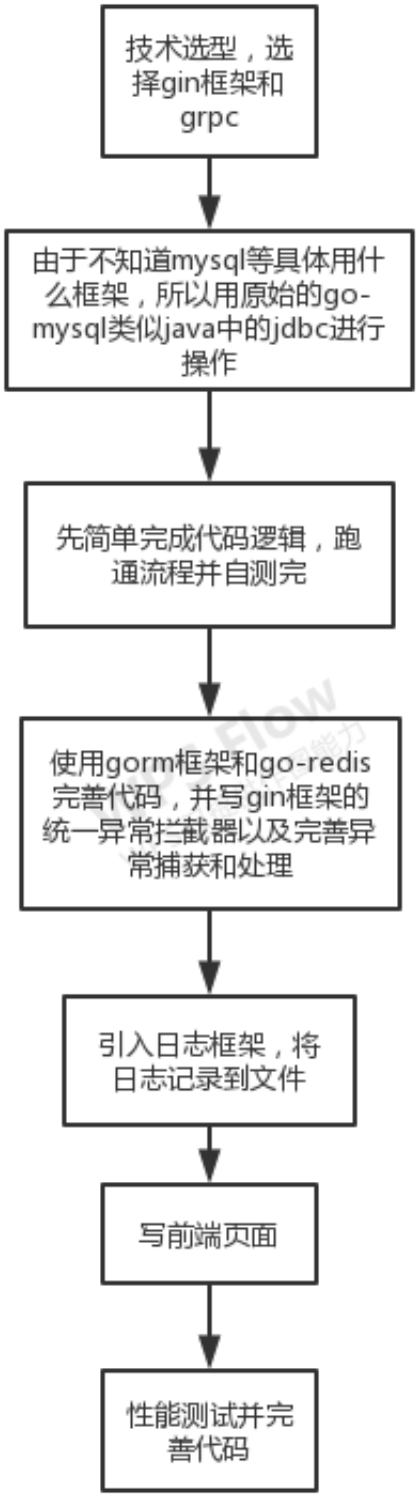
entryTask总结文档

由于没有go基础，entryTask总体用了7天，通过这次entryTask的开发，大致了解了go语言的规范以及集成mysql、redis等常用组件的方法以及调用流程，并按时完成了任务的开发，总体收获如下

- 学习了golang开发规范和gin web开发和分布式框架grpc
- 学习了golang依赖包版本控制以及引入包的方式
- 学习了golang对mysql以及redis的集成
- 学习了golang日志的配置
- 学习了jmeter性能测试

开发流程

由于不太熟，在前期投入了相对多的时间做技术选型和框架搭建 ，总体流程如下：



遇到的问题

- 不太了解golang对插件的集成，之前用spring框架时，并没有显示的main入口，而是配置一下xml文件或注解就行，但是golang需要手动在main中进行初始化操作，这个变动其实挺大的，不过万变不离其宗，其实spring最终本质有一个main入口启动并初始化插件

- logrus日志超时，刚开始用logrus作为日志系统，但是做性能测试时发现200并发下qps非常低只有几百，通过打印日志时间，发现是打日志耗时，高并发下打印一次甚至耗时几十毫秒，因此换了一个框架，用beego/logs打印日志，解决问题
- 异常捕获和处理，java中捕捉异常后可以进行处理，比如返回需要的对象，但是golang中，recover后不能进行return,最后通过在方法头中返回的信息中定义一个变量，给变量赋值，在调用端就能得到，在rpc服务端经常需要这样，因为一个接口异常不应该直接抛出而是通过接口返回result/message的形式让客户端获取然后进行相应的处理
- redis线程池配置数量太低，导致压测时timeout,调整大小就可以解决

框架总览

- gin web框架
- grpc rpc调用
- gorm orm集成
- go-redis redis集成
- beego/logs 日志系统集成
- jwt 鉴权集成

代码结构概览

代码链接为 https://git.garena.com/man.zhang/entrytask
代码目录为(以httpServer项目为例)

```
httpServer
|---README.md  //相关文档
|---Main.go    //启动类
|---main_test.go //单元测试类
|
|---config  //存放配置文件
|---|---config.go
|
|---middleware //中间件
|---|---jwt   //jwt鉴权
|---|---logger  //日志
|
|---router  //路由
|
|---servive   //接口逻辑
|---|---userServiceClient.go
```

通过将配置项、第三方(mysql,redis)、业务逻辑、路由等封装成不同的文件，达到解耦的目的

代码逻辑

- 由于entryTask是练手项目，其实逻辑比较简单，通过jwt进行鉴权，登录成功后生成token返回给前端，并将token和对应的用户信息保存起来，鉴权的时候获取header传过来的token，判断是否过期，然后接着走后续的逻辑，对于获取用户信息则从数据库获取后放入缓存下次可从缓存中获取

结果，更新用户信息后删除缓存避免前端获取到了脏数据
- 在httpServer项目中为了能够统一处理未处理的异常，写了HandlerFunc用于全局异常捕获，并统一返回给前端status 500以及result和message统一的返回信息
- 为了能够实现请求链路追踪，实现了一个HandlerFunc用于在处理业务之前，在http header中存放一个生成的X-request-Id，在业务处理的时候可以从header中取出来并打印日志，同时作为参数进行rpc调用，这样在rpc服务端打印的日志也能包含相同的requestId，同时返回给前端http header中也能看到requestId。 这种方式比较粗暴，因为对代码的侵入比较大，如果时间够的话，其实可以用jaeger等已经封装好的第三方来进行处理，避免重复造轮子
- 为了便于性能测试，开发的时候注释掉了鉴权，原有逻辑是通过token获取用户id，为了便于性能测试，测试时通过传用户id的方式进行获取id，当然原有逻辑改成必须传用户id其实也可以

待完善的地方

- 由于不太熟悉golang开发规范并且刚开始开发的时候也没有注意到golang开发规范，最后开发完时用golint代码检测，发现大量的不规范的地方，由于时间有限需要投入到需求开发，目前跳过
- 数据库设计还不够完善，没有createTime和updateTime
- 密码没有加密，应该至少在数据库中进行加密存储

对go的初步感受

- 个人觉得go启动特别快，比spring mvc框架要快很多，可能是两者的底层和特性不太一样，当然，这只是初步感受，如果项目大或者继承比较多，启动速度具体如何本人还没体验
- go语言比较简洁，相对于spring/spring mvc臃肿的配置其实要简洁很多
- 许多语法，比如struct以及指针，跟c语言有点相似
- 换汤不换药，虽然go与java有许多不同，但是对于业务流程以及对第三方的调用，其实本质都一样，毕竟逻辑都是共通的，差别只是调用方式和处理方式