### 什么是gRPC

gRPC是google开源的一个高性能、跨语言的RPC框架，基于HTTP2协议，采用ProtoBuf 定义的IDL。

gRPC 的主要优点是：

* 现代高性能轻量级 RPC 框架。
* 协定优先 API 开发，默认使用协议缓冲区，允许与语言无关的实现。
* 可用于多种语言的工具，以生成强类型服务器和客户端。
* 支持客户端、服务器和双向流式处理调用。
* 使用 Protobuf 二进制序列化减少对网络的使用。

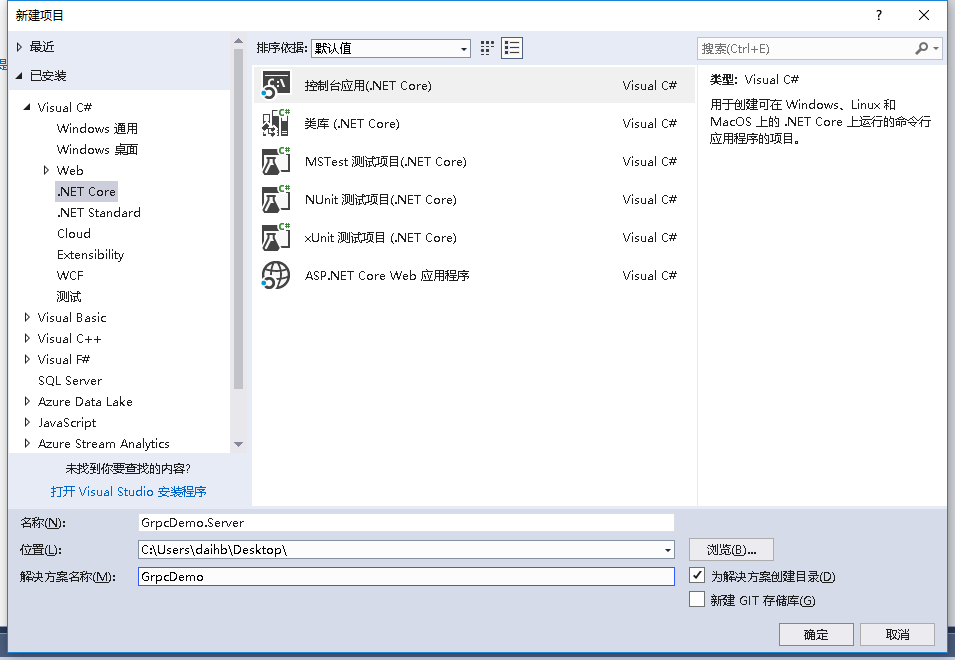
这些优点使 gRPC 适用于：

* 效率至关重要的轻量级微服务。
* 需要多种语言用于开发的 Polyglot 系统。
* 需要处理流式处理请求或响应的点对点实时服务。

更多介绍请前往 https://grpc.io/docs/guides/

### 开始

1. 打开VS2017，创建一个控制台项目，解决方案名称GrpcDemo，解决方案名称叫GrpcDemo.Server。



点击确定，然后创建项目。打开NuGet安装Google.Protobuf，Grpc，Grpc.Tools三个包。

1. 建立Protos文件夹，在文件夹下建立一个greet.proto文件。

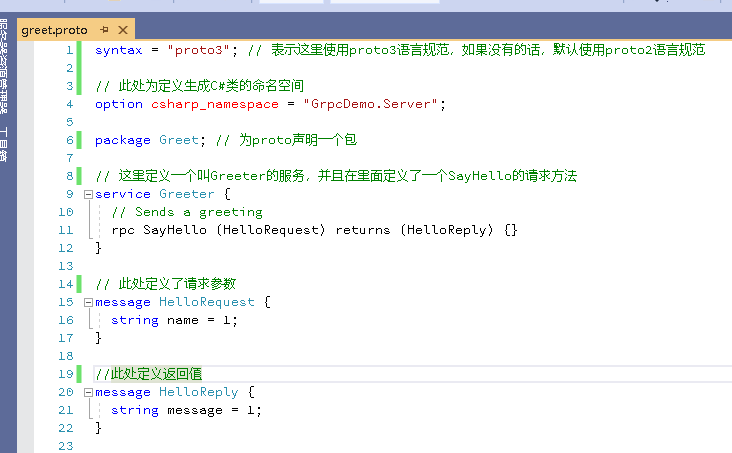


### proto文件

GRPC使用约定优先的API开发方法。默认情况下，使用协议缓冲区(Protobuf)作为接口设计语言(IDL)。这个.proto文件包含：

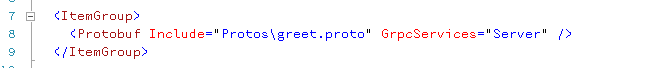
* GRPC服务的定义。
* 在客户端和服务器之间发送的消息。

有关Protobuf文件语法的更多信息，请参见[正式文件(原型)](https://developers.google.com/protocol-buffers/docs/proto3).如我们模板中创建的greet.proto 文件内容如下



* 定义Greeter服务。
* 这个Greeter服务定义SayHello请求。
* SayHello发送HelloRequest消息并接收HelloReply信息：

我们来看看，这个*.proto*文件是如何包含在项目中的呢，其实，如果你打开项目的.csproj文件就会看到，通过将该文件添加到<Protobuf>的ItemGroup中即可，如下所示：



Include属性表示将greet.proto文件包含在我们的项目当中，GrpcServices="Server"表示仅在服务器项目中生成资源文件。但是这样远远不够，以后我们可能还会在这个文件夹中增加其他proto文件。所以我们需要改动一下。

