lukex Lv1
2021年07月22日 阅读 246

Grpc初尝试

step1: 安装必要工具

shell 复制代码

```shell
go get -u google.golang.org/protobuf/cmd/protoc-gen-go
go install google.golang.org/protobuf/cmd/protoc-gen-go

go get -u google.golang.org/grpc/cmd/protoc-gen-go-grpc
go install google.golang.org/grpc/cmd/protoc-gen-go-grpc
```

step2: 在 product 目录下编辑 ProductInfo.proto 文件:

proto 复制代码

```proto
syntax ="proto3";

package product;

option go_package=".;product";

message Product {
        string id =1;
        string name=2;
        string description=3;
};

message ProductId {
        string value =1;
};
service ProductInfo {
        rpc addProduct (Product) returns (ProductId);
        rpc getProduct (ProductId) returns (Product);
};
```

step3: 在 product 目录下(也就是 ProductInfo.proto 文件相同目录)执行如下命令:

```
# 生成service的定义
protoc --go-grpc_out=./  --go-grpc_opt=paths=source_relative  ProductInfo.proto
```

step4: grpc server代码

go 复制代码

```go
type server struct {
        productMap map[string]*product.Product
        product.UnimplementedProductInfoServer
}

func (s *server) AddProduct(ctx context.Context, req *product.Product) (resp *product.Produ
        resp = &product.ProductId{}
        out, err := uuid.NewV4()
        if err != nil {
                return resp, status.Errorf(codes.Internal, "err while generate uuid", err)
        }
        req.Id = out.String()
        if s.productMap == nil {
                s.productMap = make(map[string]*product.Product)
        }
        s.productMap[req.Id] = req
        resp.Value = req.Id
        return
}

func (s *server) GetProduct(ctx context.Context, req *product.ProductId) (resp *product.Pro
        if s.productMap == nil {
                s.productMap = make(map[string]*product.Product)
        }
        resp = s.productMap[req.Value]
        return
}
```

server main.go代码:

go 复制代码

```go
package main

import (
   "context"
   "github.com/gofrs/uuid"
   "google.golang.org/grpc/codes"
   "google.golang.org/grpc/status"
```

```
    "net"
)
const port = ":9527"

func main(){
        listener, err := net.Listen("tcp", port)
        if err != nil {
                log.Println("net listen err ", err)
                return
        }
        s := grpc.NewServer()
        product.RegisterProductInfoServer(s, &server{})
        log.Println("start gRPC liston on port" + port)
        if err := s.Serve(listener); err != nil {
                log.Println("failed to serve...", err)
                return
        }
}
```

step5: grpc client代码

go 复制代码

```
package main

import (
        "context"
        "generalTest/test/grpcTest/product"
        "log"

        "google.golang.org/grpc"
)

const address = "127.0.0.1:9527"

func AddProduct(ctx context.Context, client product.ProductInfoClient) (id string) {
        aMac := &product.Product{Name: "MacBookPro 2021", Description: "From Apple Inc."}
        productId, err := client.AddProduct(ctx, aMac)
        if err != nil {
                log.Println("add product fail.", err)
                return
        }
        log.Println("add product success,id=", productId.Value)
        return productId.Value
}
```

```
                log.Println("get product,err.", err)
                return
        }
        log.Printf("get product success: %+v\n", p)
}

func main(){
        conn, err := grpc.Dial(address, grpc.WithInsecure())
        if err != nil {
                log.Println("dial connect fail.", err)
                return
        }
        client := product.NewProductInfoClient(conn)
        ctx := context.Background()

        id := AddProduct(ctx, client)
        GetProduct(ctx, client, id)
}
```

可能遇到的错误:

- 错误一: `protoc-gen-go-grpc: unable to determine Go import path for "ProductInfo.proto"`

原因: `ProductInfo.proto` 文件中需要定义 `option go_package=".;product"`

- 错误二: `--go_out: protoc-gen-go: plugins are not supported; use 'protoc --go-grpc_out=...' to generate gRPC`

原因: 新版本 `protoc` 已经不支持 `--go_out=plugins=grpc:` 。只能使用 `protoc --go-grpc_out=./`

参考问题:

protoc-gen-go-grpc: program not found or is not executable
最简单的 gRPC 教程— 1 初识 gRPC

文章分类 后端 文章标签 protobuf