درس 14

در این درس ابتدا در کلاس program سیستم مسیریابی رو تنظیم می کنیم



قسمت هایی که با رنگ زرد مشخص شدن برای مسیر یابی ضروری هستند.

بعد از این کار یک کنترولر میسازیم و با دادن روت ان را اجرا می کنیم.



قسمتی که با رنگ زرد مشخص شده است روت است و با همین ادرس می توان به این کنترولر دست پیدا کرد.

درس 15 تنظیم مسیر یابی کنترولر ها

برای تنظیم مسیر یابی کنترولر ها در جلسته قبل روت رو اکشن بود که کارایی نداشت. در این جلسه روت رو کنترولر هست. به دو روش می توان گذاشت که با توجه به تجریه استاد مورد سبز بهتر است اما روش زرد رنگ هم درست است.



درس 16

معرفی و نصب Postman

درس 17 ساخت DTO

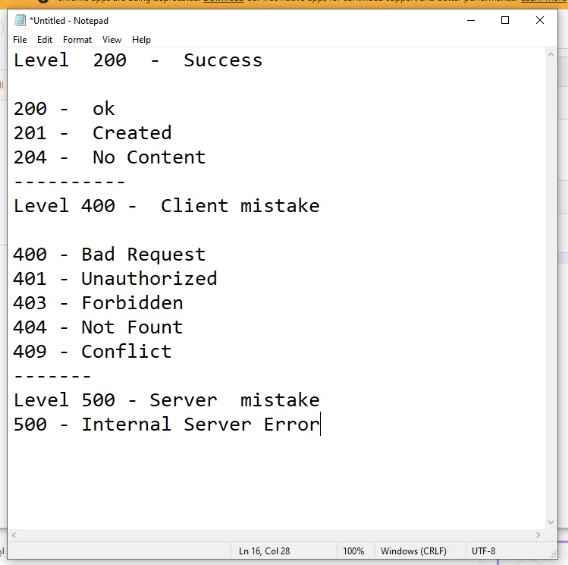
در این جلسه DTO ها رو ساختیم و مدل city بعد این کار موردی که جلسه پیش بود و اگر دو کنترلر با اندیس get داشتیم چطور این دو تا رو از هم متمایز کنیم را پاسخ دادیم.



باتوجه به قسمت زرد و سبز می توان بادادن اندیس ورودی روت این دو تایع Get ر ا از هم متمایز کنیم.

درس 18

کد های خطا در سمت سرور و کلاینت و موفقیت



درس 19

پیاده سازی کدها و کدها ی خطا و مشاهده و استفاده از IActionResult



درس 20

ایجاد مدل جدید و مقدار دهی به ان



و ایجاد ارتباط با مدل قبلی



قسمت رنگی یک پراپرتی تعریف شده که در همین سمت کلاینت مقدار دهی می شود و در دیتا بیس فیلد ندارد. فقط set دارد در سمت کلاینت محاسبه می شود و مقدار ان برگشت داده می شود. فقط readonly هست.

درس 21

در این قسمت برای pointofinterest کنترولر می نوسیم.



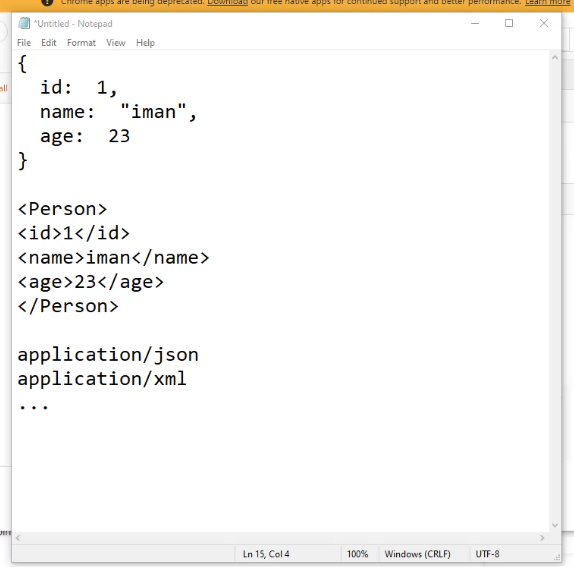
برای کل کنترولر از روت زرد رنگ استفاده می کنیم. با توجه به اینکه زیاد نمی خواهیم روت ایجاد کنیم روت را در ادامه Cities می نوسیم.

در قسمتی که سبز است روت ادامه روت زرد می شود.



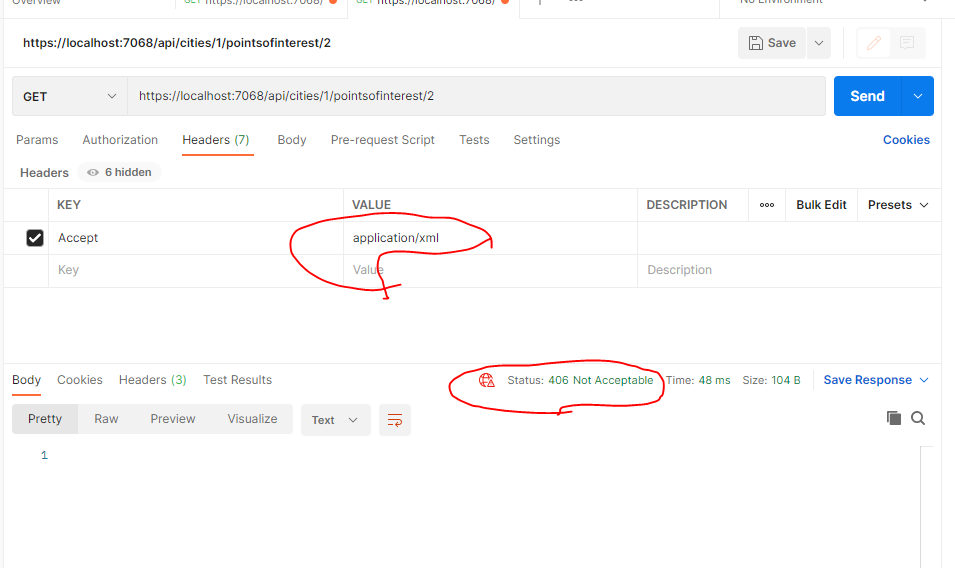
درس 22

در این جلسه در مورد فرمت وردی وخروجی صحبت شد. دو فرمت معروف JSON و XML بررسی شد. ورودی و خروجی باید فرمت مورد تایید خود را در هدر معرفی کنند.