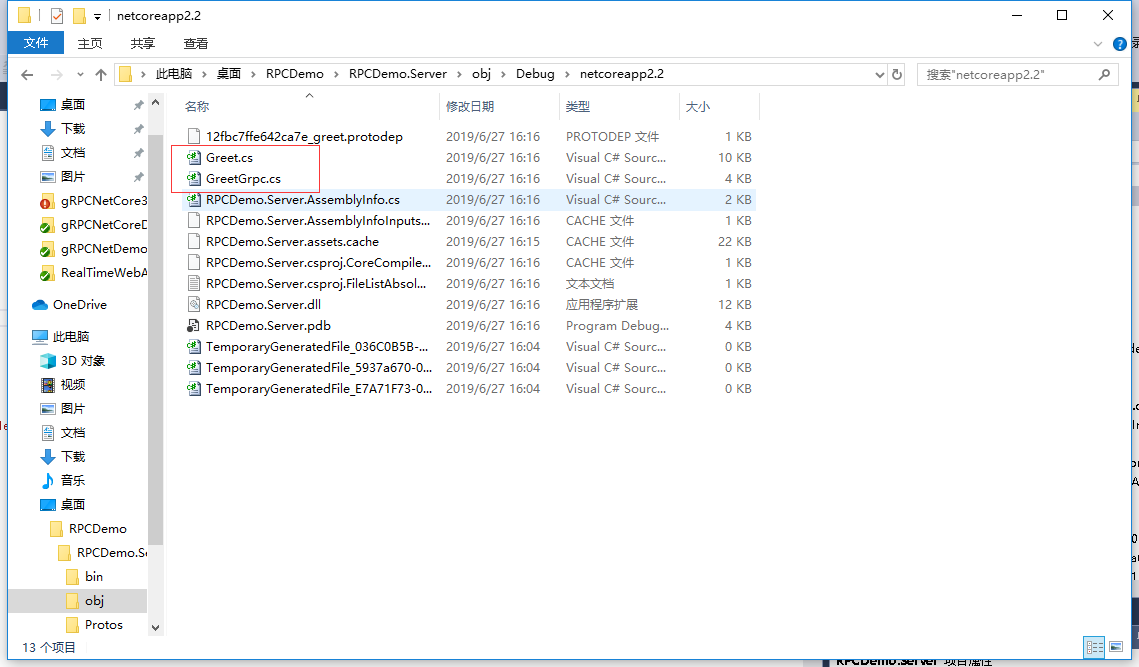
<ItemGroup>

<Protobuf Include="Protos\\*.proto" GrpcServices="Server" />

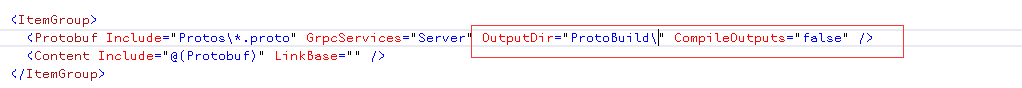
<Content Include="@(Protobuf)" LinkBase="" />

</ItemGroup>

这里表示将项目下的Protos文件夹所有以proto结尾的文件包含到项目中。再右键项目生成一下，然后打开项目的obj\Debug\netcoreapp2.2目录，就能发现两个生成的类文件。分别是Greet.cs和GreetGrpc.cs。有兴趣可以打开进行深入研究。



如果不想生成的文件放到这个目录，可以配置一下上面所说的<Protobuf>节点，配置一些具体的信息，可以配置文件的生成目录。



设置OutputDir属性就可以设置文件的输出目录，上面这里就设置了文件的输出目录为ProtoBuild目录，另外还设置了CompileOutputs="false"，这是为了防止止将生成的文件编译为程序集。编译一下，就可以看到文件出现在了ProtoBuild目录。



