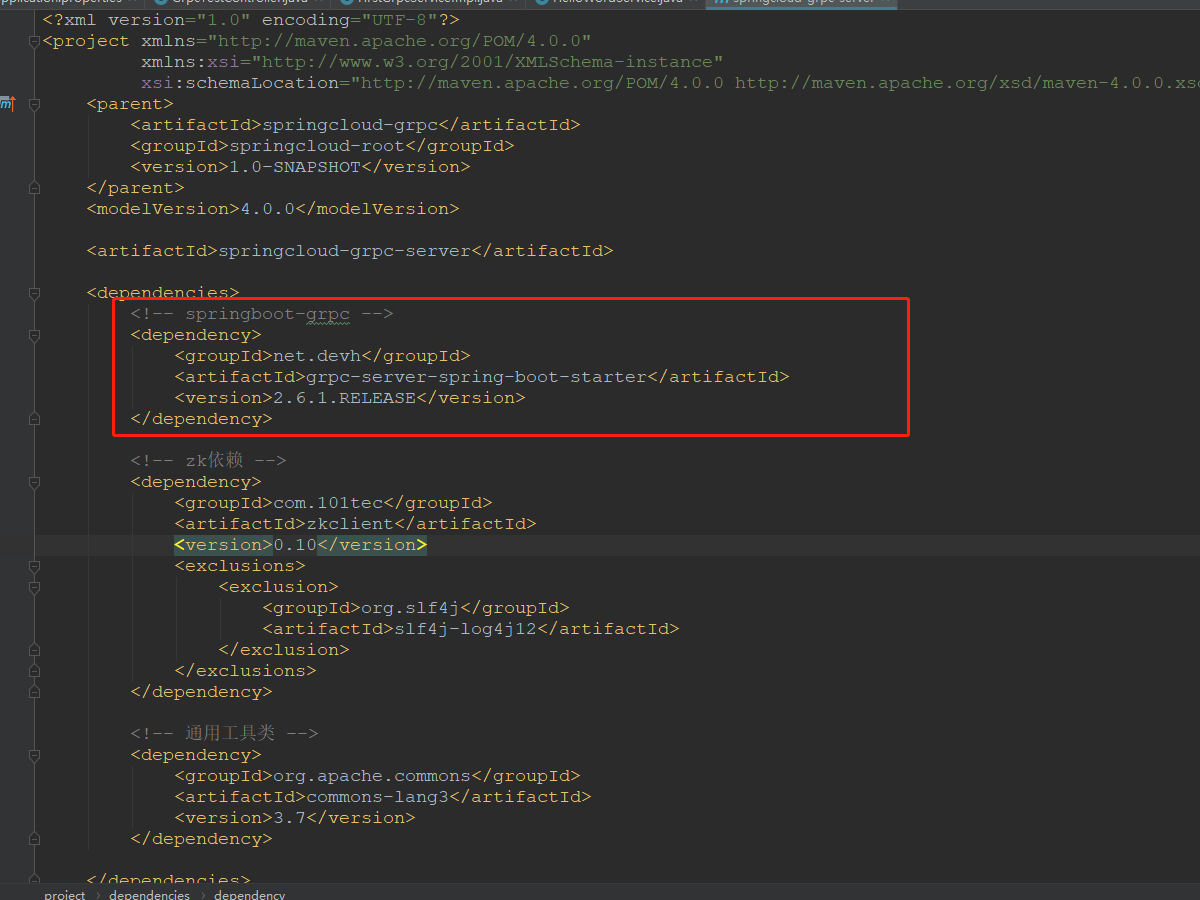
1. **纯粹提供grpc服务和调用grpc服务**

## 服务端

1. 配置pom.xml文件，导入grpc的依赖和插件



<!-- springboot-grpc -->

<dependency>

<groupId>net.devh</groupId>

<artifactId>grpc-server-spring-boot-starter</artifactId>

<version>2.6.1.RELEASE</version>

</dependency>

<!--

protocol buffers编译插件项

protoc的版本要和protobuf-java jar包的版本一致

-->

<build>

<extensions>

<extension>

<groupId>kr.motd.maven</groupId>

<artifactId>os-maven-plugin</artifactId>

<version>1.5.0.Final</version>

</extension>

</extensions>

<plugins>

<plugin>

<groupId>org.xolstice.maven.plugins</groupId>

<artifactId>protobuf-maven-plugin</artifactId>

<version>0.5.1</version>

<configuration>

<protocArtifact>com.google.protobuf:protoc:3.5.1-1:exe:${os.detected.classifier}</protocArtifact>

