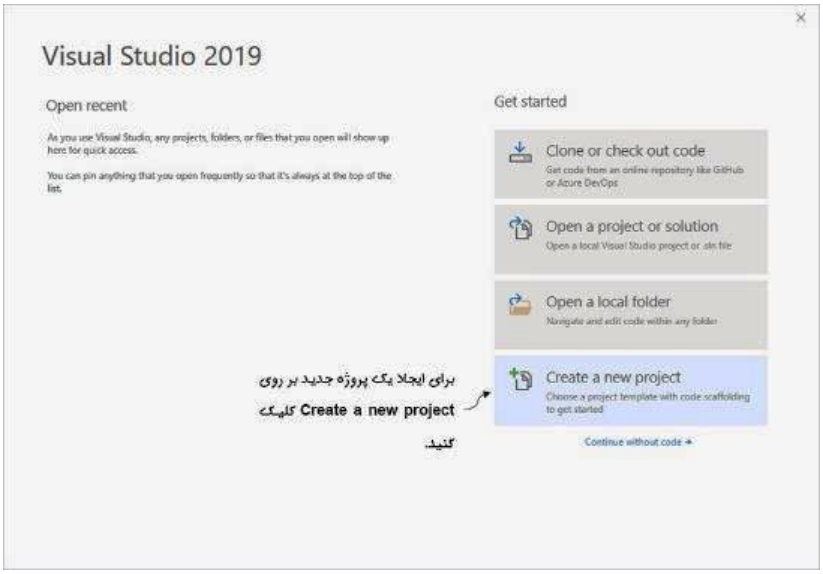
- Middleware Pipeline برای تعریف رفتارهای اپلیکیشن
- داشتن یک Dependency Injection تو کار
- ترکیب زیرساخت UI (MVC) و API (Web API)
- سیستم پیکربندی بسیار گسترده
- قابل Scalable بودن برای پلتفرم‌های Cloud (با استفاده از برنامه‌نویسی غیرهمزمان)

ایجاد اولین برنامه ASP.NET Core

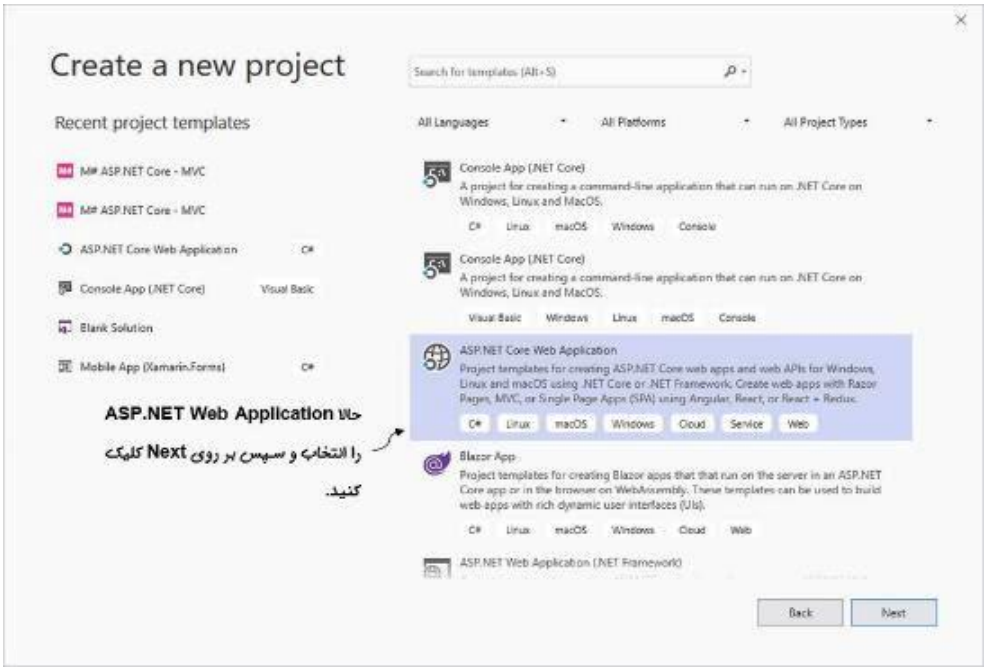
اولین وب اپلیکیشن خود را ایجاد کنید.

برای درک بهتر با ویژوال استودیو ۲۰۱۹ یک اپلیکیشن Empty از ASP.NET Core ایجاد کنید.

