View on GitHub

Sevice Computing

Principle, Technology and Architecture for building effitive, elastic and solid services on cloud

CLI 命令行实用程序开发实战 - Agenda

1、概述

命令行实用程序并不是都象 cat、more、grep 是简单命令。go 项目管理程序，类似 java 项目管理 maven、Nodejs 项目管理程序 npm、git 命令行客户端、 docker 与 kubernetes 容器管理工具等等都是采用了较复杂的命令行。即一个实用程序同时支持多个子命令，每个子命令有各自独立的参数，命令之间可能存在共享的代码或逻辑，同时随着产品的发展，这些命令可能发生功能变化、添加新命令等。因此，符合 OCP 原则 的设计是至关重要的编程需求。

任务目标

1. 熟悉 go 命令行工具管理项目
2. 综合使用 go 的函数、数据结构与接口，编写一个简单命令行应用 agenda
3. 使用面向对象的思想设计程序，使得程序具有良好的结构命令，并能方便修改、扩展新的命令,不会影响其他命令的代码
4. 项目部署在 Github 上，合适多人协作，特别是代码归并
5. 支持日志（原则上不使用debug调试程序）

2、GO 命令

GO命令 的官方说明并不一定是最新版本。最新说明请使用命令 `go help` 获取。 关于GO命令

必须了解的环境变量：**GOROOT，GOPATH**
项目目录与 gopath

2.1 go 命令的格式

使用：

```
go command [arguments]
```

版本（go 1.8）的命令有：

```
build       compile packages and dependencies
clean       remove object files
doc         show documentation for package or symbol
env         print Go environment information
bug         start a bug report
fix         run go tool fix on packages
fmt         run gofmt on package sources
generate    generate Go files by processing source
get         download and install packages and dependencies
install     compile and install packages and dependencies
list        list packages
run         compile and run Go program
test        test packages
tool        run specified go tool
version     print Go version
vet         run go tool vet on packages
```

2.2 go 命令分类

1. 环境显示： version、env
2. 构建流水线： clean、build、test、run、 （publish/git）、get、install
3. 包管理： list, get, install
4. 杂项：fmt，vet，doc，tools …

具体命令格式与参数使用 `go help [topic]`

3、准备知识或资源

3.1 Golang 知识整理

这里推荐 time-track 的个人博客，它的学习轨迹与课程要求基本一致。以下是他语言学习的笔记，可用于语言快速浏览与参考：

- 《Go程序设计语言》要点总结——程序结构
- 《Go程序设计语言》要点总结——数据类型
- 《Go程序设计语言》要点总结——函数
- 《Go程序设计语言》要点总结——方法
- 《Go程序设计语言》要点总结——接口

以上仅代表作者观点，部分内容是不准确的，请用批判的态度看待网上博客。 切记：

- GO **不是面向对象(~~OOP~~)** 的。 所谓方法只是一种语法糖，它是特定类型上定义的操作（operation）
- 指针是没有 nil 的，这可以避免一些尴尬。 `p.X` 与 `v.x` (p 指针， v 值) 在语义上是无区别的，但实现上是有区别的 `p.x` 是实现 c 语言 `p->x` 的语法糖

- zero 值好重要

3.2 JSON 序列化与反序列化

参考：JSON and Go
json 包是内置支持的，文档位置：https://go-zh.org/pkg/encoding/json/

3.3 复杂命令行的处理

不要轻易“发明轮子”。为了实现 POSIX/GNU-风格参数处理，–flags，包括命令完成等支持，程序员们开发了无数第三方包，这些包可以在 godoc 找到。

- pflag 包： https://godoc.org/github.com/spf13/pflag
- cobra 包： https://godoc.org/github.com/spf13/cobra
- goptions 包： https://godoc.org/github.com/voxelbrain/goptions
- …
- docker command 包：https://godoc.org/github.com/docker/cli/cli/command

go dead project **非常有用**

这里我们选择 cobar 这个工具。

tip: 安装 cobra

- 使用命令 `go get -v github.com/spf13/cobra/cobra` 下载过程中，会出提示如下错误

```
Fetching https://golang.org/x/sys/unix?go-get=1
https fetch failed: Get https://golang.org/x/sys/unix?go-get=1: dial tcp 216.239.37.1:443: i/o timeout
```

这是熟悉的错误，请在 `$GOPATH/src/golang.org/x` 目录下用 `git clone` 下载 `sys` 和 `text` 项目，然后使用 `go install github.com/spf13/cobra/cobra`, 安装后在 `$GOBIN` 下出现了 cobra 可执行程序。

Cobra 的简单使用

创建一个处理命令 `agenda register -uTestUser` 或 `agenda register --user=TestUser` 的小程序。

简要步骤如下：

```
cobra init
cobra add register
```

需要的文件就产生了。 你需要阅读 `main.go` 的 `main()`；`root.go` 的 `Execute()`; 最后修改 `register.go`, `init()` 添加：

```
registerCmd.Flags().StringP("user", "u", "Anonymous", "Help message for username")
```

`Run` 匿名回调函数中添加：

```
username, _ := cmd.Flags().GetString("user")
fmt.Println("register called by " + username)
```

测试命令：

```
$ go run main.go register --user=TestUser
register called by TestUser
```