<pluginId>grpc-java</pluginId>

<pluginArtifact>io.grpc:protoc-gen-grpc-java:1.14.0:exe:${os.detected.classifier}</pluginArtifact>

</configuration>

<executions>

<execution>

<goals>

<goal>compile</goal>

<goal>compile-custom</goal>

</goals>

</execution>

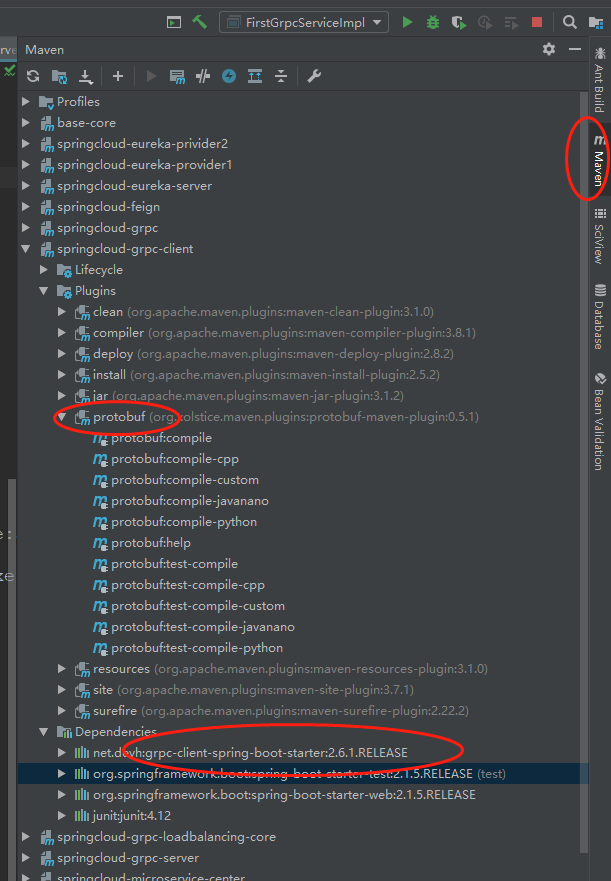
</executions>

</plugin>

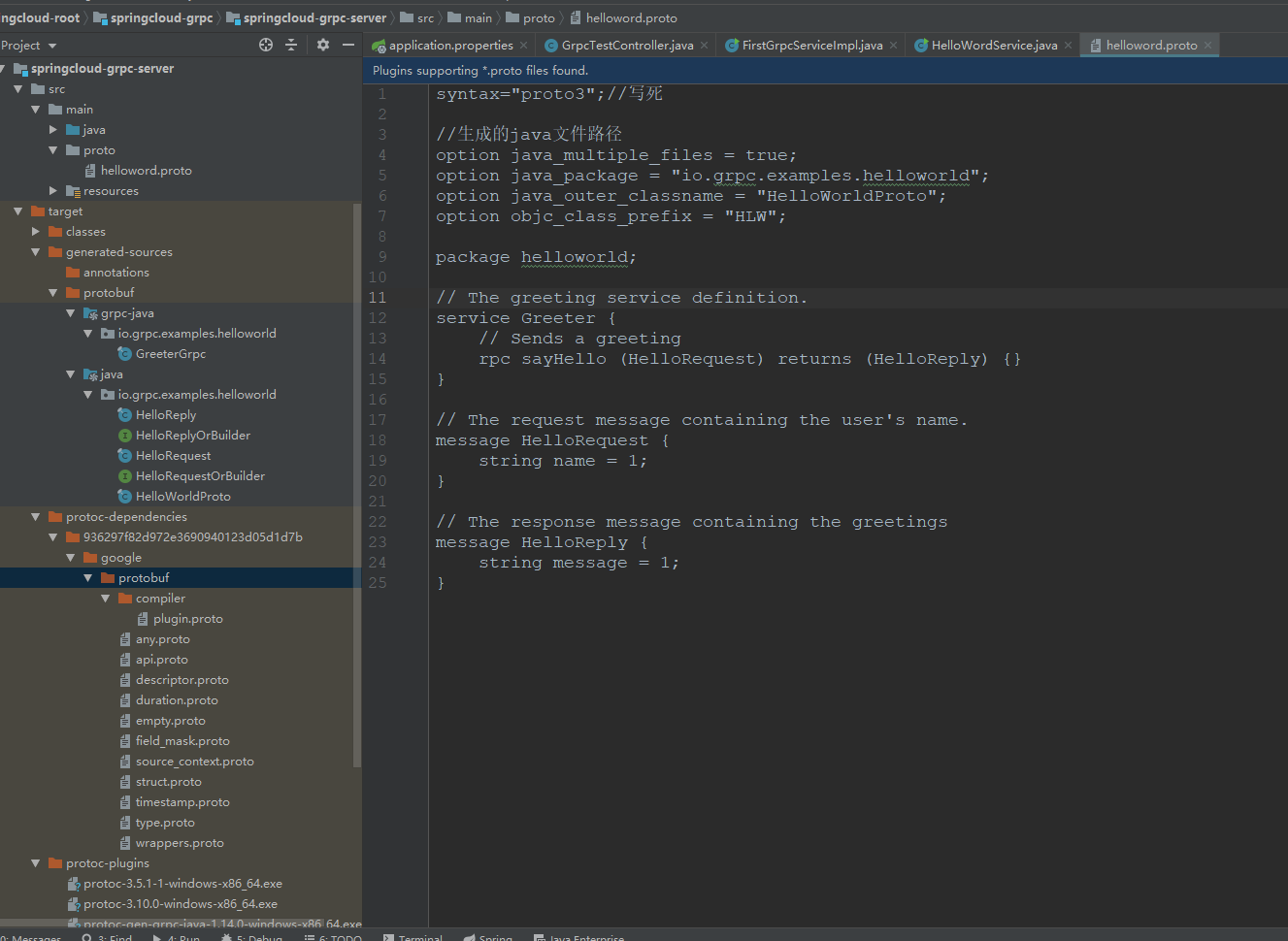
</plugins>

</build>

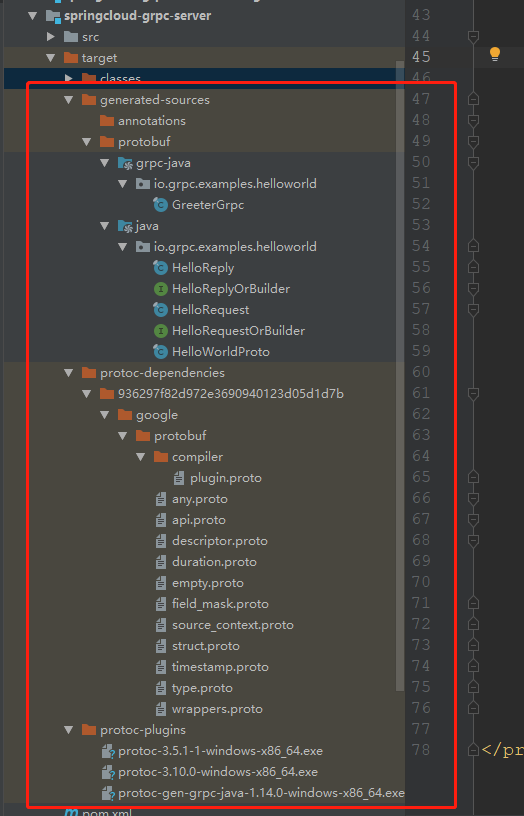
添加完依赖和插件就可以在Maven projects中看到添加的依赖了

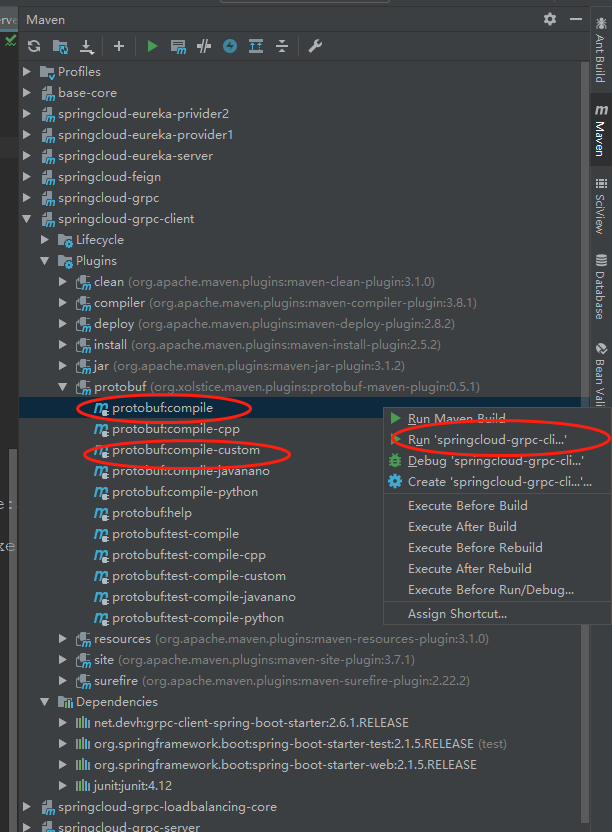


1. 编写proto文件，如下，HelloRequest为请求参数对象，String name=1,表示第一个参数是String类型的，名称是name；HelloReply是返回结果对象，String message=1,表示第一个参数是String类型的，名称是message.



1. proto文件编写完之后，需要使用插件编译proto文件生成对应的java文件。

一次选择如图中的两个选中项，然后右键->run...,就会生成target下除了class文件夹下的文件。



1. 编写服务端代码：然后运行main方法即可提供服务。

import io.grpc.Server;

import io.grpc.ServerBuilder;

import io.grpc.examples.helloworld.GreeterGrpc;

