Русский

Урок gRPC

Создание gRPC сервера

Давайте создадим файл `main.go` в папке `cmd/server`. Для начала импортируем туда сгенерированный на предыдущем шаге пакет и пакет `grpc`:

```
package main

import (
    "google.golang.org/grpc"
    pb "github.com/my-name/grpc-service-example/proto"
)
```

Далее создадим структуру `Server`, которая будет содержать в себе `pb.GeometryServiceServer`:

```
type Server struct {
    pb.GeometryServiceServer // сервис из сгенерированного пакета
}

func NewServer() *Server {
    return &Server{}
}
```

Теперь нам нужно посмотреть, какие методы интерфейса `GeometryServiceServer` нам нужно реализовать. Для этого найдём в файле `proto/geometry_grpc.pb.go` определение интерфейса (protoc сгенерировал их на основе `GeometryService` из `proto` файла):

```
// GeometryServiceServer is the server API for GeometryService service.
// All implementations must embed UnimplementedGeometryServiceServer
// for forward compatibility
type GeometryServiceServer interface {
    Area(context.Context, *RectRequest) (*AreaResponse, error)
    Perimeter(context.Context, *RectRequest) (*PermiterResponse, error)
    mustEmbedUnimplementedGeometryServiceServer()
}
```

Реализуем методы для подсчёта площади и периметра:

```
func (s *Server) Area(
    ctx context.Context,
    in *pb.RectRequest,
) (*pb.AreaResponse, error) {
    log.Println("invoked Area: ", in)
    // вычислим площадь и вернём ответ
    return &pb.AreaResponse{
        Result: in.Height * in.Width,
    }, nil
}

func (s *Server) Perimeter(
    ctx context.Context,
    in *pb.RectRequest,
) (*pb.PermiterResponse, error) {
    log.Println("invoked Perimeter: ", in)
    // вычислим периметр и вернём ответ
    return &pb.PermiterResponse{
        Result: 2 * (in.Height + in.Width),
    }, nil
}
```

А теперь запустим сам сервер:

```
func main() {
    host := "localhost"
    port := "5000"

    addr := fmt.Sprintf("%s:%s", host, port)
    lis, err := net.Listen("tcp", addr) // будем ждать запросы по этому адресу

    if err != nil {
        log.Println("error starting tcp listener: ", err)
        os.Exit(1)
    }

    log.Println("tcp listener started at port: ", port)
    // создадим сервер grpc
    grpcServer := grpc.NewServer()
    // объект структуры, которая содержит реализацию
    // серверной части GeometryService
    geomServiceServer := NewServer()
    // зарегистрируем нашу реализацию сервера
    pb.RegisterGeometryServiceServer(grpcServer, geomServiceServer)
    // запустим grpc сервер
    if err := grpcServer.Serve(lis); err != nil {
        log.Println("error serving grpc: ", err)
        os.Exit(1)
    }
}
```

Можно проверить, запускается ли наш сервис:

```
go run ./cmd/server/main.go
```

Справка