درس 23

این جلسه ادامه جلسه قبل هست و تعیین فرمت وردی وخروجی بررسی می شود. این کار در قسمت Heder صورت می گیرد





با اضافه کردن کد بالا به Program زمانی که فرمت درخواست پشتیبانی نشود کد خطای 406 نمایش داده می شود.

برای پشتیبانی از فرمت XML از باید کد زیر را اضافه کنیم.



فرمت خروجی و وردی به صورت پیش فرض json است برای تغییر این پیش فرض از کدهای زیر استفاده می کنیم:



* 1. درس 24 دانلود فایل از طریق WebApi

برای دانلود فایل ابتدا یک کنترولر میسازیم :

اکشن دانلود را میسازیم از نوع GET هست. برای شناسایی فایل دانلودی به عنوان ورودی یک ایدی به فایل دانلود شده میدهیم.

1. ابتدا مسیر فایل را می دهیم. رنگ قرمز. ازانجای که الان فایل در روت است فقط اسم فایل وارد شده اگر داخل فولدر بود آدرس ان را می دادیم.
2. از قسمت زرد رنگ زرد برای چک وجود فایل استفاده می کنیم که در صورت نبودن فایل خطا برگرداند.
3. اگر فایل وجود داشت از خط سبز رنگ برای تبدیل به بایت کردن فایل استفاده می کنیم.
4. برای دانلود باید نوع فایل را به خروجی بدهیم در aspcore ما از کتابخانه فیروزه ای برای فهمیدن نوع فایل رنگ استفاده می کنیم. البته باید این کنابخانه را به سرویس های معرفی کنیم و اینجکت می کنیم رنگ خاکستری.



1. در قسمت صورتی رنگ هم فایل را برمی گردانیم. بایت شده فایل دانلود سپس نوع فایل و اسم فایل.

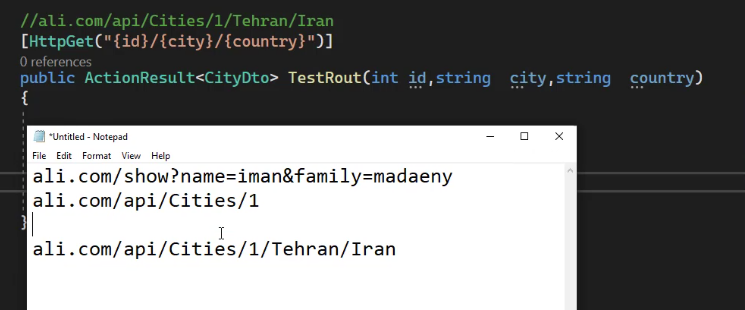


* 1. درس 25: روش های ارسا ل اطلاعات به API

1. استفاده از کوئری استرینگ ها: در این روش بعد از ? مطالب ارسالی به صورت json ارسال می شوند.



1. گرفتن اطلاعات از روت. در این مورد با قرار دادن اطلاعات در روت داده ها به api ارسال می شوند. دقت کنید روت در هنگام ران تایم قابل تغییر نیست. ولی کوئری استرینگ قابل تغییر است.



* 1. درس 26 ثبت اطلاعات از طریق frombody

در این قسمت Post بررسی می شودو اطلاعا ت FromBody به مرکز منتقل می شود.



* 1. جلسه 27 اعتبار سنجی مقادیر وردی API

در این قسمت با معرفی کتابخانه زیر اعتبار سنجی انجام شد.



* 1. درس 28 ویرایش اطلاعات از طریق HttpPut

در این جلسه از PUT برای ویرایش اطلاعات استفاده می کنیم.

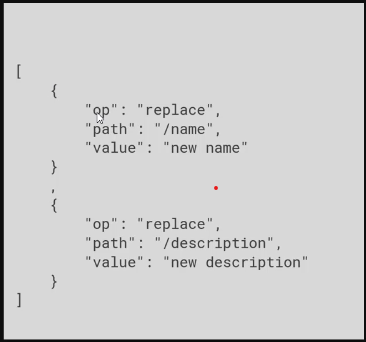


این اپ دیت کلی ابجکت هست و در ان تمام فیلدهای ابجکت به روز می شود.

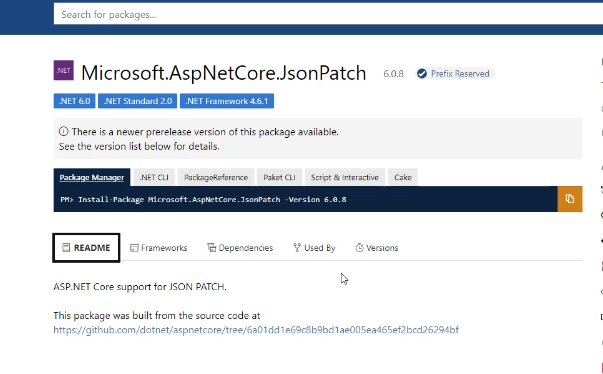
* 1. درس 29 معرفی HttpPatch برای ویرایش جزئی

برای ویرایش جزئی یک شی به جای استفاده از Put که تمام پراپرتی ها را اپدیت می کند از Patch استفاده می کنیم. در این روش بجای اینکه کل شی اپدیت شود فقط جزئی که می خواهیم را به روز می کنیم.