import io.grpc.examples.helloworld.HelloReply;

import io.grpc.examples.helloworld.HelloRequest;

import io.grpc.stub.StreamObserver;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.io.IOException;

import java.util.Optional;

public class HelloWordService {

private static final Logger logger = LoggerFactory.getLogger(HelloWordService.class);

private Server server;

/\*\*

\* 启动服务

\*

\* @param port

\* @throws IOException

\*/

private void start(int port) throws IOException {

server = ServerBuilder.forPort(port).addService(new BasicCalImpl()).build().start();

logger.info("服务已经启动,监听端口：" + port);

Runtime.getRuntime()

.addShutdownHook(

new Thread(

() -> {

logger.warn("监听到JVM停止,正在关闭GRPC服务....");

HelloWordService.this.stop();

logger.warn("服务已经停止...");

}));

}

/\*\* 关闭服务 \*/

public void stop() {

Optional.of(server).map(s -> s.shutdown()).orElse(null);

}

/\*\*

\* 循环运行服务,封锁停止

\*

\* @throws InterruptedException

\*/

public void blockUnitShutdown() throws InterruptedException {

if (server != null) {

server.awaitTermination();

}

}

/\*\*

\* 程序的主运行窗口

\*

\* @param args

\* @throws IOException

\* @throws InterruptedException

\*/

public static void main(String[] args) throws IOException, InterruptedException {

HelloWordService service = new HelloWordService();

service.start(8888);

service.blockUnitShutdown();

}

/\*\* 实现的服务类 \*/

static class BasicCalImpl extends GreeterGrpc.GreeterImplBase {

@Override

public void sayHello(HelloRequest request, StreamObserver<HelloReply> responseObserver) {

// 获取数据信息

String name = request.getName();

logger.info("受到来自客户端消息：name:"+name);

// 计算数据

HelloReply response =

HelloReply.newBuilder()

.setMessage("你好，" + name + "，welcome to grpc")

.build();

// 返回数据，完成此次请求

responseObserver.onNext(response);

responseObserver.onCompleted();

