



















Homework6k

Cobra安装

使用命令 `go get -v github.com/spf13/cobra/cobra` 下载，这个过程需要保证一定的网络通信。

此时一般会出现错误提示，新建一个目录于 `$GOPATH/src/golang.org/x`，使用 `git clone` 安装 `sys` 和 `text` 项目，最后使用 `go install github.com/spf13/cobra/cobra`，此时 `Bin` 目录下出现 `cobra` 程序








 cobra.exe	2019/10/29 20:40	应用程序	12,040 KB
 fillstruct.exe	2019/10/29 20:55	应用程序	6,883 KB
 gocode.exe	2019/10/29 20:54	应用程序	12,777 KB
 gocode-gomod.exe	2019/10/29 20:58	应用程序	12,357 KB
 godef.exe	2019/10/29 20:58	应用程序	9,377 KB
 godoctor.exe	2019/10/29 20:56	应用程序	8,083 KB
 goimports.exe	2019/10/29 20:33	应用程序	8,020 KB
 golint.exe	2019/10/29 20:59	应用程序	6,404 KB
 go-outline.exe	2019/10/29 20:54	应用程序	4,682 KB
 gopkgs.exe	2019/10/29 20:54	应用程序	5,531 KB
 goplay.exe	2019/10/29 20:55	应用程序	7,069 KB
 gorename.exe	2019/10/29 20:34	应用程序	6,393 KB
 goreturns.exe	2019/10/29 20:58	应用程序	8,656 KB
 go-symbols.exe	2019/10/29 20:54	应用程序	4,481 KB
 gotests.exe	2019/10/29 20:54	应用程序	13,261 KB
 guru.exe	2019/10/29 20:34	应用程序	9,595 KB
 hello.exe	2019/10/8 0:31	应用程序	2,028 KB
 impl.exe	2019/10/29 20:55	应用程序	8,197 KB

初始化

使用 `cobra init` 进行初始化

使用 `cobra add register` 添加 `register` 指令

使用 `cobra add login` 添加 `login` 指令

 cmd	2019/10/29 21:21	文件夹	
 cobra.exe	2019/10/29 21:21	应用程序	12,040 KB
 goimports.exe	2019/10/29 21:21	应用程序	8,020 KB
 gorename.exe	2019/10/29 21:21	应用程序	6,393 KB
 guru.exe	2019/10/29 21:21	应用程序	9,595 KB
 LICENSE	2019/10/29 21:21	文件	12 KB
 main.go	2019/10/29 22:03	GO 文件	1 KB

工作目录生成完毕

构建持久化Database

由于需要使用json进行数据持久化，首先考虑Database的搭建

首先创建一个user类和存储的列表

```
1  type User struct {
2      Name      string `json:"name"`
3      Password  string `json:"password"`
4      Email     string `json:"email"`
5      Phone     string `json:"phone"`
6  }
7
8  type Userlist struct {
9      List []User `json:"list"`
10     ID   string `json:"sid"`
11 }
```

封装添加用户函数 `func adduser(s User)`

```
1  json.Unmarshal(bytes, &u)
2  u.ID = "Base"
3  u.List = append(u.List, User{s.Name, s.Password, s.Email, s.Phone})
4
5  st, err := json.Marshal(u)
6  ioutil.WriteFile("database.txt", st, 0664)
```

类似的搭建 `func queryUser(s User) bool` 和 `func queryUserAndPassword(s User) bool` ,用于检测是否已经使用这个名字以及名字和密码是否匹配

Register功能实现

首先配置flags, register应该需要有4个参数

```
1  registerCmd.Flags().StringP("user", "u", "Anonymous", "Username")
2  registerCmd.Flags().StringP("pass", "p", "Anonymous", "Password")
3  registerCmd.Flags().StringP("email", "e", "Anonymous", "Email")
4  registerCmd.Flags().StringP("phone", "o", "Anonymous", "PhoneNumber")
```

解析参数之后，判断是否存在，如果不存在，则添加用户，否则提示错误。

```
1  if !queryUser(User{username, password, email, phone}) {
2      adduser(User{username, password, email, phone})
3      fmt.Println("register succeed. The name is", username)
4  } else {
5      fmt.Println("Already exist this user")
6  }
```

Login功能实现

配置flags, Login应该需要有2个参数

```
1 loginCmd.Flags().StringP("user", "", "Anonymous", "Username")
2 loginCmd.Flags().StringP("pass", "", "Anonymous", "Password")
```

解析参数之后，判断用户名和密码是否符合或者用户是否存在，否则提示错误。

```
1 if queryUserAndPassword(User{username, password, "", ""}) {
2     adduser(User{username, password, "", ""})
3     fmt.Println("Login succeed. The name is", username)
4 } else {
5     fmt.Println("Login failed")
6 }
```

测试

测试环境为Win10x64

```
F:\go\src\Agenda>go run main.go register --user u1 --pass p1 --email e1 --phone p1
register called
Write to database
register succeed. The name is u1

F:\go\src\Agenda>go run main.go register --user u2 --pass p2 --email e2 --phone p2
register called
Write to database
register succeed. The name is u2

F:\go\src\Agenda>go run main.go register --user u2 --pass p2 --email e2 --phone p2
register called
Already exist this user
```

可以看出能够正确的注册且能够判断是否合法。

database中的数据：



可见数据被正确的持久化

Login测试, 可见能够正确login, 并且能够判断是否合法。

```
F:\go\src\Agenda>go run main.go login --user u1 --pass p2
login called
Login failed

F:\go\src\Agenda>go run main.go login --user u1 --pass p1
login called
Write to database
Login succeed. The name is u1

F:\go\src\Agenda>go run main.go login --user u3 --pass p1
login called
Login failed
```