همانطور که از شکل زیر مشخص هست op عملیات را مشخص می کند که در اینجا جایگزینی هست path پراپرتی یا کلید را مشخص می کندو value مقدار جدید را مشخص می کند اگر قرار بود کلید دیگری تغییر بکند با کاما جدا می کنیم و json جدید را می نوسیم:



برای استفاده از Patch بایداز پکیج زیر استفاده کنیم:



* 1. درس 30 نصب بسته ها مورد نیاز Patch و تغییر فرمت از Json.Net

در این قسمت دو پکیج زیر را به لایه Api اضافه می کنیم:



سپس در فایل program.cs این سرویس را اضافه می کنیم:



* 1. درس 31پیاده سازی متد Patch

قسمت زرد رنگ ورودی های متد هستند و قسمت سبز مربوط به ورودی Json هستند که با پکیجی که نصب کردیم در واقع Json ورودی را به مدلی که می خواهیم تغییر میدهد.

بعد از پیدا کردن شهر ازId در قسمت آبی رنگ PointOf.. را پیدا می کنیم. از انجایی که این مورد را از حافظه خوانیدم در اسم ان از FromStore استفاده می کنیم که مشخص کنیم این از حافظه خوانده شده و با Object که از ورودی می اید وبرای Patch کردن متفاوت است.

در قسمت صورتی یک مدل میسازیم که قرار است تغییرات روی ان اعمال شود. دقت کنید که مقدار اولیه ان را از FromStore می دهیم و می سازیم. از انجایی که FromStore از دیتا بیس خوانده شده و ممکن است به صورت ناخواسته و بدون اعتبار سنجی اطلاعتی را با Update به دیتابیس اعمال کنیم . این مدل را میسازیم.

در قسمت قرمز ورودی Patch را روی مدلی از روی اصلی ساختیم اعمال می کنیم ToPatch . برای اعتبار سنجی Modelstate را هم میدهیم. ازانجای که اگر Modelstate درست نباشد هیچ خطای نمی دهد (تابع Apply)در ادامه خودمان مشخص می کنیم در صورت درست نبودن ان خطا بدهد.

در اخر هم در صورت درست بودن همه موارد روی مدل اصلی که از روی دیتابیس خوانیدم اعمال می کنیم.

در شکل زیر هم روش نوشتن تغییرات برای Patch می نویسیم. Path فیلد مورد نظر را مشخص می کند OP عملیات را مشخص می کند و Value مقدار جدید را مشخص می کند . دقت کنید که ابتدا از[] براکت استفاده کردیم و به ما این امکان را می دهد که اگر قصد تغییر فیلد دیگری را داشتیم در ادامه اضافه کنیم.





* 1. درس 32 اعتبار سنجی مدل ها در HttpPatch

تفاوت Update عادی با Put و Patch این است که در روش عادی اگر به یک فیلد مقداری ندهیم (قصد تغییر ان را نداشته باشیم)، ان فیلد در زمان Update با Null جایگرین می شود ولی Patch فقط مقدار فیلد مورد نظر تغییر می کند و بقیه به همان صورت قبل باقی می مانند.

در قسمت زرد رنگ درست بودن مدلی که قرار است به مدل اصلی FromeStore بررسی شده است. قسمت قبلی مدل که از وردی امده است (PatchDocument) را بررسی می‌کند و در ادامه مدل که قرار است به مدل که روی دیتا بیس برود بررسی می شود.



* 1. درس33 پیادهسازی HTTPDELETE روی پروژه

در این قسمت متد Delete را پیاده کردیم. به صورت زیر"



* 1. درس 34معرفی Dependencyinject و IOCدرAsp core

صحبت در مورد DI واینکه با استفاده از تزریق وابستگی سرویس ها ولایه ها رااز هم جدا می کنیم.

قسمت اضافه کردن سرویس ها که به کانتینر معرف است و در این قسمت با معرفی سرویس و در ادامه با تزریق سرویس در مکان هایی که نیاز داریم از ان استفاده می کنیم.

