**广 州 商 学 院**

**实验报告（第 2 次）**

实验名称 微服务之间的通信GRPC

实验时间 2025.03.18

同组同学 无 小组分工 独立完成

**一、实验目的**

1.掌握GRPC的基本原理：理解 GRPC 的概念、原理、应用场景。理解gRPC的核心思想，包括其基于HTTP/2协议、使用Protocol Buffers作为接口定义语言（IDL）的特点。学习gRPC相较于传统RESTful API在性能、效率和跨语言支持方面的优势，特别是在处理复杂数据结构和高并发场景下的表现。

2.熟悉Protocol Buffers协议语言：通过实际操作，掌握如何使用Protocol Buffers定义服务接口和消息格式，并生成对应的语言代码（如Golang）。学习如何配置和启动GRPC服务器，编写客户端代码进行服务调用，以及如何处理错误和异常情况。

3.熟悉GRPC通信开发流程：通过实验，更加深刻体会微服务之间的通信grpc的开发流程。

**二、实验仪器设备或材料**

1.计算机：运行Windows、Linux或macOS操作系统的计算机。

2.Go语言环境：已安装并配置好Go语言环境，包括GOPATH、GOROOT等环境变量的设置。

3.文本编辑器或IDE：GoLand等，用于编写Go代码。

4.MySQL环境：用于存储Casbin策略。

5.Protobuf 序列化协议语言：已安装好，并安装好生成golang语言的工具。

**三、实验原理**

1.GRPC基础架构

GRPC是一种高效的远程过程调用（RPC）框架，它基于HTTP/2协议，并使用Protocol Buffers作为接口定义语言（IDL）。GRPC支持多种编程语言，使得跨语言开发变得非常方便。在本实验中，我们将使用Golang来实现客户端和服务端。

2.HTTP/2：GRPC利用HTTP/2的优势，如多路复用、头部压缩和服务器推送，从而提供更高效的网络传输。

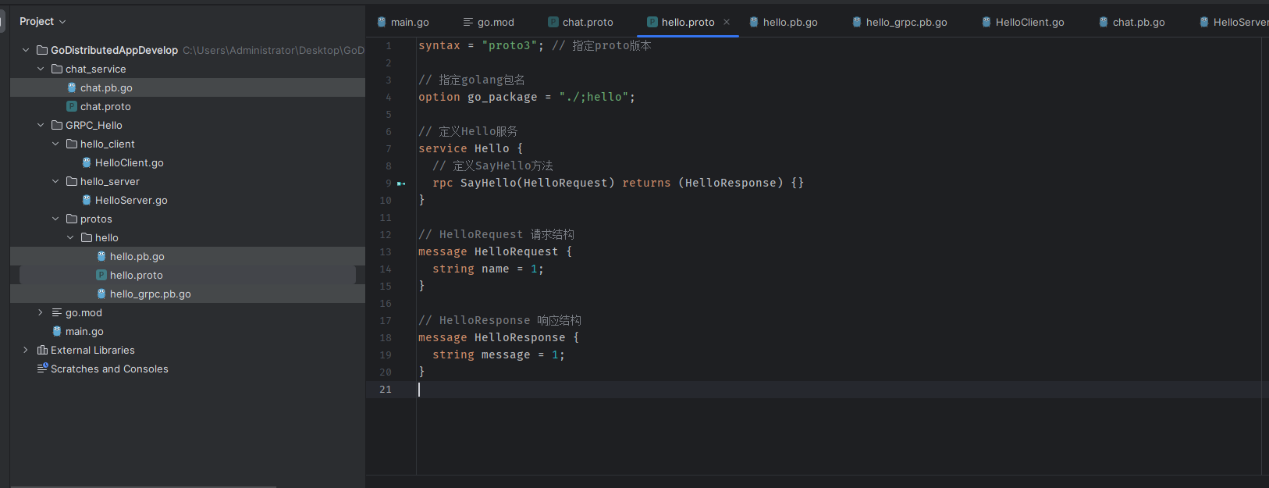
3.Protocol Buffers：用于定义服务接口和消息格式。Protocol Buffers生成的代码可以自动处理序列化和反序列化，简化了数据传输的过程。