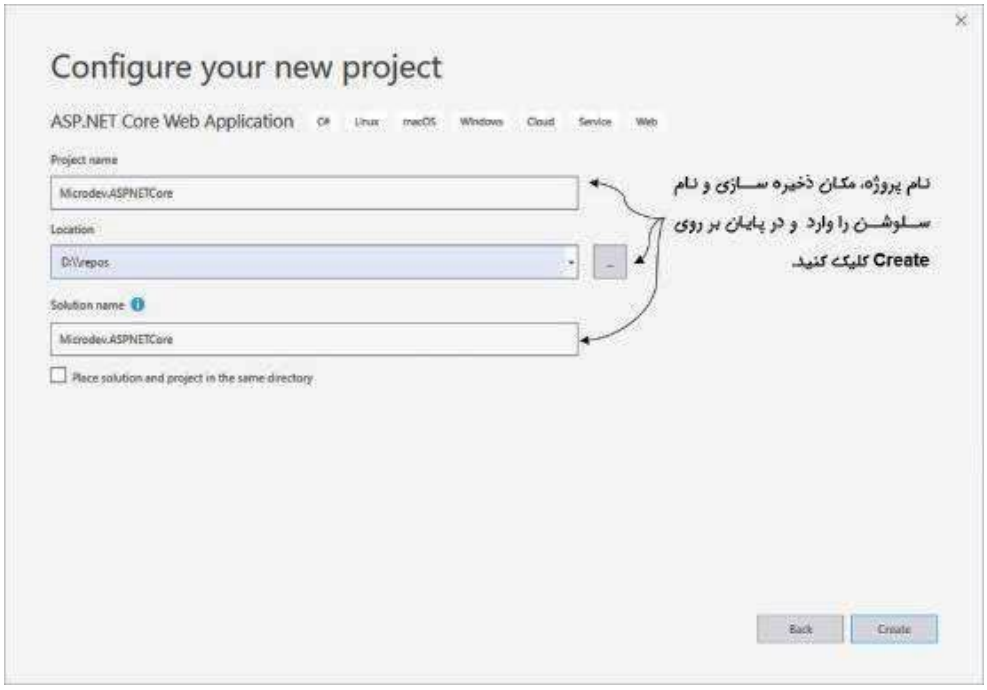
- ویژوال استودیو ۲۰۱۹ را باز کنید و بر روی Create a new project کلیک کنید.



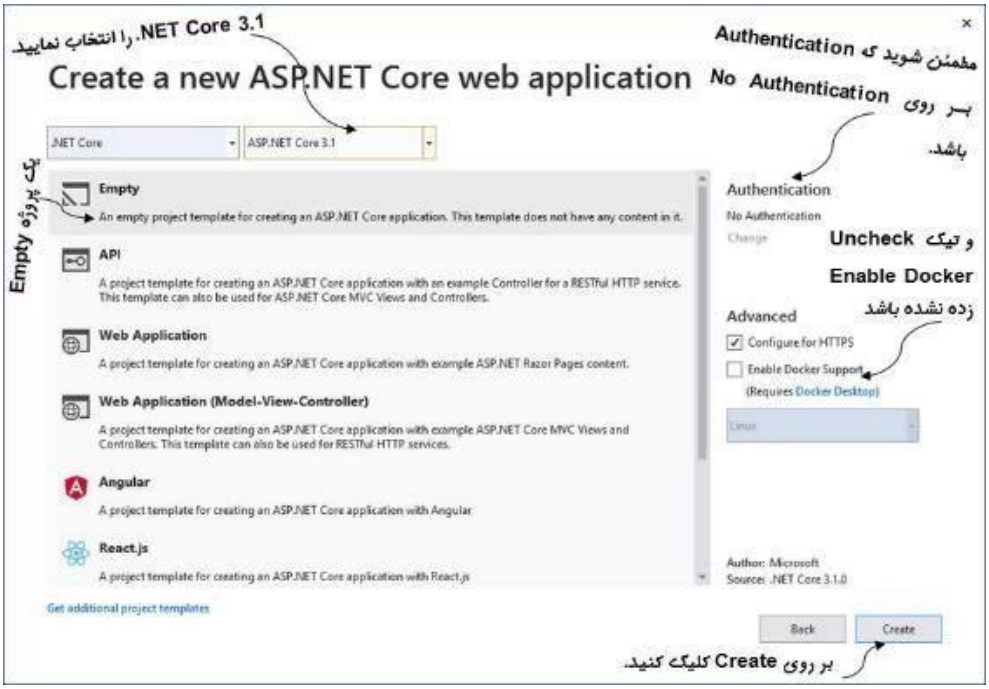
- در کادر بعدی ASP.NET Core Web Application را انتخاب و بر روی Next کلیک کنید.