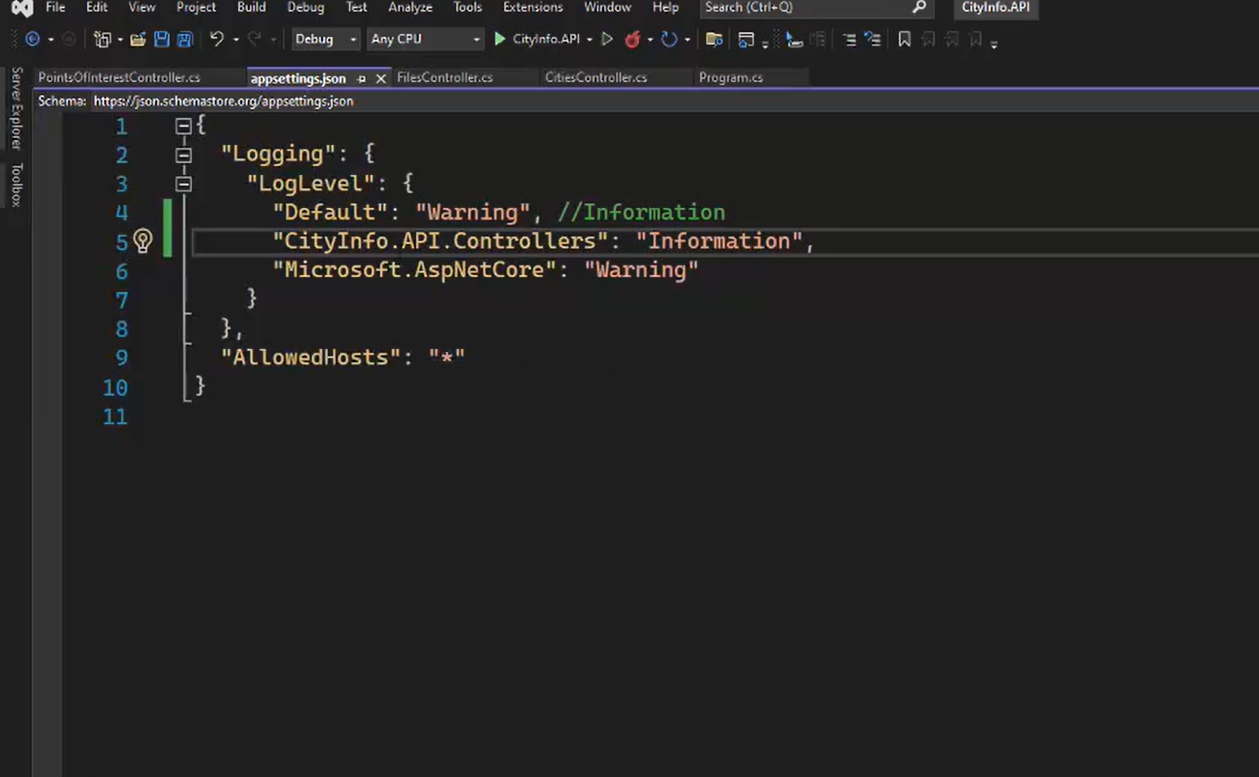
* 1. درس 35 استفاده از سیستم Logging با استفاده از Injection

سرویس Loging به صورت توکار در .NET قرار داده شده است برای همین نیاز نیست سرویس در فایل Program.cs معرفی شود. فقط کافی است در هر کنترلر تزریق شود. در این مورد تزریق به صورت کانستراکتور (قسمت زرد رنگ )است. در قسمت سبز مشخص می کنیم که اگر ورودی Logger Null بو یک خطا بده. رنگ صورتی روش استفاده از را نشان می دهد.

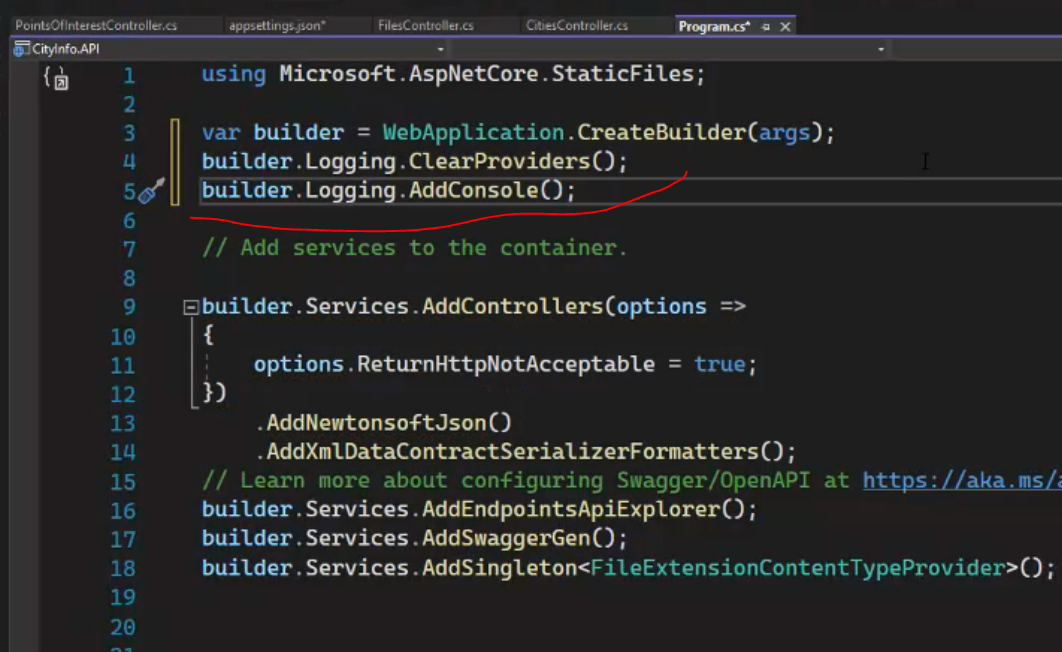


قسمت زیر هم تنظیمات Logger را در کل پروژه نشان می دهد.می توان تنظیمات ان را برای کل پروژه تغییر داد. برای یک کنترلر خاص هم می توان تنظیم کرد.





برای دسترسی و کنترل Logger از طریق هسته .Net هم می توان به صورت زیر در فایل Program.cs اقدام کرد.