}

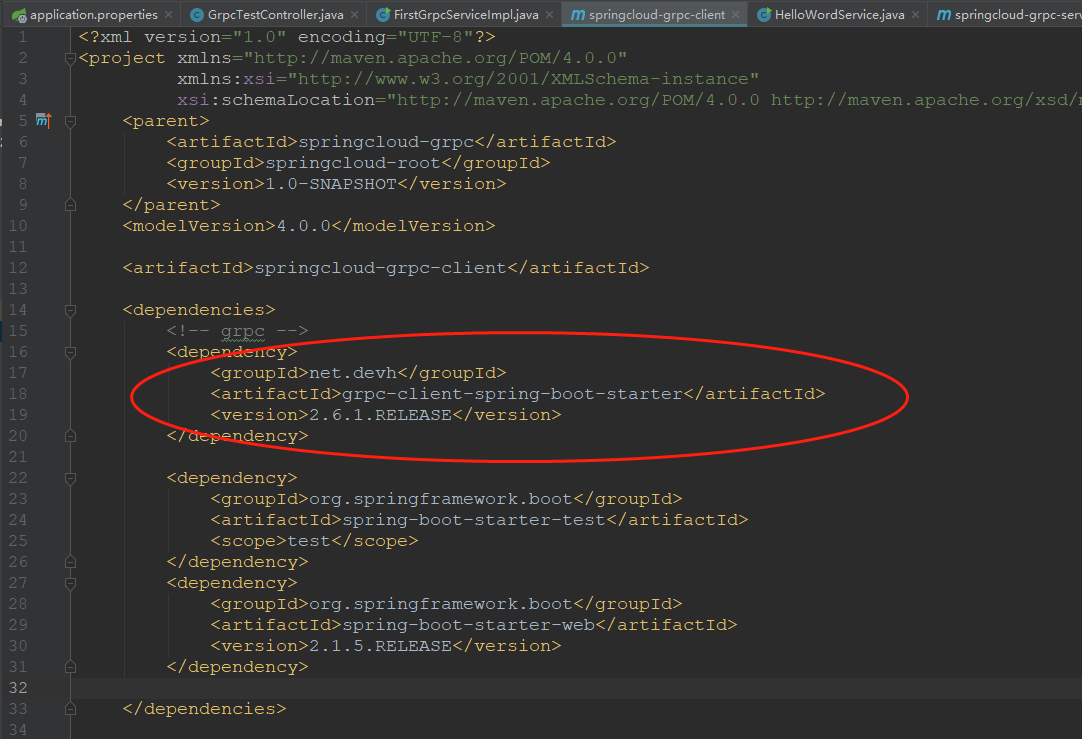
}

}

到此，服务端代码编写完成。

## 客户端

1. 配置pom.xml文件，导入grpc的依赖和插件



<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<parent>

<artifactId>springcloud-grpc</artifactId>

<groupId>springcloud-root</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>springcloud-grpc-client</artifactId>

<dependencies>

<!-- grpc -->

<dependency>

<groupId>net.devh</groupId>

<artifactId>grpc-client-spring-boot-starter</artifactId>

<version>2.6.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>2.1.5.RELEASE</version>

</dependency>

</dependencies>

<!--

protocol buffers编译插件项

protoc的版本要和protobuf-java jar包的版本一致

-->

<build>

<extensions>

<extension>

<groupId>kr.motd.maven</groupId>

<artifactId>os-maven-plugin</artifactId>

<version>1.5.0.Final</version>

</extension>

</extensions>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.xolstice.maven.plugins</groupId>

<artifactId>protobuf-maven-plugin</artifactId>

<version>0.5.1</version>

<configuration>

<protocArtifact>com.google.protobuf:protoc:3.5.1-1:exe:${os.detected.classifier}</protocArtifact>

<pluginId>grpc-java</pluginId>

<pluginArtifact>io.grpc:protoc-gen-grpc-java:1.14.0:exe:${os.detected.classifier}</pluginArtifact>