- حالا نام پروژه را Microdev.ASPNETCore بگذارید، سپس مکان ذخیره‌سازی و نام Solution را همانند تصویر وارد و بر روی Create کلیک نمایید.

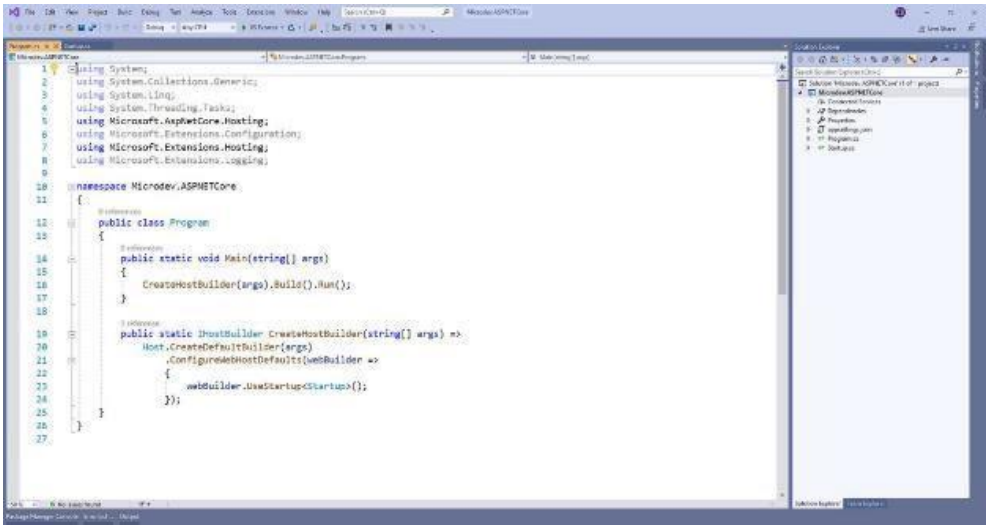


- در کادر بعدی Empty و Asp Net Core 3.1 را انتخاب کنید.



نکته

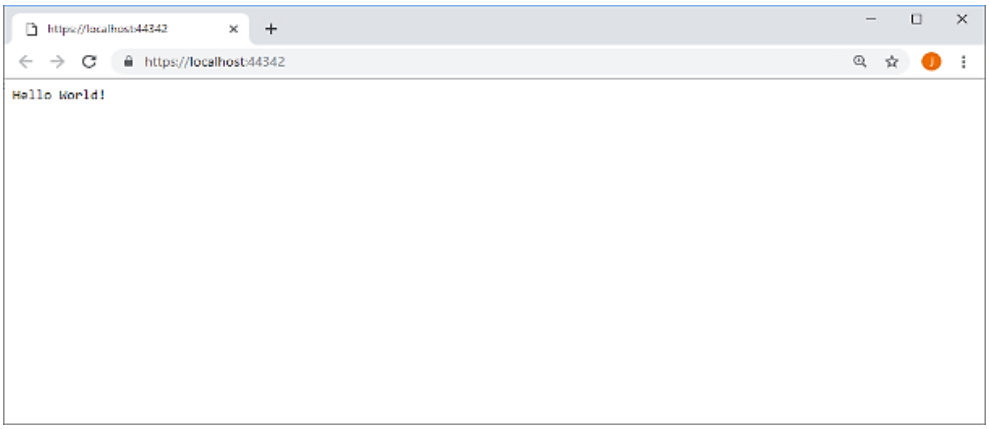
مطمئن شوید که Enable Docker Support تیک نخورده و اپلیکیشن بر روی No Authentication تنظیم شده است. منتظر بمانید تا Visual Studio اپلیکیشن شما را ایجاد نماید.



اولین وب اپلیکیشن ایجاد شد.

