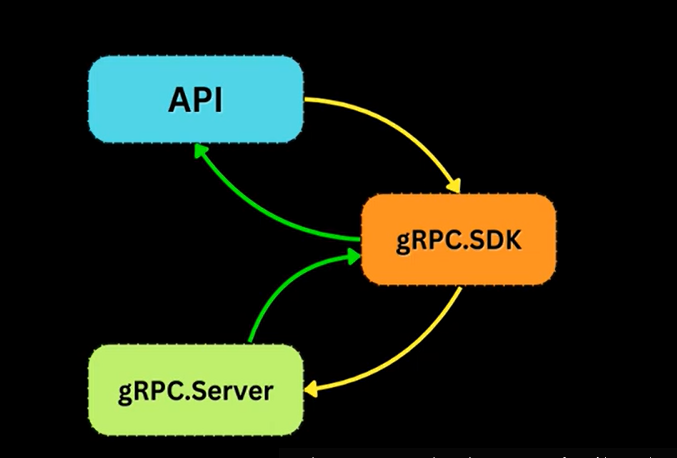
Grpc

یک راه بسیار خوب برای ارتباط backend به backend است همچنین برای ارتباط بین سرویس ها در معماری microservice نیز کاربرد دارد. در حال حاظر مرورگر ها از grpc پشتیبانی نمی کنند اما در حال انجام کار های برای حل این مسئله هستند.

سناریو:



General remote procedure call که توسط گوگل ساخته شده است.

زمانی که یک پروژه grpc service در asp.net ایجاد می کنیم ساختار آن به صورت زیر است.

فولدر :

Protos و Services

Grpc از پروتکل buffer استفاده می کند که از proto buff به عنوان serializer استفاده می کند مهم ترین نکته grpc استفاده از proto buff است در http ما از json یا xml برای ارسال و دریافت درخواست ها استفاده می کردیم که هنگام serialize کردن حجم زیادی می گیرد.

در grpc به دلیل این که به صورت پیشفرض از proto buff استفاده می شود به هنگام serialization تمامی داده ها به bytes تبدیل می شود که بسیار فشرده است.

Point اصلی این است که وقتی داده ای را در شبکه ارسال می کنیم هرچه حجم کمتری داشته باشد پهنای باند کمتری اشغال می کند و کارایی یا effiency بهتری دارد.

**فولدر Protos**

این فولدر دارای محتوای proto است که یک قرار دادی است که grpc فقط آن را می شناسد پس بنابر این هر منطقی را باید در این فایل قرار بدهیم.

کلمه کلیدی service :

این کلمه مانند یک interface است

کلمه کلیدی rpc :

در واقع یک متد است remote procedure call است یعنی به جای این که یک api را در سرور دیگری صدا کند یک rpc یا remote method control را صدا می کند که در یک می تواند قسمتی از یک سرور دیگر باشد

الگوی تعریف

Service teset {

rpc MethodName (input parameter) returns (outPutParameter);

}

کلمه کلیدی message :

یک کلاس است در سی شارپ

نکته : در زمان runtime فایل proto تبدیل به یک کلاس سی شارپی می شود.

در زمان build نیز این کار را انجام می دهد. می توانید در فولدر service با نام سرویس خود فایل ایجاد شده را ببینید. به طور مثال اگر یک property جدید به فایل request model خود اضافه کنیم به صورت auto generated فایل service جدید ساخته می شود و تغییرات انجام شده در دسترس است.

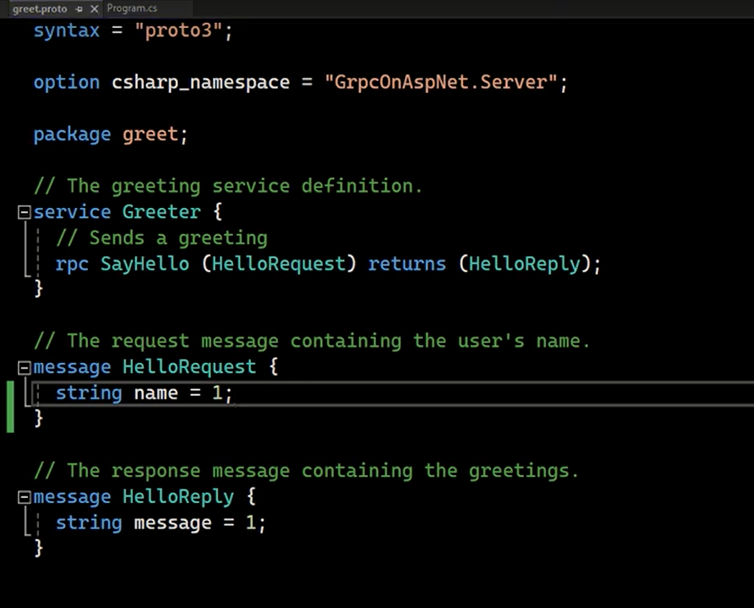
نکته:

فایل proto برای هر property به یک index نیاز دارد که برای سریالایز کردن درخواست و پاسخ کاربرد دارد و با استفاده از index این کار را انجام میدهد به دلیل این که خروجی ما bytes است و از index ها برای فشرده سازی output استفاده می کند نه نام property ها.