关于Protobuf的详细配置信息，可以参考下面的链接。里面还有各种奇奇怪怪的配置。

<https://github.com/grpc/grpc/blob/master/src/csharp/BUILD-INTEGRATION.md>

编译项目能够将proto文件生成类文件，是引用Grpc.Tools了这个工具包的原因。

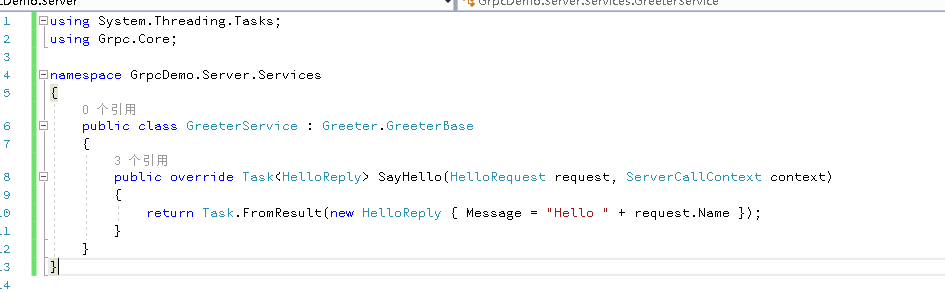
### Services 文件夹中的具体的gRPC服务

创建Services文件夹，并创建GreeterService类。

下面再看一下services目录里的文件。我们知道Grpc.Tools工具包将根据.proto文件的定义翻译并生成对应的C#类型的文件。

对于服务器端资产，将生成一个抽象的服务基类型。基类型包含在.proto文件中包含的所有GRPC调用的定义。然后，您将创建从此基类型派生的具体服务实现，并实现GRPC调用的逻辑。对于前面描述的greet.proto示例，将生成包含虚拟SayHello方法的抽象GreeterBase类型。具体的实现GreeterService重写该方法并实现处理GRPC调用的逻辑。