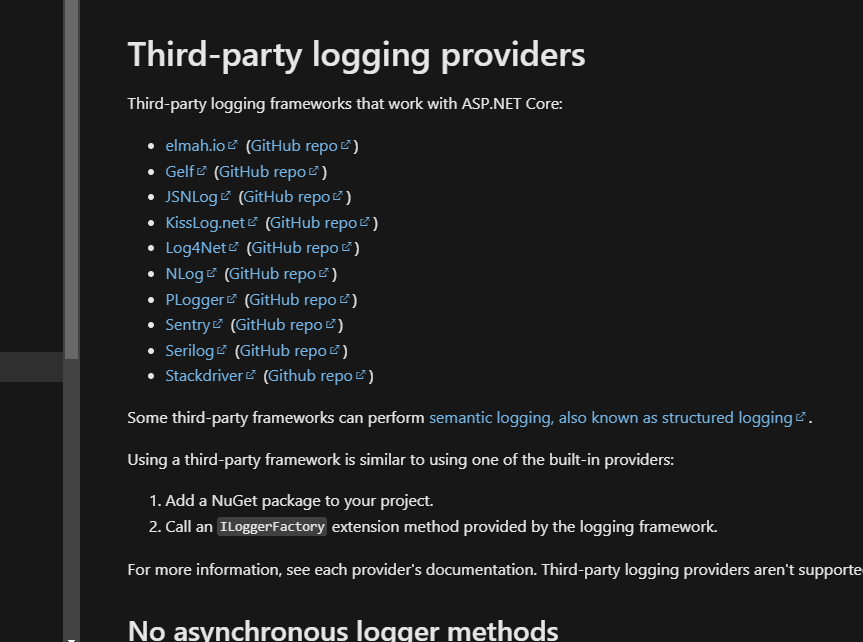
* 1. درس 36 لاگ زدن خطا ها در API

در این قسمت به لاگ زدن خطا ها پرداختیم که با عنوان زرد رنگ لاگ خطا ثبت می شود. قسمت سبز رنگ متنی است که در سیستم لاگ نشان داده می شود و قسمت ابی متنی است که در صفحه برای کاربر نشان داده می شود.