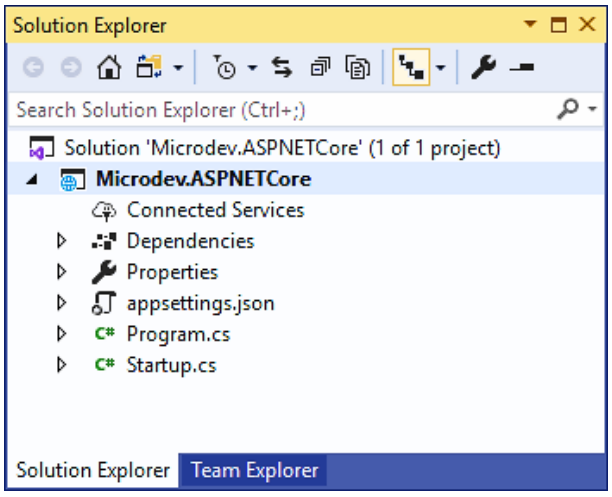
اجرای وب اپلیکیشن

برنامه را اجرا کنید. پروژه را Build و سپس F5 را بزنید. (یا اینکه بر روی پیکان سبز رنگ در نوار ابزار کنار IIS Express کلیک نمایید.)



درک ساختار پروژه

به عناصر درون Solution نگاه بیندازید:



- **Connected Services و Dependencies:** این دو قسمت برای نمایش تمام وابستگی‌های NuGet packages، وابستگی‌های Client-Side و سرویس‌های از را دور وابسته به پروژه می‌باشد.
- **Properties:** فولدر Properties تنها یک فایل launchSettings.json برای کنترل نحوهٔ اجرای Visual Studio و Debug اپلیکیشن دارد.
- **Program.cs و Startup.cs:** این دو فایل هم برای راه‌اندازی وب سرور و Pipeline استفاده می‌شود.

کلاس Program

کلاس Program مسئول پیکربندی بسیاری از زیرساخت‌های اپلیکیشن شماست و تمامی اپلیکیشن‌های ASP.NET Core با این فایل شروع می‌شوند.

```
public class Program
{
    public static void Main(string[] args)
        { CreateHostBuilder(args).Build().Run();
    }
public static IHostBuilder CreateHostBuilder(string[]args)
    =>Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
        webBuilder.UseStartup<Startup>();

        });
}
```

متد Main نقطه شروع برنامه است که در آن بسیاری از پیکربندی‌های برنامه انجام می‌شود. هنگامی‌که اپلیکیشن شما Start می‌شود، این متد باید یک Host را پیکربندی و اجرا کند که این Host وظیفه مدیریت Startup و Lifetime اپلیکیشن را به عهده دارد و حداقل باید پیکربندی سرور و Request processing Pipeline را انجام دهد.

شاید بپرسید Host چیست؟ برای کنترل شروع و پایان اپلیکیشن، مدیریت چرخه عمر اپلیکیشن و تمام منابع وابسته به آن (مثل Dependency injection، Logging، Configuration) پیاده‌سازی‌های IHostedService را درون یک object به نام Host کپسوله می‌کنیم.

پیکربندی، ساخت و اجرای Host معمولاً توسط کدی که در کلاس Program نوشته شده انجام می‌شود. ابتدا متد Main، متدی با نام CreateHostBuilder را جهت پیکربندی و ساخت شی Builder فراخوانی و در پایان متد Build و Run شی Builder را صدا می‌زند.

نکته

- اگر از Entity Framework Core استفاده می‌کنید، باید Signature متد CreateHostBuilder را تغییر ندهید، زیرا Entity Framework Core tools برای اینکه بتواند بدون اجرای اپلیکیشن، پیکربندی Host را انجام دهد، از متد CreateHostBuilder استفاده می‌کند.
- کلاس Startup که در متد جنریک UseStartup <> رفرنس داده شده، جایی است که پیکربندی سرویس‌ها و تعریف Middleware Pipeline انجام می‌شود.

کلاس Startup

اپلیکیشن‌های ASP.NET Core برای پیکربندی برخی رفتارهای اپلیکیشن از یک کلاس Startup استفاده می‌کنند. این کلاس از هیچ کلاس پایه‌ای ارث‌بری نمی‌کند و هیچ اینترفیسی را هم پیاده‌سازی نخواهد کرد و کار پیکربندی سرویس و تعیین Middleware Pipeline را انجام می‌دهد.

کلاس Startup برای بارگذاری و پیکربندی کامپوننت‌های اپلیکیشن، شامل دو متد ConfigureServices و Configure می‌باشد.

نکته

- HostBuilder ایجاد شده در کلاس Program ابتدا متد ConfigureServices و سپس Configure را صدا می‌زند، بنابراین تمام سرویس‌های رجیستر شده در متد ConfigureServices در متد Configure قابل دسترسی هستند.
- سرویس، اشاره به کلاسی دارد که بتواند یک Functionality را در اپلیکیشن فراهم کند.