正如GrpcDemo.Server项目中的Services\GreeterService.cs中的代码



这样子就编写好了服务，下面我们要在Program.Main来启用服务。

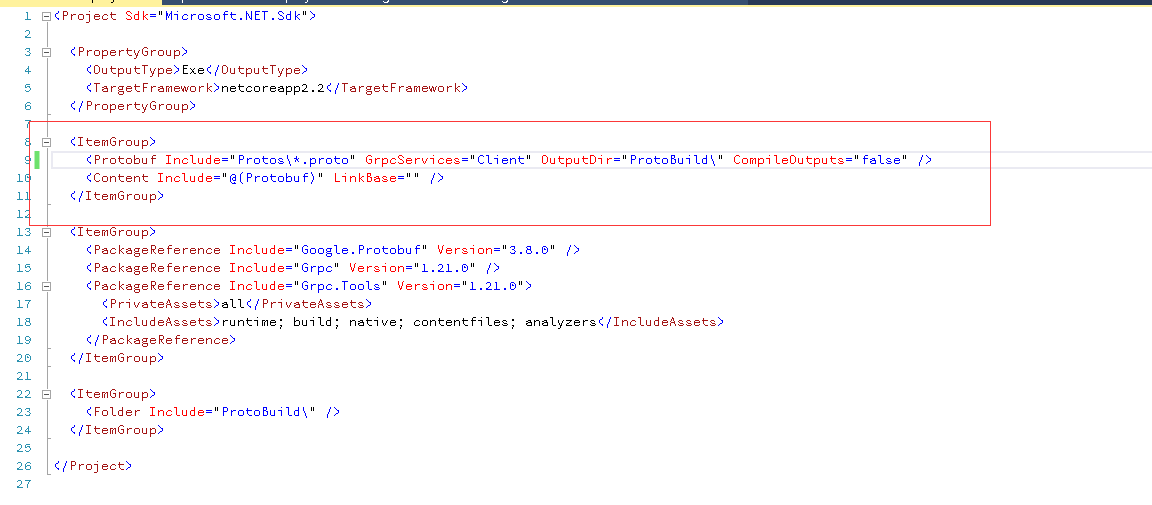


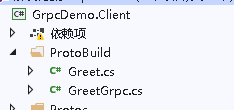
### 编写客户端调用

右键解决方案创建一个GrpcDemo.Client的项目，打开NuGet安装Google.Protobuf，Grpc，Grpc.Tools三个包。将服务端项目的Protos文件夹复制到客户端项目，新建ProtoBuild文件夹存放proto编译后的类。

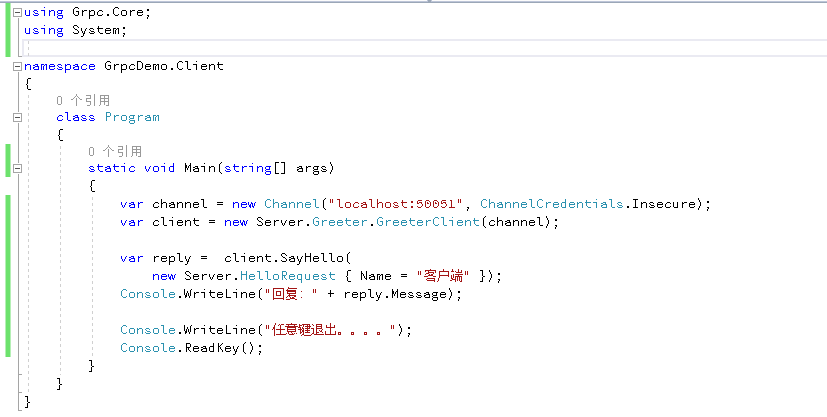


然后更改客户端项目的csproj文件，修改为以下内容。与服务端不同的是Protobuf节点的GrpcServices="Client"属性，表示仅在客户端项目生成。



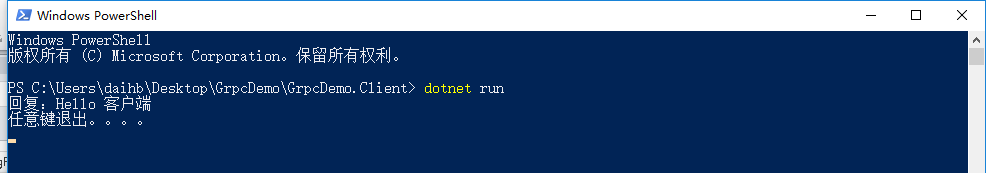
然后右键生成项目，这时候ProtoBuild目录下应该有生成的类文件了

在Program文件中编写以下代码进行调用服务端：



然后先启动服务端项目，再启动客户端，就能成功调用服务端的方法了。





可以看到成功的调用的服务端的服务，并且输出返回信息了。