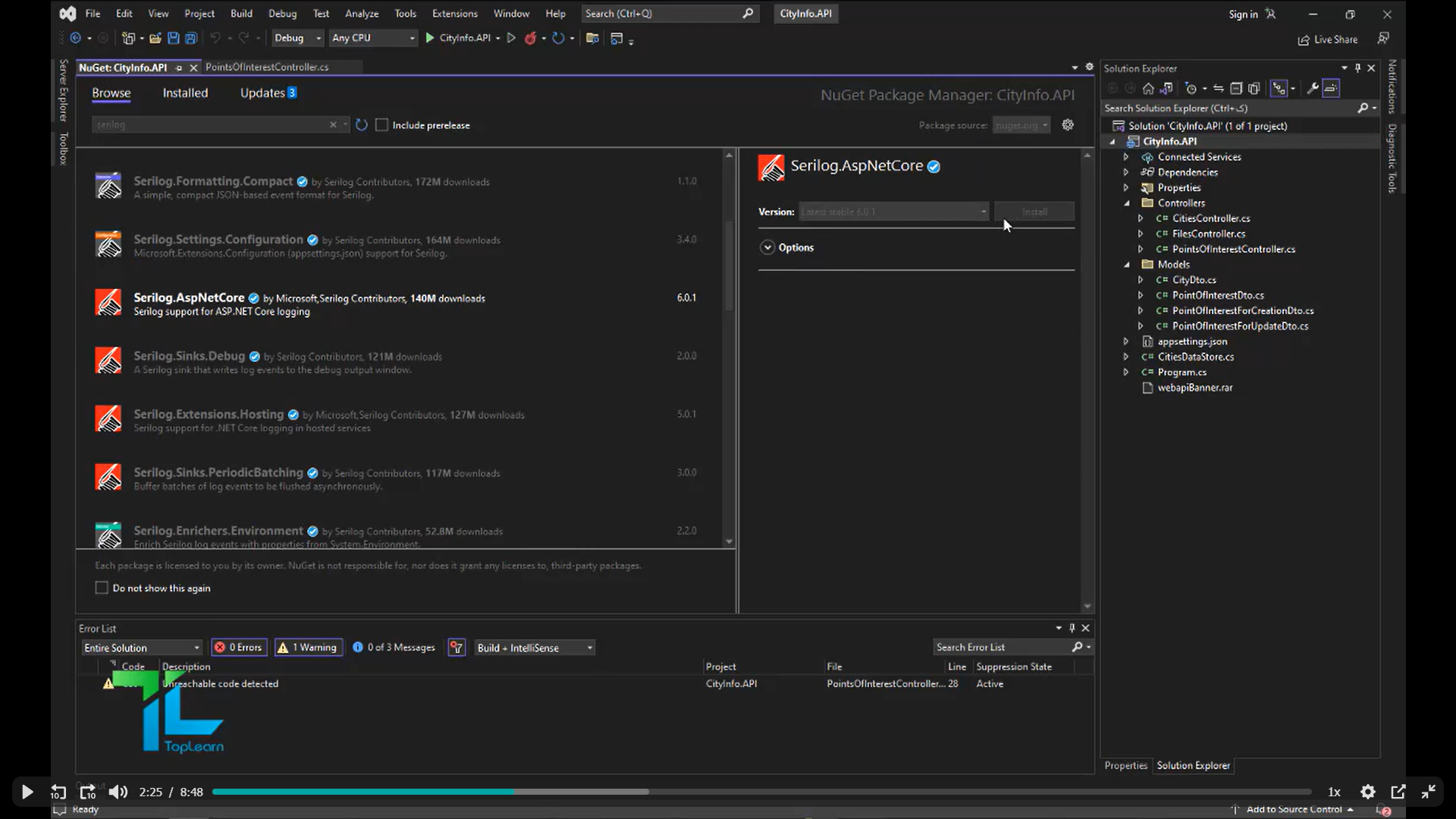


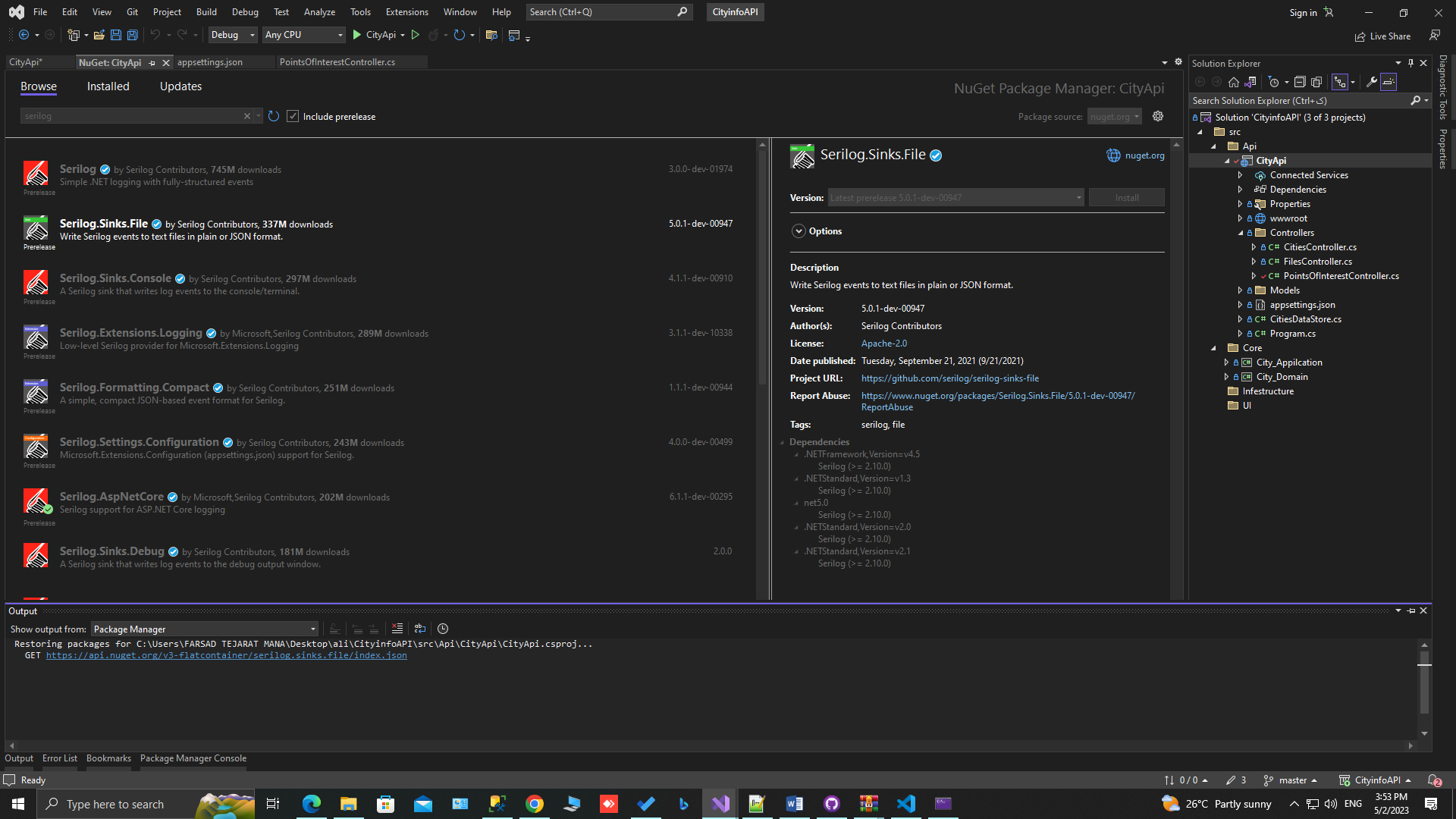
* 1. درس 37لاگ زدن Logger در فایل WebApi

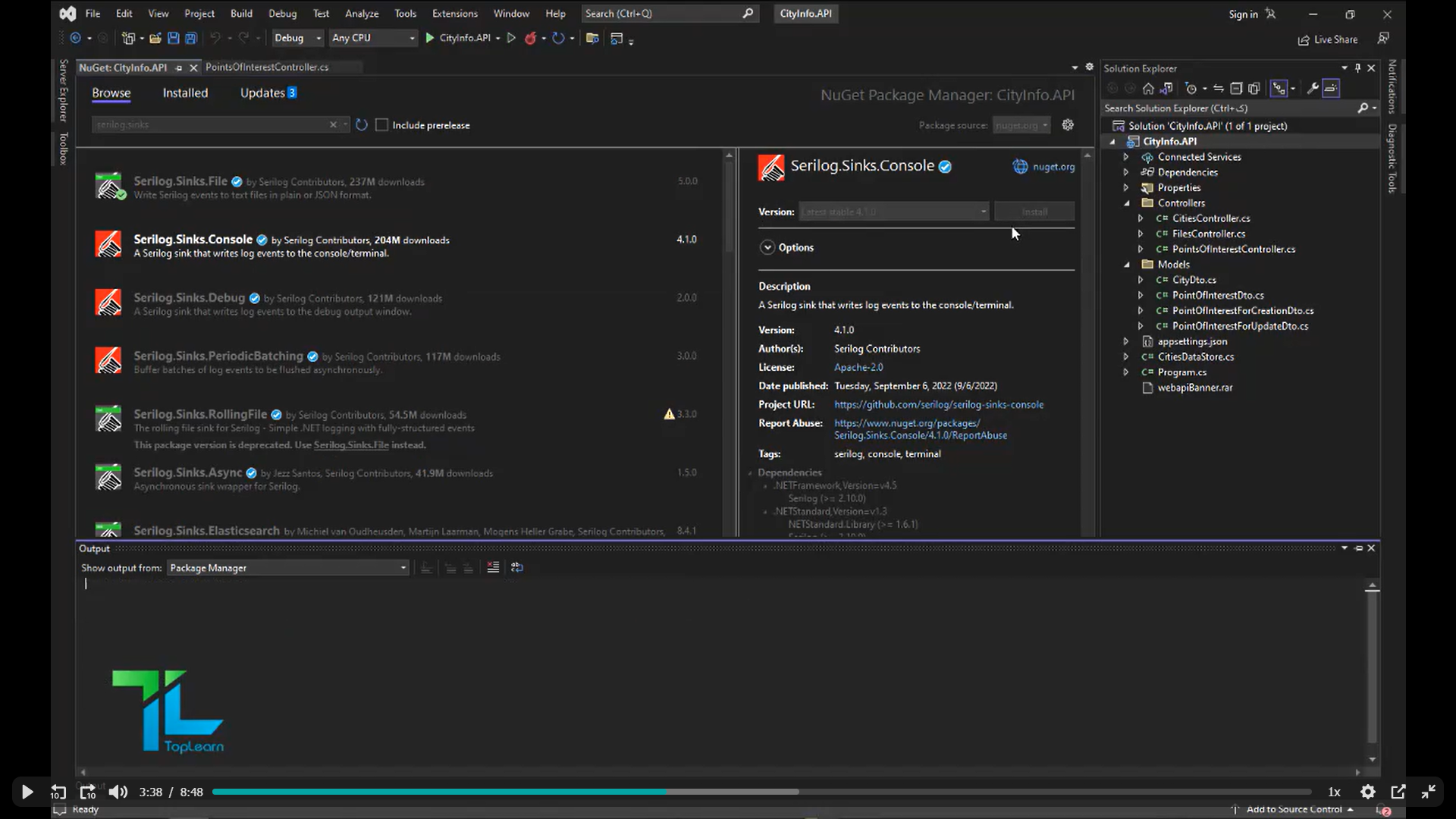
برای نگه داری لاگ های برنامه از Privider های دیگر استفاده می کنیم.