متد ConfigureServices

متد ConfigureServices یک متد اختیاریست که رجیسترکردن سرویس‌های اپلیکیشن با استفاده از سیستم تزریق وابستگی ASP.NET Core (Dependency Injection) درون این متد انجام می‌شود. بنابراین این متد جایی است که تزریق وابستگی در آن پیکربندی می‌شود.

تزریق وابستگی (Dependency Injection) چیست؟

تزریق وابستگی یک تکنیک بسیار مهم و کاربردیست که باعث می‌شود Instance های یک کلاس در زمان اجرا، ایجاد شده و شما به‌راحتی کدهای Tastable و Loosely Coupled بنویسید.

برای مثال: فرض کنید شما یک کلاس EmpolyeeService و یک کنترلر EmpolyeeController دارید که این کنترلر در زمان اجرا نیاز به یک Instance ای از EmpolyeeService دارد.

رویکرد کلی این است که هر زمان که نیاز به یک سرویس داشتید از کلمه کلیدی new استفاده کنید و یک Instance از آن سرویس ایجاد نمایید. متأسفانه مشکل این رویکرد این است که کد شما به پیاده‌سازی خاصی وابسته (Tightly couple) می‌شود و نگهداری و تست اپلیکیشن سخت خواهد شد.

Tightly couple چیست؟

Tightly couple یعنی یک کلاس به پیاده‌سازی کلاس دیگر وابسته باشد. در این روش تغییر کلاس کمکی بدون دست زدن به کلاس اصلی غیرممکن خواهد بود. چون در کلاس اصلی مستقیماً به کلاس کمکی رفرنس داده شده است.

اما راه‌حل این مشکل: بعد از نوشتن سرویس، شما می‌توانید وابستگی‌های خود را اعلام کرده و اجازه دهید تا کلاسی دیگر این وابستگی‌های را برای شما فراهم کند. این تکنیک با نام تزریق وابستگی یا معکوس شدن کنترل IOC (Inversion Of Control)شناخته می‌شود.

در برنامه‌های ASP.NET Core، رجیستر شدن سرویس‌ها در متد ConfigureServices انجام می‌شود و هر زمان که از یک ویژگی جدید ASP.NET Core در اپلیکیشن‌تان استفاده کنید، باید به این متد برگردید و سرویس‌های مورد نیاز را اضافه نمایید.

```
public class Startup
{
  public void ConfigureServices(IServiceCollection services)
   {
   services.AddScoped<EmpolyeeService>();
   }
}
```

پارامتر IServiceCollection (در ورودی این متد) شامل لیستی از سرویس‌های مورد نیاز اپلیکیشن است.

◄ AddScoped چیست؟

AddScoped متدی است که طول عمر یک سرویس را مشخص می‌کند. این طول عمر بدین‌صورت است که با هر بار درخواست وب، یک Instance جدید از کلاس EmpolyeeService ایجاد خواهد شد.

نکته

- اگر فراموش کنید یک سرویس را رجیستر نمایید، در زمان اجرا یک InvalidOperationException دریافت خواهید کرد.

متد Configure

Configure متدی است که با استفاده از اکستنشن متدهای موجود بر روی IApplicationBuilder ماژول‌هایی را به Request Pipeline اضافه و رفتارهای اپلیکیشن را تعریف می‌نماید. به این ماژول‌ها Middleware گفته می‌شود.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}
                app.UseRouting();
                app.UseEndpoints(endpoints =>
```

```
                {
                    endpoints.MapGet("/", async context =>
                    {
await context.Response.WriteAsync("Hello World!");
 });
 });
}
```

همانطور که می‌بینید این متد معمولاً دو پارامتر می‌پذیرد:

IApplicationBuilder و IWebHostEnvironment

- پارامتر IApplicationBuilder، ترتیب اجرای Middleware ها را تعریف می‌کند. بنابراین Middleware تنها با این اینترفیس به Pipeline اپلیکیشن اضافه خواهد شد. برای مثال UseDeveloperExceptionPage()، اکستنشن متدی بر روی آرگومان IApplicationBuilder است.
- پارامتر IWebHostEnvironment شی است که اطلاعاتی در مورد محیط جاری را در خود نگه می‌دارد. بنابراین از آن می‌توان برای ارائه رفتارهای متفاوت در محیط‌های توسعه استفاده نمود.