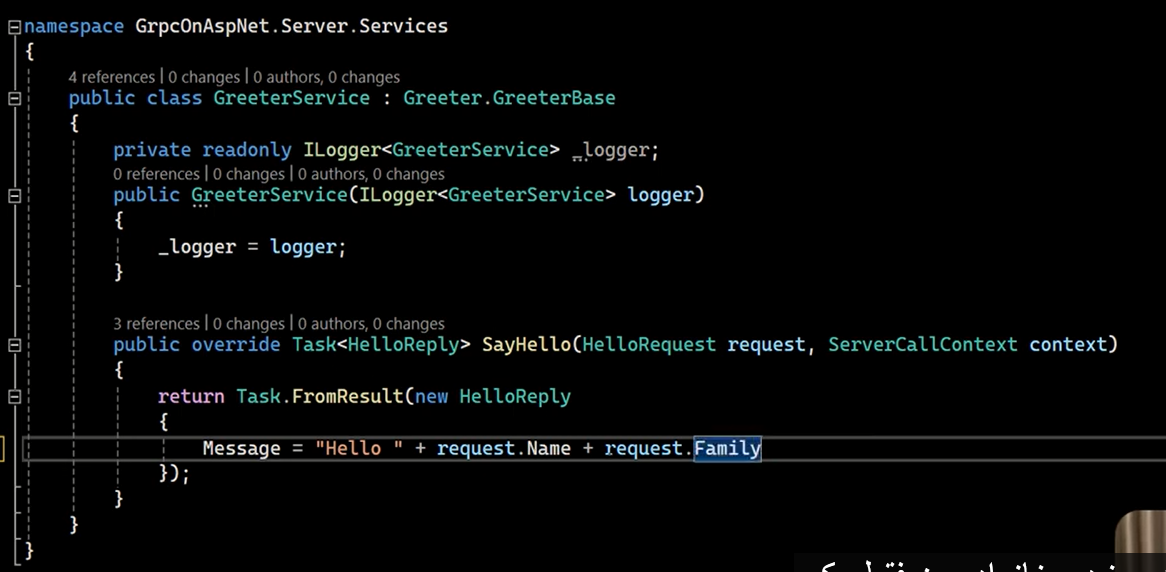
**زمانی که در حال طراحی یک مدل هستید باید حواستان به index ها باشد و نباید در آینده تغییر کند.**