</configuration>

<executions>

<execution>

<goals>

<goal>compile</goal>

<goal>compile-custom</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

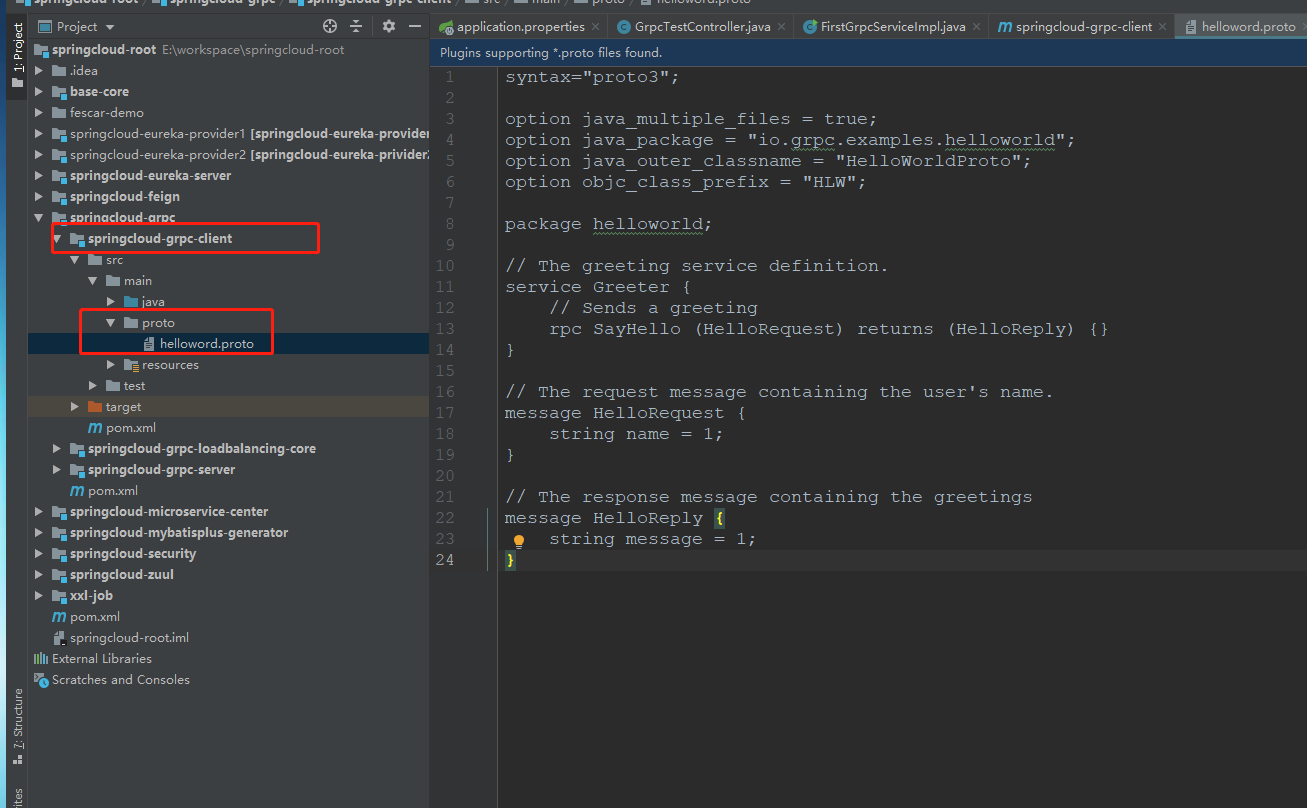
</build>

</project>

服务端是引入grpc-server-spring-boot-starter,而客户端是引入grpc-client-spring-boot-starter

添加完依赖和插件就可以在Maven projects中看到添加的依赖了。

1. 直接把服务端的proto文件复制到客户端中



1. 跟服务端同样的方式生成proto文件对应的java文件

4、编写客户端代码：然后运行main方法即可提供服务。

package com.xiongjun.grpc.client.grpcservice;

import io.grpc.ManagedChannel;

import io.grpc.ManagedChannelBuilder;

import io.grpc.examples.helloworld.GreeterGrpc;

import io.grpc.examples.helloworld.HelloReply;

import io.grpc.examples.helloworld.HelloRequest;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class FirstGrpcServiceImpl {

private static final Logger logger = LoggerFactory.getLogger(FirstGrpcServiceImpl.class);

private ManagedChannel managedChannel;

private GreeterGrpc.GreeterBlockingStub blockingStub;

public FirstGrpcServiceImpl(String host, int port) {

this(ManagedChannelBuilder.forAddress(host, port).usePlaintext(true));

}

public void sendMessage() {

HelloRequest request = HelloRequest.newBuilder().setName("zhangsan").build();

HelloReply response = blockingStub.sayHello(request);

System.out.println("收到服务端消息："+response.getMessage());

}

FirstGrpcServiceImpl(ManagedChannelBuilder<?> channelBuilder) {

managedChannel = channelBuilder.build();

blockingStub = GreeterGrpc.newBlockingStub(managedChannel);

}

public static void main(String[] args) {

String host = "127.0.0.1";

FirstGrpcServiceImpl client = new FirstGrpcServiceImpl(host, 8888);

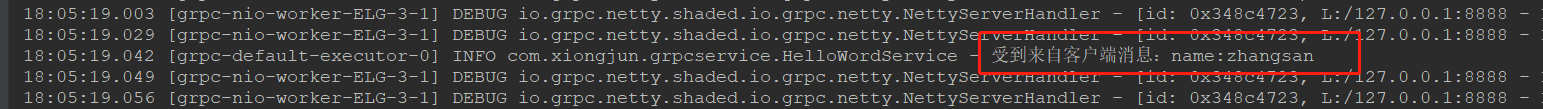
client.sendMessage();

}

}

验证结果：

服务端：



客户端：