**四、实验内容与步骤**

1.下载proto win版本



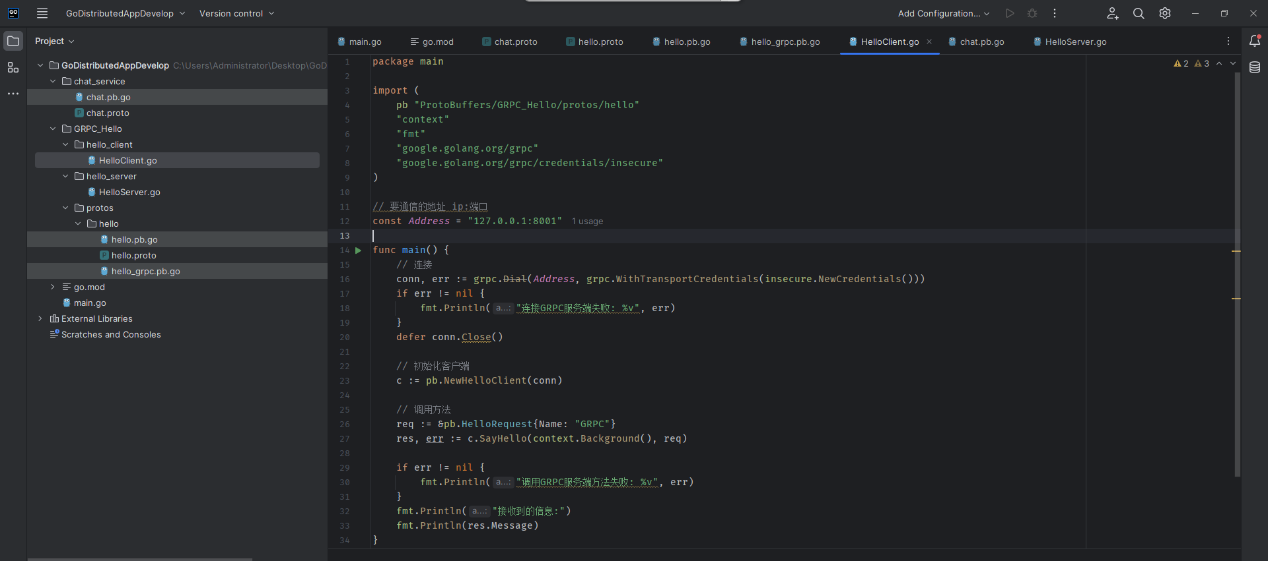
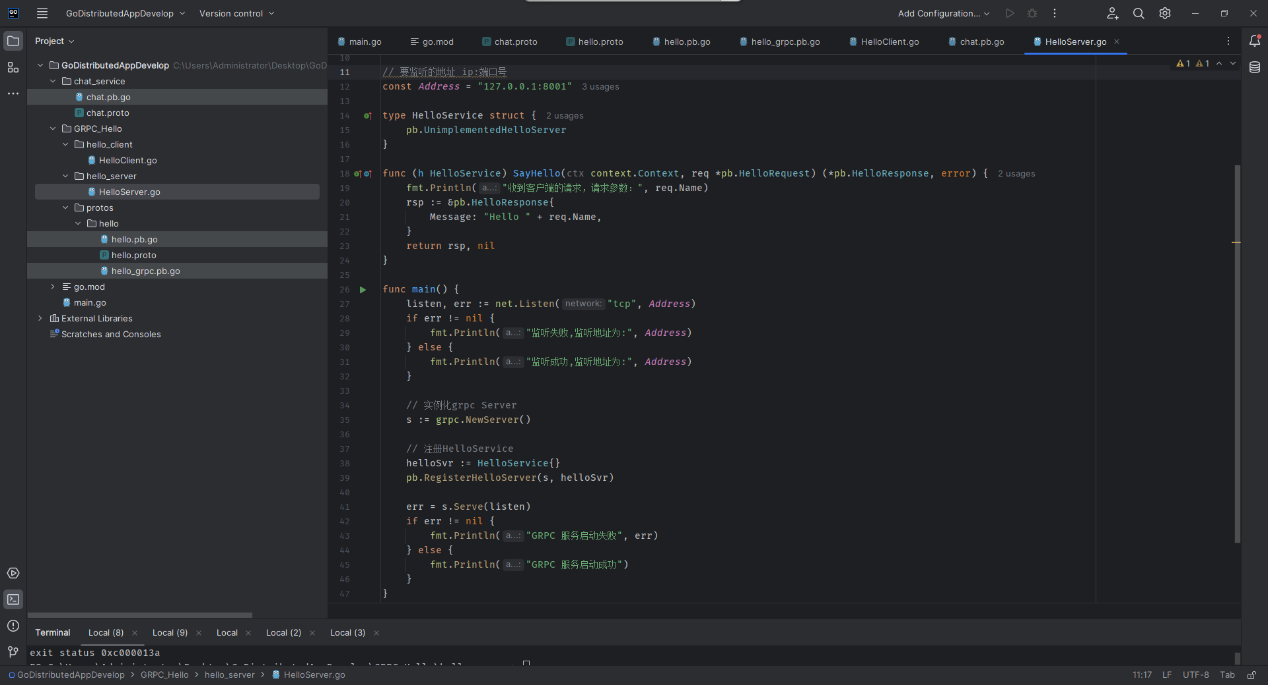
2.建立相关包，并创建Hello.proto



3.创建go.mod并根据获取导入包所需的包文件



4. 创建HelloClient.go和HelloServer.go



**五、实验结果与分析**

1.启动服务



2.启动客户端



3.返回服务端查看内容



**六、结论与体会**

本次关于微服务之间通信 GRPC 的实验，让我收获满满。我深入理解了 GRPC 基于 HTTP/2 协议和使用 Protocol Buffers 作为接口定义语言的原理，以及其在性能、跨语言支持方面的优势。在实践过程中，学会了使用 Protocol Buffers 定义服务接口和消息格式，并能成功搭建 GRPC 服务器和客户端进行通信。

不过，在配置环境和处理代码细节时遇到了一些难题，比如 proto 文件的配置问题，但这也锻炼了我的问题解决能力。后续我会继续深入研究 GRPC，探索其在更复杂场景下的应用，提升自己的开发技能。

**七、教师评语**