**اگر** تغییری ایجاد شود در index ها می تواند سبب بروز خطای احتمالی در سریالایز کردن message های قدیمی شود

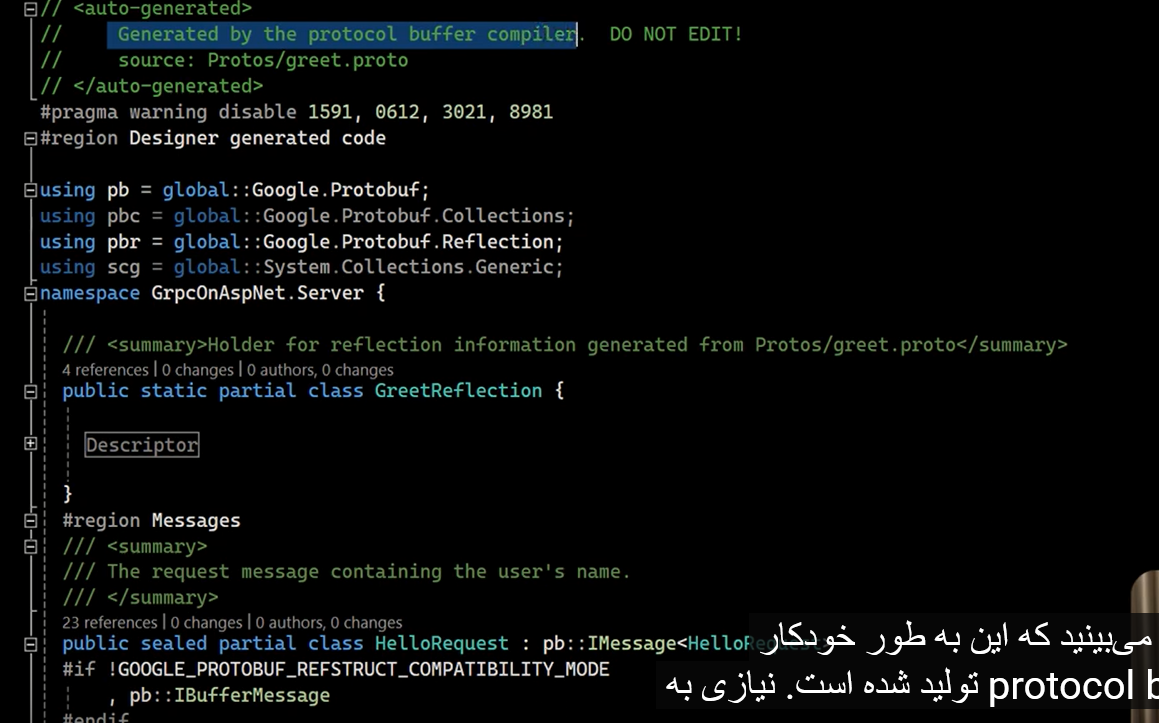
فایل proto



فایل service



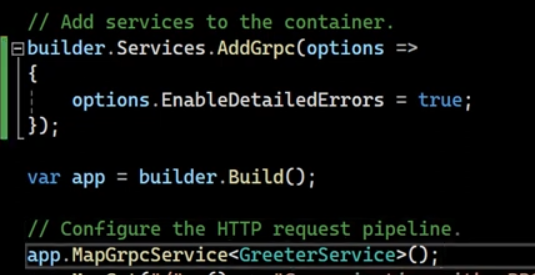
فایل autoGenerated که نباید تغیر کند.



در فایل program.cs نیز می توانیم تنظیماتی را دلخواه انجام دهیم.

EnableDeatiledErrors = true مناسب دیدین جزئیات خطای بیشتر.

IgnoreUnknowServices زمانی که از سرور grpc شما یک متدی که پیاده سازی نشده است را فراخوانی کنند این باعث می شود خطا یا ex رخ ندهد و ادامه فرایند پردازش درخواست به middleWear های دیگر برود. به صورت دلخواه true یا false می تواند ست شود. False مناسب تر است.



Sdk

به دلیل این که امکان فراخوانی grpc به صورت مستقیم توسط مرورگر وجود ندارد باید از sdk به عنوان یک قرار دارد برای ارتباط با سرور grpc استفاده کنیم.

یک classLibrary به نام GrcSampleSdk

پکیج های مورد نیاز classLibrary ما:

Google.Protobuf برای سریالایز کردن درخواست و پاسخ

Grpc.net.client چون پروژه کلاینت ما است

Grpc.tools مسئول تبدیل فایل proto به کلاس های c# است.

Grpc.net.client.factroy کمتر استفاده می شود برای مواقعی که بخواهیم سمت کلاینت sdk را پیاده سازی کنیم.

نکته : protobuf در واقع قراردادی بین grpc client و grpc server است. برخی از توسعه دهندگان فایل protobuf در سمت کلاینت هم کپی و پیست می کنند که پیشنهاد نمی شود سبب تکرار کد است و هر تغییری نیاز به کپی مجدد دارد.

راهکار ها:

1-ایجاد یک پروژه مشترک

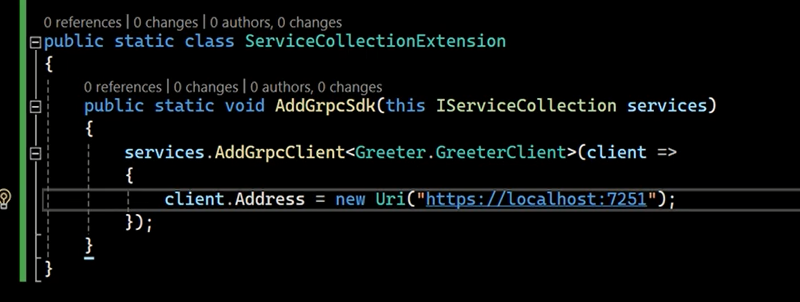
2- از پروژه grpc server و فایل csproj باید یک خط را کپی کرده و در فایل client یا classLibrary خود و در csproj آن اضافه کنیم.

نکته : در سمت کلاینت بایدgrpcService = “Client” تغییر پیدا کند



حالا اگر پروژه را build بگیریم میبینم که قابلیت استفاده از کلاس های grpc در پروژه کلاینت ما اضافه شده است.

در قطعه کد زیر سرویس grpc خود را به پروژه وب اضافه می کنیم با استفاده از یک Ex متد.



نکته : از خوبی های grpc این است که زمانی که در rpc method تعریف می کنیم هم sync هم async را خود auto generated می سازد.

نکته : برای ex handling در grpc می توان از کلاس RpcException استفاده کرد و پروپرتی statusCode را از آن دریافت کرد که با http status فرق دارد.