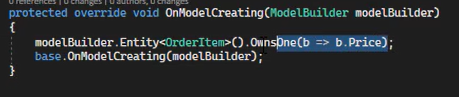
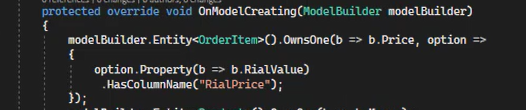
**Ownde Type**

در طراحی یک enitity یک property داریم که خود از نوع complex است یعنی یک کلاس است مثل address می توان تنظیمی در Ef انجام داد که این را به عنوان جدول جداگانه در نظر نگیرد و در دیتابیس با استفاده از یک convention مثل address\_ally نام filed ها را برای در جدول parent در نظر بگیرد

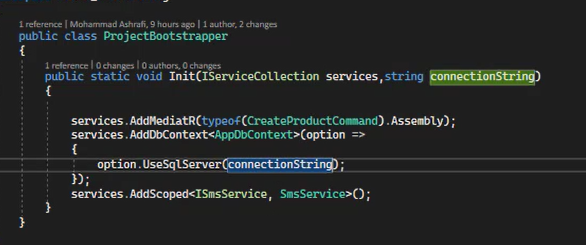


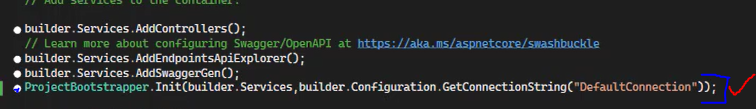
می توان به صورت کاستوم هم با استفاده از fluent api کاری کرد از نام را بر اساس convention قرار ندهد.



نکته:

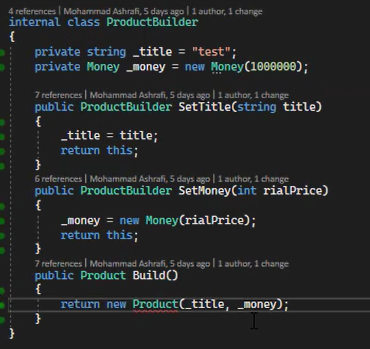
می توانیم در کلاس های جداگانه سرویس های خود را ریجستر کنیم کد تمیز تر شود و آن ها را به فایل program.cs اضافه کنیم.

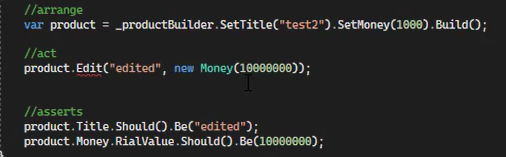




**Builder**

مثال استفاده از روش Builder برای ساخت یک Object محصل Product





**نکته unitTest**

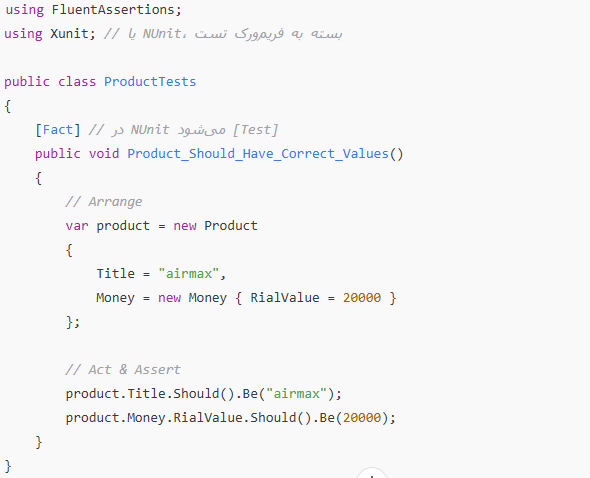
**FluentAssertions چیست؟**

FluentAssertions یک کتابخانه‌ی متن‌باز در .NET است که اجازه می‌دهد **عبارات تستی به شکل زبانی طبیعی** نوشته شوند. برخلاف روش‌های سنتی مقایسه با Assert.Equal(...) یا Assert.AreEqual(...)، این کتابخانه خوانایی کد تست را افزایش می‌دهد.

**نصب FluentAssertions**

برای استفاده از FluentAssertions، ابتدا باید آن را نصب کنید. اگر از **Visual Studio** یا **.NET CLI** استفاده می‌کنید

* خوانایی بالا
* پیام خطای توصیفی و قابل فهم هنگام شکست تست
* پشتیبانی از انواع مختلف object‌ها، collection‌ها، exceptions و async methods



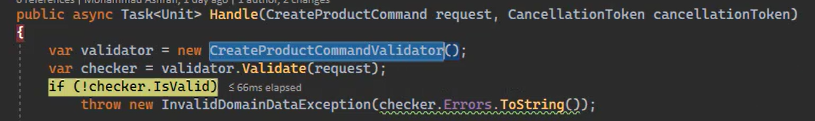
Nuget – Microsoft.Extensions.DependencyInjection

این بسته ابزاری برای پشتیبانی از di برای طراحی نرم افزار و رسیدن به ioc است و با استفاده از آن بدون نیاز به ثبت سرویس در di-container می تواند از آن نمونه بسازید و استفاده کنید.

راه حل دیگر استفاده از IServiceProvider است و ایجاد scope از سرویس.

Nuget – FluentValidation

یک پکیج است که با استفاده از آن می توانیم کار ولیدیشن مدل ها را با کلاس های جداگانه پیاده سازی کرده و استفاده کرده

****

**Pipe line behavior**

امکانی مانند middle wear ها در asp است که به ما این اجازه را می دهد که قبل و بعد request را مدیریت کنیم

با استفاده از mediatR

در کد زیر Next() به معنی اجرای command است و قبل و بعد آن نیز مشخص است.



نکته :

Pipeline behavior در di container باید register شود.

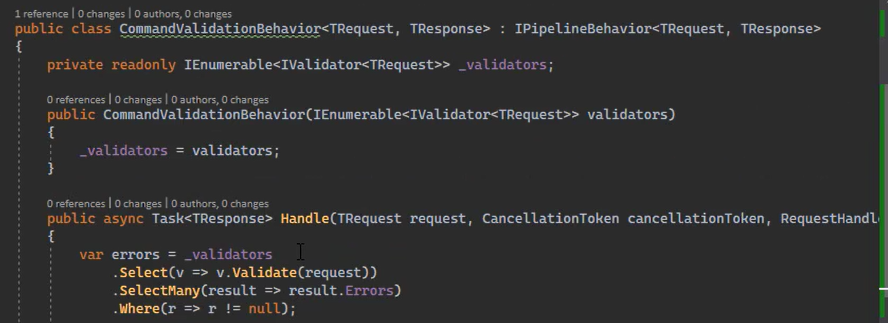
نکته:

Open Generic برای حالتی است که نوع مشخص نکنیم IpipelineBahavior<,>

Close Generic برای حالتی است که نوع مشخص کنیم CommandValidationBehavior<T,R>



مثال پیاده سازی شده در پروژه



به دلیل استفاده از fluentValidation در این روش لازم است که کلاس ولیدیشن خود را به di container معرفی کنیم و با استفاده از کتابخانه FluentValidaion.Dependency.injecion

RabbitMq

برای مدیریت ایونت ها و ارسال آن ها به صورت sync , async کاربرد دارد و همچنین با استفاده از MediatR نیز می توان publish و get مربوط به Event ها را مدیریت کرد