微服务框架开发三

联系QQ：2816010068，加入会员群

二叉树出品

目录

- 共性抽象
- 代码生成工具设计
- 代码生成工具开发

共性抽象

```go
import (
    "log"
    "net"

    pb "github.com/ibinarytree/koala/example/grpc_example/hello"
    "golang.org/x/net/context"
    "google.golang.org/grpc"
)

const (
    port = ":50051"
)

type server struct{}

func (s *server) SayHello(ctx context.Context, in *pb.HelloRequest) (*pb.HelloResponse, error) {
    return &pb.HelloResponse{Reply: "你好 " + in.Name}, nil
}

func main() {
    lis, err := net.Listen("tcp", port)
    if err != nil {
        log.Fatal("failed to listen: %v", err)
    }
    s := grpc.NewServer()
    pb.RegisterHelloServiceServer(s, &server{})
    s.Serve(lis)
}
```

共性抽象

- 存在的问题
  - 每个服务都需要从零写代码
    - 重复无意义的工作
    - 代码风格不统一
    - 开发效率低
  - 缺乏抽象，扩展性差
    - 不支持中间件
    - 用户定制化差
    - 代码维护化差

代码生成

- 脚手架自动生成
  - 开发效率高
    - 无需从零开发
  - 代码风格统一
    - 服务可维护性高
    - 代码质量高

自动代码生成工具设计

- 服务目录规范
  - controller：存放服务的方法实现
  - idl：存放本服务的idl定义
  - main：存放服务的入口代码
  - scripts：存放服务的脚本
  - conf：存放服务的配置文件
  - app/router：存放服务的路由
  - app/config：存放服务的一些配置
  - model：存放服务的实体代码
  - generate：grpc生成的代码

自动代码生成工具设计

- 核心流程
  - 生成项目的目录
  - 使用官方工具，生成generate目录下的grpc官方代码
  - 解析proto3语法的idl文件，拿到所有的元数据
  - 根据元数据，生成controller目录下的接口实现代码
  - 根据元数据，生成main目录下的入口
  - 根据元数据，生成scripts目录下的启动脚本
  - 根据元数据，生成conf目录的配置文件
  - 根据元数据，生成app/router目录下的路由代码
  - 根据元数据和配置文件，生成app/config目录下的配置读取代码

自动代码生成工具设计

- 代码生成工具框架搭建
  - 使用github.com/urfave/cli搭建
  - 命令行参数设计
    - -f 指定idl的文件
    - -o 指定代码生成的路径
    - -c 指定生成客户端调用代码
    - -s 指定生成服务端框架代码

自动代码生成工具设计

- 接口设计
  - 每个核心功能抽象成一个generator

```
type Generator struct {
    Run(opt *Option) error
}
```

```
type Option struct {
    Proto3Filename string
    Output         string
    GenClientCode  bool
    GenServerCode  bool
}
```

自动代码生成工具设计

- 生成项目的目录
  - 关键点：使用os.Mkdir或os.MkdirAll创建

自动代码生成工具设计

- 调用官网工具生成grpc代码
  - exec.Command执行外部命令
  - protoc --go_out=plugins=grpc:. hello.proto

自动代码生成工具设计

- 生成controller的接口实现代码
  - 使用github.com/emicklei/proto解析proto3的idl文件
  - 拿到idl定义的方法和结构体等元数据
  - 遍历元数据，生成接口实现代码

自动代码生成工具设计

- 生成服务的main入口代码
  - 使用上一章的样例代码为模板进行生成