参考文档：

- 官方文档 **推荐**
- golang命令行库cobra的使用 中文翻译

4、agenda 开发项目

4.1 需求描述

- 业务需求：见后面附件
- 功能需求： 设计一组命令完成 agenda 的管理，例如：
  - agenda help： 列出命令说明
  - agenda register -uUserName –password pass –email=a@xxx.com： 注册用户
  - agenda help register： 列出 register 命令的描述
  - agenda cm … : 创建一个会议
  - 原则上一个命令对应一个业务功能
- 持久化要求：
  - 使用 json 存储 User 和 Meeting 实体
  - 当前用户信息存储在 curUser.txt 中
- 开发需求
  - 团队：**2-4人**，一人作为 master 创建程序框架，其他人 fork 该项目，所有人同时开发。团队 **不能少于 2 人**
  - 时间：**两周完成**
- 项目目录
  - cmd： 存放命令实现代码
  - entity： 存放 User 和 Meeting 对象读写与处理逻辑
  - 其他目录： 自由添加
- 日志服务
  - 使用 log 包记录命令执行情况

4.2 第一周任务

1. 按 3.3 安装 cobra 并完成小案例
2. 按需求设计 agenda 的命令与参数（制品 cmd-design.md）
3. master 创建项目，提交到 github， 其他人 fork 该项目
4. 每人分别创建属于自己的命令（命令实现 Print 读取的参数即可），提交并归并。确保不同人管理不同文件，以便于协作
5. 如时间富余，请完成 User 和 Meeting 实体 json 文件读写

思考

假设你要建立项目 `testcobra`，你必须建立在 `$GOPATH\src\$your_git_account\$repo\testcobra` 目录下，才能正常实现协作。采用 golang 术语，`$GOPATH\src\$your_git_account\$repo\` 是你当前的 **工作区** 。

问题：如果你的项目不能公开代码，甚至不能使用 github，如何实现多人协作呢？

4.3 第二周任务

1. 添加 log 服务，记录用户的操作过程，以及关键的输出
2. 约定 `entity` 和 `cmd` 之间的接口服务，实现 agenda，并在 `README.md` 文件中给出简要使用说明和测试结果
3. （**可选**）在项目中添加 `.travis.yml` 文件，并添加测试程序。让你的项目“持续集成” – “CI” 了！

思考

为什么 `cobra` 的设计，让我们构建命令行程序变的如此简单？如何持续提升我们的设计能力。

4.4 可选任务（博客）

由于课程 golang 部分非常快，项目涉及许多知识点，需要您分享经验和英文资料翻译，以帮助您的同学入门。主要包括：

- 运用面向对象的设计思想，仅使用 `pflag` 包，实现一个简版 `pcobra` 库取代 `cobra`（不需要 fork cobra）。并使得你的 agenda 项目能正常运行
- git 和 travis 在项目中的使用经验
- log 的使用与 debug 技术
- go 语言的一些较难理解的知识

分数，就是你贡献的回报！ 请在课程群博客收集提交博客 url

4.5 评价标准

- 提交与贡献均衡性与持续性（使用 Github Insights 分析）
- Readme 的质量
- 部分代码检视

附件：Agenda 业务需求

用户注册

1. 注册新用户时，用户需设置一个唯一的用户名和一个密码。另外，还需登记邮箱及电话信息。
2. 如果注册时提供的用户名已由其他用户使用，应反馈一个适当的出错信息；成功注册后，亦应反馈一个成功注册的信息。

用户登录

1. 用户使用用户名和密码登录 Agenda 系统。
2. 用户名和密码同时正确则登录成功并反馈一个成功登录的信息。否则，登录失败并反馈一个失败登录的信息。

用户登出

1. 已登录的用户登出系统后，只能使用用户注册和用户登录功能。

用户查询

1. 已登录的用户可以查看已注册的所有用户的用户名、邮箱及电话信息。

用户删除

1. 已登录的用户可以删除本用户账户（即销号）。
2. 操作成功，需反馈一个成功注销的信息；否则，反馈一个失败注销的信息。
3. 删除成功则退出系统登录状态。删除后，该用户账户不再存在。
4. 用户账户删除以后：
    - 以该用户为 发起者 的会议将被删除

- 以该用户为 参与者 的会议将从 参与者 列表中移除该用户。若因此造成会议 参与者 人数为0，则会议也将被删除。

创建会议

1. 已登录的用户可以添加一个新会议到其议程安排中。会议可以在多个已注册 用户间举行，不允许包含未注册用户。添加会议时提供的信息应包括：
   - 会议主题(title)（在会议列表中具有唯一性）
   - 会议参与者(participator)
   - 会议起始时间(start time)
   - 会议结束时间(end time)
2. 注意，任何用户都无法分身参加多个会议。如果用户已有的会议安排（作为发起者或参与者）与将要创建的会议在时间上重叠 （允许仅有端点重叠的情况），则无法创建该会议。
3. 用户应获得适当的反馈信息，以便得知是成功地创建了新会议，还是在创建过程中出现了某些错误。

增删会议参与者

1. 已登录的用户可以向 自己发起的某一会议增加/删除 参与者 。
2. 增加参与者时需要做 时间重叠 判断（允许仅有端点重叠的情况）。
3. 删除会议参与者后，若因此造成会议 参与者 人数为0，则会议也将被删除。

查询会议

1. 已登录的用户可以查询自己的议程在某一时间段(time interval)内的所有会议安排。
2. 用户给出所关注时间段的起始时间和终止时间，返回该用户议程中在指定时间范围内找到的所有会议安排的列表。
3. 在列表中给出每一会议的起始时间、终止时间、主题、以及发起者和参与者。
4. 注意，查询会议的结果应包括用户作为 发起者或参与者 的会议。

取消会议

1. 已登录的用户可以取消 自己发起 的某一会议安排。
2. 取消会议时，需提供唯一标识：会议主题（title）。

退出会议

1. 已登录的用户可以退出 自己参与 的某一会议安排。
2. 退出会议时，需提供一个唯一标识：会议主题（title）。若因此造成会议 参与者 人数为0，则会议也将被删除。

清空会议

1. 已登录的用户可以清空 自己发起 的所有会议安排。