برای مثال در پروژه از Logger Serilogاستفاده می کنیم. برای استفاده از این لاگر سه پکیج زیر را دانلود می کنیم:







بعد از نصب پکیج ها لاگر ار در فایل Program.cs کانفیگ می کنیم:

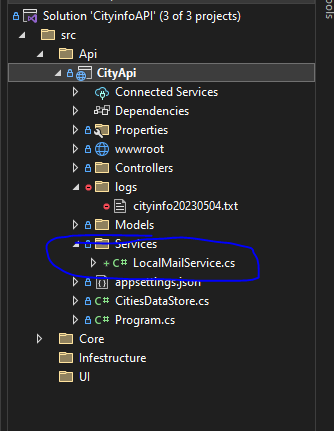
لاگر را به صورت زیر تعریف می کنیم. قسمت زرد رنگ مکان ذخیره لاگ ها و اینک روزانه درست شود را تعیین می کنیم. در قسمت صورتی رنگ هم به کور دستور می دهیم که از این لاگر استفاده کند.



در نهایت بعد از لاگ ها را می توانیم با این Provider مدیریت کنیم.

* 1. درس 38 ساخت سرویس ارسال ایمیل در WebApi

برای ایجاد سرویس ایمیل فولدر Services را ایجاد می کنیم واز کلاس LocalMailService را هم ایجاد می کنیم:





* 1. درس 39 ثبت سرویس ایمیل در Container و پیاده سازی DI

در این قسمت سرویس ایمیل را در فایل Program.cs معرفی و کانفیگ می کنیم. قسمت زرد رنگ این معرفی را نشان می دهد.



سرویس ایمیل را به صورت زیر Inject می کنیم:

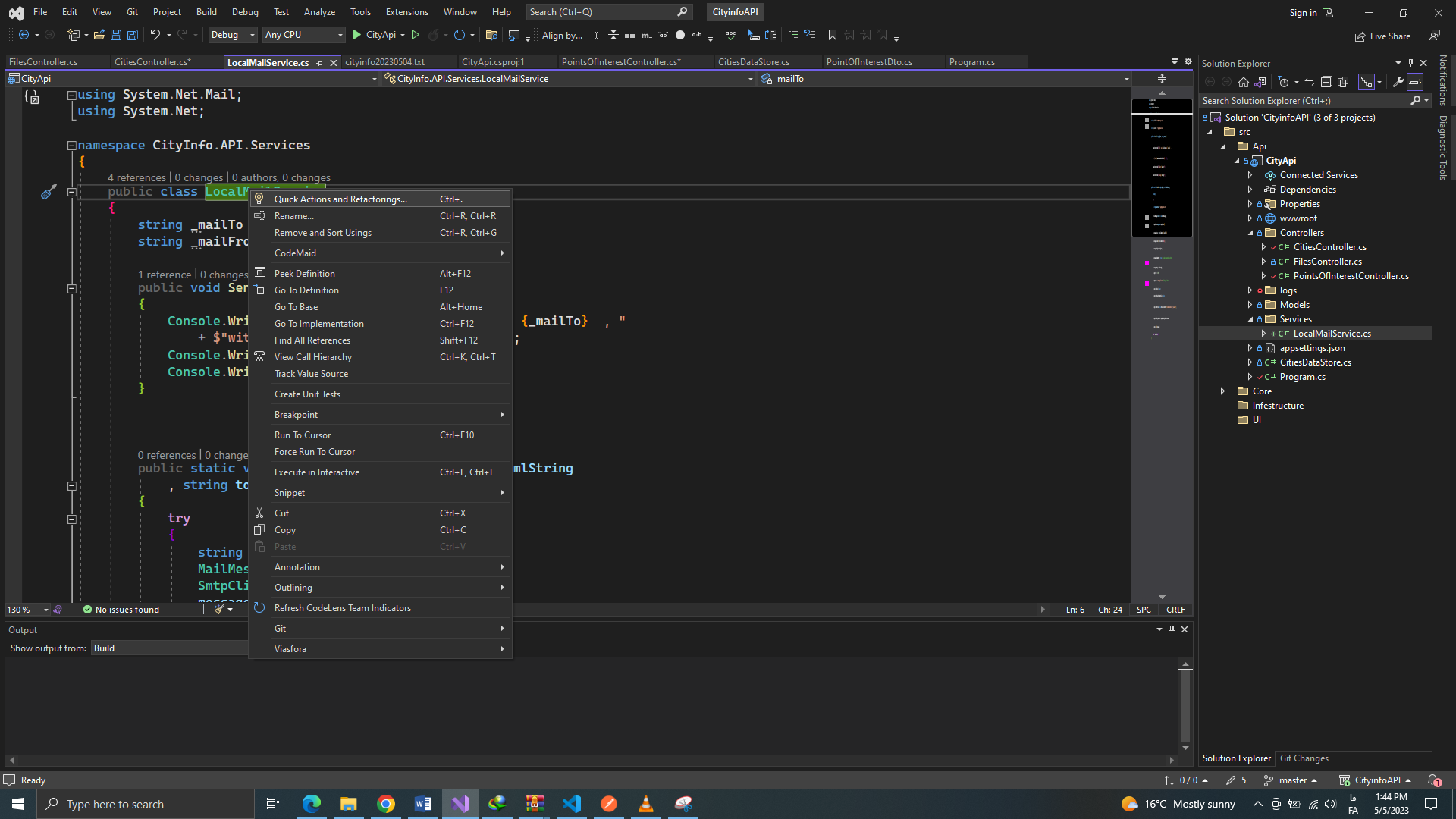


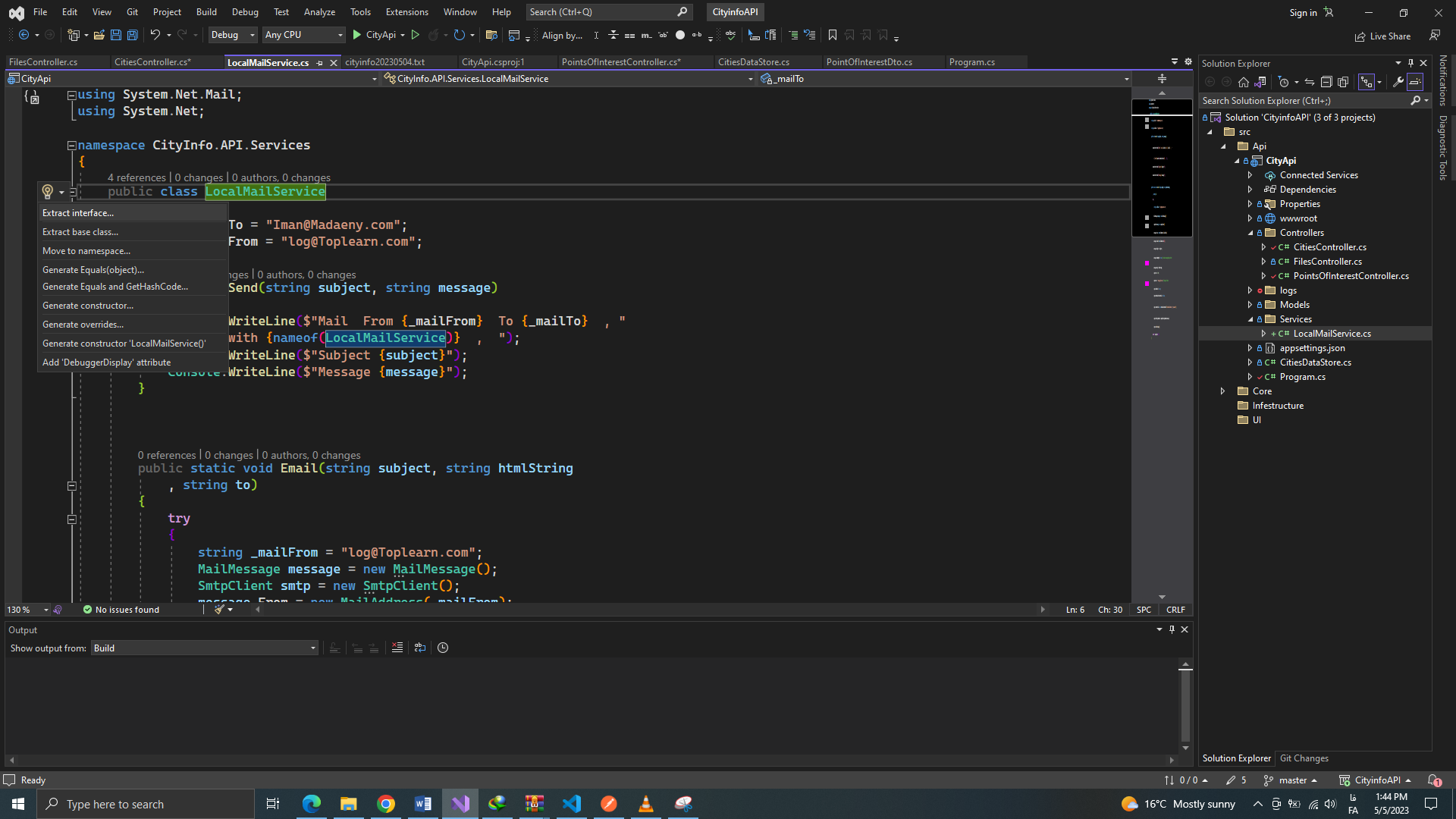
و در داخل متدها به صورت زیر استفاده می کنیم:



* 1. درس 40 بهبود ساخت و تعریف سرویس ها در WebApi

در این درس برای استفاده بهتر از سرویس ها و رعایت عدم وابستگی به سرویس ها از Interface یا ..استفاده می کنیم. برای ساخت اینترفیس به صورت زیر عمل می کنیم:





اینترفیس به صورت زیر ساخته می شود:



بعد از ساخت اینترفیس سرویس دیگری را برای ایمیل معرفی می کنیم:



و فایل Program.cs را هم به صورت زیر تغییر میدهیم:



DI را هم تغییر میدهم:

